A Hybrid Access Method for Bibliographic Records

Abraham BOOKSTEIN: The University of Chicago Graduate Library School, Chicago, Illinois.

This paper defines an access method for bibliographic records that combines features of the search key approach and the inverted file approach. It is a refinement of the search key technique that permits its extension to large files. A method by which this approach can be efficiently implemented is suggested.

INTRODUCTION

A major problem in the development of computerized files of bibliographic records is the creation of a convenient and economical mechanism to access the records. As the problem of organizing a file for efficient access is a general one, a number of structural devices have been suggested. Hsiao and Harary propose an abstract model for file structure that encompasses those that are discussed most frequently.[1] Lefkovitz discusses these techniques in more detail and considers the advantages of each for implementation, while Dodd and Knuth describe the data structures needed in implementing such files.[2–4]

These works reveal the interrelation between a file's organization and its retrieval capability, but the determination of which routes of access to provide must be the task of those responsible for creating the file. Such a determination may involve consideration of both the intrinsic structure of the items represented by the file and the conditions under which the file is to be used. They will influence which file organization should be chosen.

Because of the complexity inherent in collections of bibliographic items, the problem of determining suitable access routes to library files has been a challenging one. Almost any datum may, on some occasion, be a useful means of entering the file. Dimsdale and Heaps, in their discussion of a file structure for an on-line catalog, explicitly propose words from the title, authors, and Library of Congress call numbers.[5]

In this paper we shall consider the problem of accessing a known item by means of information contained in the author and title field. We shall concentrate on two approaches that have received much attention—the use

of a truncated search key, referred to simply as search key, and the use of Boolean expressions of key words from the title. Both of these are intended to allow a user simple entry into the file when the full field of information is long, complicated, or, perhaps, incompletely known by the user. The authors and titles of books often share these characteristics.

Each of the approaches, taken by itself, has its strengths and its weaknesses. We will discuss each technique in turn, and then suggest an elaboration of the search key technique that incorporates some features of the Boolean search technique; this combination of techniques should enable systems that are committed to the use of search keys as a primary access route to extend this technique to large files. It introduces into the search key approach some of the flexibility of the key word approach.

SEARCH KEYS

This approach defines at least one special field, the search key, for each item represented in the file, and allows retrieval of the record for an item by inputting the value of its search key.[6–8] The search key should be constructed so as to allow its evaluation from data that are available at the time of access. The main advantage of this approach as it is usually implemented has been its great simplicity—for a broad variety of materials, the key can be readily evaluated and quickly entered into the system. The most heavily discussed defect of this approach is that it will sometimes retrieve a considerable number of records to a single request. Consider, for example, these works:

1. *Rams*ay, Blanche Margaret. *Rel*ation of various climactic factors to the growth and development of sugar beets, and
2. *Rams*ey, Ian Thomas. *Rel*igious Language.

The popular (3, 3) search key, constructed by concatenating the first three letters of the author's name and the first three letters of the first significant word of the title, would represent each of these by the key RAM, REL. This defect becomes particularly severe with certain corporate entries and works such as conference proceedings. Furthermore, this difficulty can be expected to become aggravated as the file increases in size or, equivalently, as some items are given multiple search key values; the latter may be required in order to alleviate the problems inherent in having to access items with ambiguous or multiple forms of titles. Attempts to remedy the difficulty of multiple retrievals have resulted in increasingly complex keys, defeating the purpose for which this technique was originally proposed. A more complex key makes greater demands on the user, encourages mistakes on entry, and also might increase the likelihood of two individuals deriving different keys for the same item.

INVERTED FILES

In this approach, a user attempts to retrieve a record by forming a

Boolean expression of key words taken from various fields of the desired record.[9, 10] Stanford University's BALLOTS, for example, allows the user to enter the file by means of words taken from the title of a book. Two advantages of this approach as compared to the search key are that: (a) the user need not know the information required to form a search key, for example the first word of the title; and (b) the user is able to enter the system by what appears to him to be the most distinctive terms in the title, thereby minimizing false drops. Users of BALLOTS have found that because of the speed at which computers operate, usually the indexes can be manipulated and a record retrieved immediately, or in a very short period of time. Fayollat gives an estimate of two to five seconds as the response time.

