A Compact Free-Keyword File Structure for Author–Title–Keyword Searching. An Application to an NMR Bibliographic Database

R. GEOFF. DROMEY*

Research School of Chemistry, Australian National University, P.O. Box 4, Canberra, A.C.T. 2600, Australia

Received June 1, 1977

Construction, maintenance, and searching of a bibliographic database can be costly in terms of both storage required and file operation times. A compact derived numeric file structure is offered as a flexible and efficient alternative to conventional multilist and inverted list implementations. The new system places no limitations on the keyword set and it is also very easy to update. An implementation has been made for an NMR author–title–keywords database. A performance evaluation is given.

INTRODUCTION

An author–title–keyword file is representative of a wide ranging set of information storage and retrieval systems. Its records are short and multiply keyed. Individual keys are usually generic although some (the author keywords) may be unique. Multilist or inverted–list structures are commonly used in the search file implementations for bibliographic data.[1-5] The latter structure is often chosen for economy of storage. If the key dimension can be restricted, the hybrid inverted list file structure[2] is probably the best implementation. In the situation where it is desirable for both the key dimension and record number dimension to be open-ended, the choice of a suitable data structure is not clearcut. The systems that have been mentioned do not accommodate these open-ended requirements smoothly.

In the discussion which follows it will be shown how a compact sequential structured file can be used to handle open-ended bibliographic files. The proposed system requires less storage than other possible alternatives, and it is relatively efficient for retrieval. The major advantages of the newly proposed data structure are that it places no restrictions on the key dimension and it is easy to update.

1. KEY-TO-RECORD CORRESPONDENCE TABLES

In conventional inverted bitmaps each key is assigned a fixed length linear array of bits with the total number of bits corresponding to the total number of records on a one-to-one basis. For a particular key, bits are set for all record numbers which contain that key. Bits not reflecting record number presence are set to zero. If the ith record contains a given key, then the ith bit in the corresponding array will be set. This data structure is very efficient for Boolean operations, but at the same time it is very costly in terms of storage.

If instead of this one-to-one approach a given key-record number combination is represented by a set of bits which characterize the particular key, then a much more compact system can be constructed. In the new structure each record number is made up of a series of bit vectors that uniquely characterize the keys associated with that particular record number. A given key that occurs in two different records will have exactly the same bit vector representation in both instances. The bit vectors, which are fixed in length, need contain only a relatively small number of bits (25 is more than sufficient) to give a high protection against the possibility of key collision. The bit-vector approach has been used successfully in a chemical name retrieval system.[6] In that instance a unique bit vector was derived for each individual chemical name. Only one bit vector was needed for each record number.

* Address correspondence to Department of Computing Science, University of Wollongong, Wollongong, N.S.W. 2500, Australia.

Table I. Character Composition, Character Adjacency, and Integer Representations for Some Closely Similar Chemical Names

name	character composition	character adjacency	integer representation
6β-hydroxyprogesterone	484	10942	4850942
11α-hydroxyprogesterone	479	10828	4800828
11β-hydroxyprogesterone	480	10864	4810864
14-hydroxyprogesterone	471	10572	4720572
19-hydroxyprogesterone	476	10747	4770747
6α-methylprogesterone	437	9376	4379376
6β-methylprogesterone	438	9391	4389391

To accommodate the varying number of keys that may be associated with a flexible author–title–keyword system, some additional conventions are needed. Before these conventions are discussed, the procedure for binary representation of keys will be briefly summarized.

