ACCESS METHODS: A BRIEF OVERVIEW FOR THE 1971 ACM SIGFIDET WORKSHOP

by

Jack E. Shemer

Xerox Data Systems
701 South Aviation Boulevard
El Segundo, California 90245

Data Management Systems tend to be constructed upon one or more file organization techniques. An essential criterion for success of these systems is performance. Therefore, apart from considerations such as cost, compatibility, etc., the selection of the "best" system is frequently reduced to the selection of the file organization technique(1) best suited to intended applications. The algorithms employed for storing and retrieving data(2) are fundamental considerations since they directly affect mean retrieval time, insert/delete time, sequential processing efficiency, storage utilization, etc.

Typically, file organization techniques are classified in terms of their store-retrieve philosophies. Two common philosophies are: 1) the calculated access* (or hash table) method and 2) the tree structured approach. Given reasonable storage constraints, the general objective is to achieve expeditious retrieval of a record (or sequence of records) given a key (or keys) which uniquely identify said record(s).

The calculated access method(3) features an algebraic transformation T which converts a key (identifier or property) into a subfile address, where a subfile is the set of all records which, under the transformation T, yield the same address. To retrieve a record, its identifier is transformed; an address is thus obtained. The records stored at this address are examined (by key identifiers) in an effort to locate the referenced record. Sometimes the subfile is a portion of the main file, but when data records are large in relation to key-address pairs, the subfile is frequently treated as an index to the data file.

For efficiency reasons, the physical space allocated to a subfile is generally allocated a priori, hence there must be some provision to accommodate additional records which under T map into the same subfile. There are

* Often referred to as "hashing" or "scatter storage technique."

basically two approaches to this problem: 1) integrated overflow and 2) separate overflow. In the case of the former, additional records are filed in the unused space which was a priori allocated to other subfiles, in accordance with a "collision strategy" (typically, "linear probing" or "quadratic probing"); whereas with separate overflow, additional records are kept in separate space which is allocated as overflow occurs. Such overflow records are filed in accordance with a separate overflow strategy (such as "direct chaining" or "rescattering").

A tree structured file(4) can be thought of as a directed graph topology which contains no closed loops and has at most one branch entering each node. A common example of a tree structured file is one which is organized as a multi-level index structure. By this technique, k index files are maintained, each of which is constructed as a (logically) ordered set of sequential subfiles. The lowest level index file (the k^{th} level) is a logically connected set of subfiles which contains as many records as the main file. The next level, index file (k-1), serves as an index to the k^{th} index file and accordingly contains a record for each subfile of the k^{th} index file. Thus, for such a structure, the retrieval process involves k+1 steps, from the root (index file 1) through the other k-1 index files to a terminal node (main file record).

Each record is, in effect, described by k indexes (identifiers), where a data value** of each respective identifier need occur only once (in the ordered structure) to represent its occurrence in many logical records when there is a commonality of paths.

With this brief tutorial, let us focus upon the four papers of the session on "Access Methods." These deal with more particular aspects of the above topics. The paper by Nijssen deals with batch processing of files which are structured according to calculated access methods. Knott's

** The occurrence of a data value is a node in the tree structure (a branch is a directed connection to a node at the next level).

paper elaborates upon schemes for dynamic management of hash tables; whereas the paper by Rotwitt and deMaine focuses upon minimizing the total node requirements of tree structured files, while at the same time keeping the search time within tolerable limits. Finally, the paper by Bayer describes the maintenance and construction of binary B-trees. While the papers are oriented primarily toward the organization of large files, the results apply to other situations as well (in particular, the structuring of a symbol table for a compiler).

REFERENCES

1. Collmeyer, A. J., "File Organization Techniques." IEEE Computer Group News, Volume 3, No. 2, March/April 1970, pp. 3-12.

2. Collmeyer, A. J., and J. E. Shemer, "Analysis of Retrieval Performance for Selected File Organization Techniques." Proceedings 1970 FJCC, pp. 20-210.

3. Peterson, W. W., "Addressing for Random-Access Storage." IBM Journal of Research and Development, Volume 1, 1957. pp. 130-146.

4. Sussenguth, Jr. E. H., "Use of Tree Structures for Processing Files." Communications of the ACM, Volume 6, May 1963, pp. 272-279.