The most direct way to implement this approach would be to access each record in the file and compare it to the request. For any but the smallest files this would be unreasonably costly in computer time. An alternative, and customary, implementation involves maintenance of indexes of key words. While experience with this approach, as at the BALLOTS project, recommends this as a workable implementation, it can be costly in terms of the computer costs involved with upkeeping the indexes.

HYBRID APPROACH

We offer for consideration an elaboration of the search key approach that incorporates aspects of the key word approach. It is intended as an alternative to developing increasingly complex keys for systems adopting a search key approach, but for which a simple search key retrieves too many items; possibly this approach can be selectively applied to the more troublesome parts of the file, such as to items with corporate authors. This approach associates a search key with each record, hopefully one that is simple and easily derived. A user would begin by entering into the system the search key. If the system finds that the number of items that would be retrieved exceeds a preset threshold, it would output a message requesting that the user enter a set of key words taken from various fields in the records; the title would be very useful in this regard. The system first generates a subfile of records having the desired search key. If a hashing technique is used, constructing this subfile can be accomplished quickly and at relatively little cost in space for tables.[11] Once the smaller file is formed, a complete search of the full records can be made for the key words. Since the system operates in two phases, it is less sensitive to the number of records the search key retrieves as far as user considerations are concerned. Ease of use becomes the dominating objective in designing the search key. Experience to date suggests that even a very simple search key will almost always produce less than thirty records with files having in the order of 100,000 records. However, a complete search of a reduced file of thirty records should be feasible; in fact, usually the subfile will be no larger than two or three records. From one point of view, in the hybrid system, we can

think of the search key not as an access mechanism, as earlier, but rather as a file reduction mechanism. This system trades the cost of maintaining and storing large indexes for an increase in costs of computer processing; only relatively easily maintained hash tables for fixed length search keys need be maintained. An accurate assessment of these costs can be made only after the statistical characteristics of various search keys have been explored.

OBSERVATIONS

If it should be desired to implement a hybrid system, the following observations would be in order:

1. Among the current concerns of facilities with large bibliographic files is file compaction. If records will have to be searched for key words, this consideration will influence planning of compaction techniques. For example, a technique such as COPACK, which completely scrambles the bits in a record, would not be permissible.[12] Use of variable length codes for characters, such as in Hoffman coding, would allow searches for key words; most likely such a search would be implemented by attempting to match substrings of bits rather than matching on the full word level.[13] Another common compaction technique, bigram coding, would also complicate the separation of words unless the blank were prevented from combining with other characters; because of the frequency with which the blank occurs with other characters, this restriction would interfere considerably with the efficacy of the technique.[14] A different approach would be to recognize that each word could have only two "spellings," depending on what happened to the blank preceding the word, and both spellings could be tested. (A brief survey of the above compaction techniques has been conducted by Fouty.[15])
2. Though a complete search for key words would be feasible on a small file, it is possible to expedite the search considerably by means of a technique devised by Malcolm Harrison, which involves adding a fixed number of bits, or signatures, to each field on which a search can take place; these additional bits are derived in a well-defined way from the original field.[16, 17] This subfield is a fixed-size representation of the full field in a form that can be used to very rapidly eliminate most records which would not pass the key word matching test. It is stored in the index to the file along with the address of the record. Though this preliminary test is not foolproof, it could considerably reduce the size of the subfile that requires a more costly complete search, thereby reducing the number of disc accesses. If this procedure is adopted, a possible sequence of events would be as follows:
   (a) A user inputs a search key and, perhaps, a couple of key words. These may be words he is certain are in the title, although the

name of a series, the author, or subject headings would also represent candidates.