2. THE BINARY REPRESENTATION OF KEYS

The words that must be represented in a bibliographic file (authors, titles, keywords, etc.) are all derived from a fixed alphabet. The alphanumeric character set (1, 2, 3, . . ., 9, 0, A, B, C, . . ., Y, Z) should be sufficient to represent all keywords that are likely to be encountered. It is convenient to ascribe to keywords two numeric indices, character composition C_C, and character adjacency C_A. They are defined as:

$$C_C = \sum_{j=1}^{N} P_j$$

$$C_A = \sum_{j=2}^{N-1} P_{j-1} P_{j+1}$$

where P_j is the position of the jth character in the alphanumeric character set. Taken in combination using the formula

$$K_{AC} = C_C \times 10000 + C_A$$

(if $C_C > 1000$ then it is multiplied by only 1000), it is possible to derive integer representations in the range 0 → 10 million. Binary bit vectors can be derived from the integer representation of keys, e.g.,

$$\text{TRIPLET} \rightarrow 1702766_{10} \rightarrow 0000110011111101101101110_2$$

where C_C = 170 and C_A = 2766. The Boolean strategy for keyword retrieval (resolution) by combination of the 25 bit arrays (not bit vectors) follows directly from the keyword's binary representation. For a bit vector of only 25 bits there are more than 10 million possible binary representations. In a database, the number of keywords is likely to be only in the thousands, and so the possibility of two distinct keywords

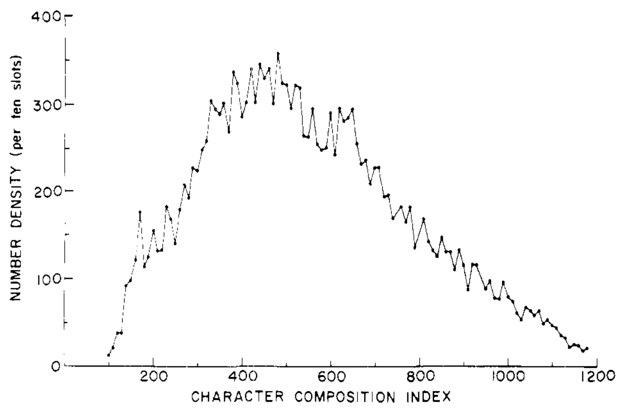

Figure 1. Plot of character composition density (C_C) for 19 689 chemical names.

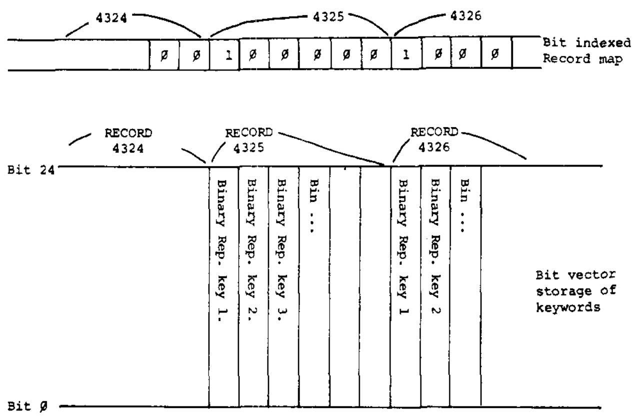

Figure 2. Scheme for bit vector storage of keywords. The record boundaries are bit indexed.

possessing the same binary representation is unlikely. The integer representations for some closely similar chemical names are given in Table I. It can be seen from Table I that if by chance two names have the same character composition index, it is extremely unlikely that they can have identical character adjacency indices. To add further to the evidence against the likelihood of collisions, a plot was made for the character composition indices of 19 689 chemical names (Figure 1). This distribution proved to be very broad with no character composition key containing more than 40 elements. Hence there is only a very small risk of collisions when character adjacencies are incorporated.

3. A BIT-INDEXED RECORD MAP

Thus far only individual key representations have been considered. It is now necessary to look at how the variable component record structure is mapped and maintained. All keys require a 25-bit "vertical" bit vector. Viewed horizontally (Figure 2) there are 25 fixed length bit arrays. The number of bits in each array is equal to the sum of all keys for all records. To allow flexibility and maximum use of storage, the file structure must accommodate variability in the number of keys (and hence bit vectors) that are associated with individual records. This can be accomplished with one additional horizontal bit array. A bit is set in the added horizontal bit array at every position corresponding to the first vertical key bit vector associated with each record. To identify the records active after a set of Boolean operations, a bit count must be performed on the bit array. This can be done efficiently with a byte lookup-table that maps from the value of a byte to the number of bits set in the byte. Greater efficiency can be obtained with a small additional table that stores the record number counts at evenly distributed positions throughout the

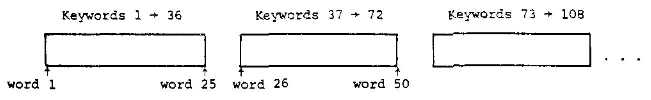

Figure 3. File organization for keyword bit vectors.

file. In addition to serving as a record pointer, the bit array can also play an important role in file maintenance.