(b) On the basis of the search key the system creates a subfile of record addresses and signatures taken from the index—if the user is unfortunate the subfile would have a large number of records.

(c) A rapid preliminary search of the signatures using the Harrison technique is made of the reduced file to test whether the key words could possibly be part of a record. This pass eliminates a number of records; how efficient this technique is will depend on the number of bits the system associates with each representative field.

(d) Finally, the full records of the remaining items are retrieved and a full search is made.

At any point, if the subfile is too large, the system may request additional key words.

EXAMPLE OF TECHNIQUE IMPLEMENTATION

How to create a signature for a record is best explained by means of an example. Many variants are possible, and we have chosen a simple one for the purposes of illustration. The signature we shall create will consist of one word of thirty-two bits. We proceed as follows:

1. List all the substantive words of the title, e.g., Relation, various, climactic, . . . , beets, if we consider one of the titles mentioned above.
2. Truncate each word to, say, the first four characters: Rela, vari, . . . , beet. Other truncation sizes, or no truncation at all, may be elected.
3. For each string of characters produced in this way, form the two consecutive strings of three characters. For example, "vari" contributes "var" and "ari." Since the first word is already represented in the search key, we may use only the second three-letter string for that word—here "Rela" is represented only by "ela." Implicit in this implementation is the assumption that if a user remembers anything about a word, he will correctly remember at least its first three characters, and that the first four characters go a long way toward giving the word away.
4. Finally, we turn on a bit in the signature for each three-character string, essentially creating a hash code of thirty-two bits. The code should incorporate information from all three characters. For purposes of illustration, the following method will suffice: (a) for each letter in a three-letter string, substitute the rank of that letter in the alphabet, beginning with 01 for a—thus "ela" becomes 05,12,01; (b) consider the string of digits as a single six-digit number, and multiply that number by 1111—thus "ela" becomes 51201 and then 56884311; (c) divide by 32 and use the remainder as the address of the bit which is to be turned on. The string "ela" is thus associated with bit number 23, where the leftmost bit is the 0^{th} bit. As the algorithm is

applied to each three-character string, the signature is formed. The book by Blanche Margaret Ramsay is accordingly represented by:

01000011100100011000010100100101

Similarly the book by Ian Thomas Ramsey is represented by:

00000000000000010000000001000010

Suppose a patron, or a cataloger, wishes to see the record associated with Mr. Ramsey's book on religious language. He would enter the search key, RAM,REL, and, say, the word "language." Among the index entries retrieved by the search key will be the desired book, and also the book by Ramsay, dealing with sugar beets. The signature for the word "language" has bits numbered 30 and 25 turned on. Since the Ramsay book does not have both of these turned on (in this case neither bit is turned on), it is immediately eliminated; the actual records retrieved from the file will be only those for which both bits are on. Though it is quite possible that false drops can be incurred in this way, clearly many incorrect records are easily eliminated. Note also that the user need input only as much of the word as he has confidence in, provided that at least three characters are produced.

Use of the above technique leaves a number of decisions that still must be made by the system designer. Among these are:

1. Should a signature be associated with each item, or only a part of them, for example, with corporate authors?
2. How much truncation is appropriate, if any? If no truncation is used, then the user can input fragments of words, including fragments taken from the middle of a word, as well as full words. On the other hand, as the signature fills up, the probability of a false drop increases.

Earlier research contains a formula that allows us to estimate this effect.[18] Consider a title with six significant words. Fayollat has found that in a file of biomedical serials, about 83 percent of all items will be of this size or less.[19] Similarly, let us assume that the average word in the title is made up of eight characters, a modal number of characters in Fayollat's data base. If the user requests a term composed also of eight characters, then Table 1 estimates the probability of a false drop as a function of the truncation size.

Table 1. Probability of False Drops as Function of Truncation Size.