4. FILE ORGANIZATION FOR RESTRICTED CONCEPT SEARCHES

To gain maximum flexibility and efficiency from the proposed bit vector representation, it is best to use a method of file organization that departs from the "horizontal" bit array structure implied in the preceding discussion. The "vertical" structure proposed requires that the file be organized in 25-word segments, each segment possessing a representation for as many keywords as there are bits in the computer word being used. For a computer with 36-bit word size, the first 25 words would contain bit vector representations for the first 36 keywords in the file and so on. The bit-indexed record map is still used to define the bounds of the keywords associated with individual records (see Figure 3).

There are a number of advantages that can be derived from this file structure. To add a new record in sequence it is only necessary to either update or add a localized 25-word segment. In keyword retrieval operations the file reading sequence can be overlapped with the Boolean processing by using a double buffering technique. *Boolean operations are needed on only approximately 15% of the file*[6] *to achieve keyword resolution.* The processing of a given segment is halted as soon as *all* the bit vectors in the word fail to follow the bit sequence associated with the keyword being searched for. The number of Boolean NOT operations needed for retrieval can be minimized by doing first AND operations on the bits set in the order from the most significant to the least significant bit. The NOT operations are then performed in the word order corresponding to the transition from the least significant to the most significant bit not set. It is important to appreciate that the retrieval operation being performed here is really a *restricted sequential search* (with only 15% of file being processed on the average) rather than a conventional inverted file search.

For combined keyword searches, on a relative scale, the search for the first keyword is the most costly. The results of the initial keyword search can be used to restrict the second and subsequent keyword searches. The assumption (or constraint) that no record will possess more keywords than there are bits in a computer word provides the mechanism for progressively restricting the search space within the context of variable keyword record number populations. That is, if the first keyword is found to be active in a given 25-word segment, then for the next keyword search the 25-word segments on either side must be included to ensure that there is no possibility of missing the paired-keyword combination associated with the particular record number (that is, assuming there will be no more than 36 keywords in a given record). A list must be kept of the record numbers active after the first keyword search to ensure that no false drops are introduced by using the 25-word segments on either side.

At the expense of a *local* sort on the keywords associated with each individual record as added to the file, it is possible to further reduce the amount of logic processing for combined keyword searches. If the vectors for a particular record are stored in ascending order according to their binary representations, then for the second and subsequent keyword searches it is necessary to search only the set of current 25-word segments and either those immediately preceding or succeeding the current segment in each case. The latter choice is made by comparing the magnitude of the binary representation of the previous keyword with that of the present keyword.

An author–title–keyword system has been implemented for an NMR database on a PDP 11/45. The present file contains only approximately 6000 key entries, and so it was not considered suitable for testing retrieval performance of the newly proposed file structure. To simulate a more realistic database a disk file with 360 000 dummy key entries was constructed on a UNIVAC 1108 computer. A typical keyword search was simulated by reading the file in 16 000-word segments using overlapped I/O with Boolean processing on 15% of the words in each case. Only 0.35 s of CPU time was needed for the Boolean processing. The storage required for the 360 000 key entries was only 250 000 words (36 bits each). This compares favorably with other possible search file implementations. The file implementation described is suited for retrieving documents efficiently on the basis of an author and/or restricted concept keyword search.

Unfortunately for searching a file on a much wider scale where there is a need to combine a large number of keywords on a logical OR basis, the present implementation is not nearly so attractive. There is, however, a mechanism for establishing an efficient multikeyword retrieval system that retains the desirable easy update and free keyword characteristics of the file structure that has just been described.