Truncation length	*Probability of false drop*
3	.17
4	.10
5	.08
6	.08
7	.08
8	.09

It is seen that for this typical case, the method eliminates about 90 per-

cent of the false drops. It must be understood that longer titles, or titles made up of longer words, will be more likely to be erroneously retrieved; on the other hand, the user can increase his precision by inputting a larger number of terms. The above calculation assumes that terms in the request and in the title are independent; of course, all items having the same search key as the request and sharing the discriminant word will be retrieved; presumably the user will minimize this effect by choosing distinctive words. Fayollat finds that 50 percent of the words appearing in his titles occur only once.

CONCLUSION

In conclusion, we propose a technique for entering a bibliographic data base that retains the simplicity of search keys while also including some of the flexibility that Boolean expressions of key words have for uniquely defining an item. In such a system, the only indexes that must be maintained are the hash tables; the other indexes, such as title words, are replaced by the search algorithms. If a signature, the supplementary field described above, is also stored in the index, this approach reduces the number of disc accesses. A major limitation of this approach is that a user must be able to provide a search key; this is shared, however, with systems depending exclusively on search keys. Furthermore, since the system is capable of handling larger numbers of returns on the search key, there is greater inducement to associating more search key values for each item. Thus such a hybrid system allows groups that find search keys an attractive access technique to extend this approach to file sizes which strain the capacities of the direct approach.

REFERENCES

1. D. Hsiao and F. Harary, "A Formal System for Information Retrieval from Files," *Communications of the ACM* 13:67–73 (Feb. 1970).
2. D. Lefkovitz, *File Structures for On-Line Systems* (New York: Spartan Books, 1969).
3. G. Dodd, "Elements of Data Management Systems," *Computing Surveys* 1:117–35 (June 1969).
4. D. Knuth, *Fundamental Algorithms, the Art of Computer Programming,* Vol. 1 (New York: Addison-Wesley, 1968).
5. J. J. Dimsdale and H. S. Heaps, "File Structure for an On-Line Catalog of One Million Titles," *Journal of Library Automation* 6:37–55 (March 1973).
6. F. G. Kilgour, P. L. Long, and E. B. Leiderman, "Retrieval of Bibliographic Entries for a Name-Title Catalog by Use of Truncated Search Keys," *Proceedings of the ASIS* 7:79–82 (1970).
7. P. L. Long and F. G. Kilgour, "A Truncated Search Key Title Index," *Journal of Library Automation* 5:17–20 (March 1972).
8. A. Landgraf, K. Rastogi, and P. Long, "Corporate Author Entry Records Retrieved by Use of Derived Truncated Search Keys," *Journal of Library Automation* 6:156–61 (Sept. 1973).
9. James Fayollat, "On-Line Serials Control System in a Large Bio-Medical Library.

Part II. Evaluation of Retrieval Features," *Journal of the ASIS* 23:353–58 (Nov.–Dec. 1972).
10. A. H. Epstein et al., articles in *Proceedings of the ASIS* 10 (1973).
11. A. Bookstein, "Double Hashing," *Journal of the ASIS* 23:402–5 (Nov.– Dec. 1972).
12. B. A. Marron and P. A. D. De Maine, "Automatic Data Compression," *Communications of the ACM* 10:711–15 (Nov. 1967).
13. W. D. Maurer, "File Compression Using Hoffman Coding," in *Computing Methods in Optimization Problems 2,* from Second International Conference on Computing Methods in Optimization Problems (New York: Academic Press, 1969), p.242–56.
14. W. D. Schieber and G. W. Thomas, "Compaction of Alphanumeric Data," *Journal of Library Automation* 4:198–206 (Dec. 1971).
15. Gary Fouty, Unpublished Master's thesis, University of Chicago.
16. M. Harrison, "Implementation of the Substring Test by Hashing," *Communications of the ACM* 14:777–79 (Dec. 1971).
17. A. Bookstein, "On Malcolm Harrison's Substring Testing Technique," *Communications of the ACM* 16:180–81 (March 1973).
18. Ibid.
19. Fayollat, "On-Line Serials Control System."