5. FILE ORGANIZATION FOR ONE-PASS MULTIKEYWORD RETRIEVAL

To construct an efficient one-pass multiple keyword retrieval system, it is desirable that the number of matching operations be kept to a minimum. In a conventional inverted file, multiple retrievals are accomplished by performing Boolean OR operations on individual inverted keyword lists or bit arrays.

An entirely different approach to multiple keyword searching is needed if a noninverted file (with the characteristics outlined above) is to be effective as a retrieval system. To satisfy the constraints of easy update, easy maintenance, and unrestricted keywords, the file must be constructed record by record in a sequential fashion. As before, a pairwise numeric representation of keywords can be employed. However, in this case, it is convenient to regard the character composition and character adjacency as composite numbers rather than as sets of bit patterns requiring Boolean operations for resolution.

The next step is to set up a table that will accommodate all possible character adjacencies (in this case a table of 32767 words of 16 bits). The character compositions of the set of keywords sought after (maybe 5 $\rightarrow$ 30 keywords) are then entered into the table at the locations C_A derived from their character adjacencies (i.e., the keyword TRIPLET with $C_C = 170$ and $C_A = 2766$ results in the value 170 being stored at location 2766 in the table). Should, by chance (less than 1 in 10^6 for a set of 30 keywords) two of the keywords in the set have the same C_A, then the following procedure can be used to resolve the "collision". The C_C value can be stored at the first available location at a distance which is a multiple of C_C from the original C_A address; this is closely similar to the way collisions are resolved in a hash table. The original C_A value can also be made *negative* to indicate that a collision has occurred and that the search should be pursued by way of C_C displacements. For practical purposes the risk of false drops is so small that it can be ignored.

With this table configuration the keyword search then proceeds by probing the search table with the character adjacencies of keywords in the author and the keyword file. *In this way all keywords are tested for a match simultaneously.* Since in a typical keyword search only a very small fraction of the probes are likely to involve a hit upon a nonzero location (i.e., typically only 30 out of the 32 767 locations are likely to be occupied in a given search), the processing cost beyond reading the file is very small. An implementation of this latter system (which is trivial) has been tested on a 192K Interdata 7/32 computer. For this system, character compositions have been placed in a table with a 16-bit word size. Unfortunately, the amount of data was too small to allow a meaningful performance comparison with alternative systems. However, because the amount of processing beyond that of reading in the file is very small and because the file updating and maintenance requires the minimum of software development and CPU time, it is felt that the file structure proposed may prove attractive. In particular it is suitable for applications where a workable retrieval system is needed with the minimum amount of development commitment. From a knowledge of the file access performance for a given system, a user can easily approximate retrieval times.

CONCLUSIONS

The file structure implementations described have been designed for minimum cost, practical applications. They are compact, flexible, and relatively efficient for searching and maintaining a large author–title–keyword database. No restrictions or limits are placed on the keyword set, and costly preliminary sorting of record numbers in terms of keywords is not needed. Open-ended addition of new records to the file is simple and cheap to perform. The burden of constructing and maintaining an inverted file that requires considerable CPU time and software commitments is avoided.

ACKNOWLEDGMENT

The author would like to thank Richard Bramley for providing the NMR database and Patrick Keogh for writing the UNIVAC test program. The author wishes to thank James Rush, Antonio Zamora, and the editor for their constructive comments.

REFERENCES AND NOTES

(1) C. T. Meadow, "The Analysis of Information Systems", 2nd ed, Melville, Los Angeles, Calif., 1973.
(2) D. Lefkovitz, "The Large Data Base File Structure Dilemma", *J. Chem. Inf. Comput. Sci.*, **15**, 14 (1975).
(3) D. Lefkovitz, "File Structures for On-line Systems", Spartan Books, New York, N.Y., 1969.
(4) D. R. King, "The Binary Vector as the Basis of an Inverted Index File", *J. Lib. Autom.*, **7**, 307–314 (1974).
(5) A. F. Cardenas, "Analysis and Performance of Inverted Data Base Structures", *Commun. ACM.*, **18** 253–263 (1975).
(6) R. G. Dromey, unpublished results.