1975

Selection of File Organization Using An Analytic Model

S. B. Yao

A. G. Merten

Report Number:
75-151

SELECTION OF FILE ORGANIZATION USING AN ANALYTIC MODEL

by

S. B. Yao*
A. G. Merten**

CSD TR 151

Proceedings of the International Conference
on
Very Large Data Bases
September 22-24, 1975

* Computer Sciences Department - Purdue University - Lafayette, Indiana

** Graduate School of Business Administration - The University of Michigan - Ann Arbor, Michigan

SELECTION OF FILE ORGANIZATION USING AN ANALYTIC MODEL

by

S. B. Yao
Department of Computer Science
Purdue University
Lafayette, Indiana 47907

and

A. G. Merten
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48104

ABSTRACT

The problems associated with file design have recently received increased attention. One approach to their solution has been the development of file models. These models can be employed to study alternate file structures and aid the file design process. In this paper, a single model and cost function is developed to characterize most of the file structure alternatives and the selection of file structures for a design problem is automated. A file design system is developed that can be used by a file designer to select good file organizations from a large number of alternatives. The computer program which implements the model uses analytic optimization techniques to select file organizations. The output of the design system is a class of file structures specified by its average characteristics and the details of the actual file structure can be determined by simulation or other techniques.

This research was supported in part by the Air Force Office of Scientific Research, Air Force Systems Command, under Grant No. AFOSR-72-2219.

1. INTRODUCTION

Many file structures and access methods are available, and each has advantages and disadvantages with respect to particular file system objectives. No universally optimal file structure exists and no general method is available for selecting an optimal file structure for a particular application. Specific techniques are usually adequate in certain situations, but the heuristics used to match techniques to design objectives are primitive. Selection is based largely on the designer's intuition and experience, both usually qualitative and sometimes incorrect [14].

The problems associated with file design have recently received increased attention. One approach to their solution has been the development of file models. These models can be employed to study alternate file structures and aid the file design process. A number of file models have been proposed, but none of them adequately characterizes all of the essential properties of a file.

A survey by Dodd [2] concluded that file structures are constructed by combining three basic techniques: sequential, random, and list. Senko, et al., [9] and Lum, et al. [6] have modeled and studied the indexed sequential access method in detail. Hsiao and Harary [3] introduced a formal model for list oriented file structures. Lefkovitz [5] and L. Martin [7] have also modeled list oriented file structures, and have developed a set of cost equations.

A different level of analysis is the optimization of file structure design. Accurate evaluation must be achieved before optimization can be performed. Severance [10] constructed a simulation model based on a file structure component model. File structures are generated and evaluated for a given set of design requirements, and good structures are selected.

These models provide cost equations for specific file structures. To select a file structure, many models may be necessary. In this paper, a single model and cost function is developed to characterize most of the file structure alternatives and the selection of file structures for a design problem is automated. A file design system is developed that can be used by a file designer to select good file organizations from a large number of alternatives. This generalized model makes explicit the principles underlying data base construction. The computer program which implements the model uses analytic optimization techniques to select file organizations.

2. DEFINITIONS AND TERMINOLOGY

Data are symbolic descriptions which record a set of facts about some set of entities in the real or abstract world. An attribute is the name of some property of the entity to be described. A value set is a set of symbols or values which the attribute can assume. An item is an ordered pair of an attribute and a value in its value set. A record is a finite set of items with at most one item for each attribute. A file is a finite set of distinct records. A data base is a union of files.

From an external (user) point of view, the basic elements of a data base are records. Records are accessed for retrieval, modification, or deletion. In order to facilitate accessing of records within a data base, certain attributes must be made known to the user so that access requests can be described. These attributes are called retrieval attributes. A keyword is an item containing a retrieval attribute. The K-set is the set of all records containing the keyword K. Thus every keyword defines a K-set. Assuming that each record contains at least one keyword, each record is contained in at least one K-set. A retrieval request or query to the data base is a Boolean function of keywords, that is, an expression of keywords with the unary operator $\neg$ (not) and binary operators $\vee$ (or) and $\wedge$ (and). The response set of a query consists of the records satisfying the query description. The response set of a query is always obtained from the Boolean set operations on the corresponding K-sets.

3. FILE ACCESS MODEL

When a data base is accessed to retrieve a certain record, only part of the data base must be searched. This may involve table look-up, indexing, list traversing, etc. These procedures and structures are in fact partitioning the search space (records in the data base) into groups so that the search may be quickly narrowed down to smaller and smaller groups until the appropriate record is located. Such a self-refining process can be modeled by an access tree; the branches at a particular access tree node represent the alternatives which partition the search space into groups.

The immediate identifiable file search stages are attribute, keyword, and record. Attributes and keywords are specified by the query and their representations in file can be searched. Secondary data stored with these representations are then utilized to search for the appropriate records. The access paths required for this procedure constitute an access tree where the access paths are indicated by the hierarchy relationships.

Figure 1 shows the access tree of a file where the three stages are shown. Each triangular represents the set of access paths from the node at the top of the triangular to the nodes in the bottom. The following observations can be made:

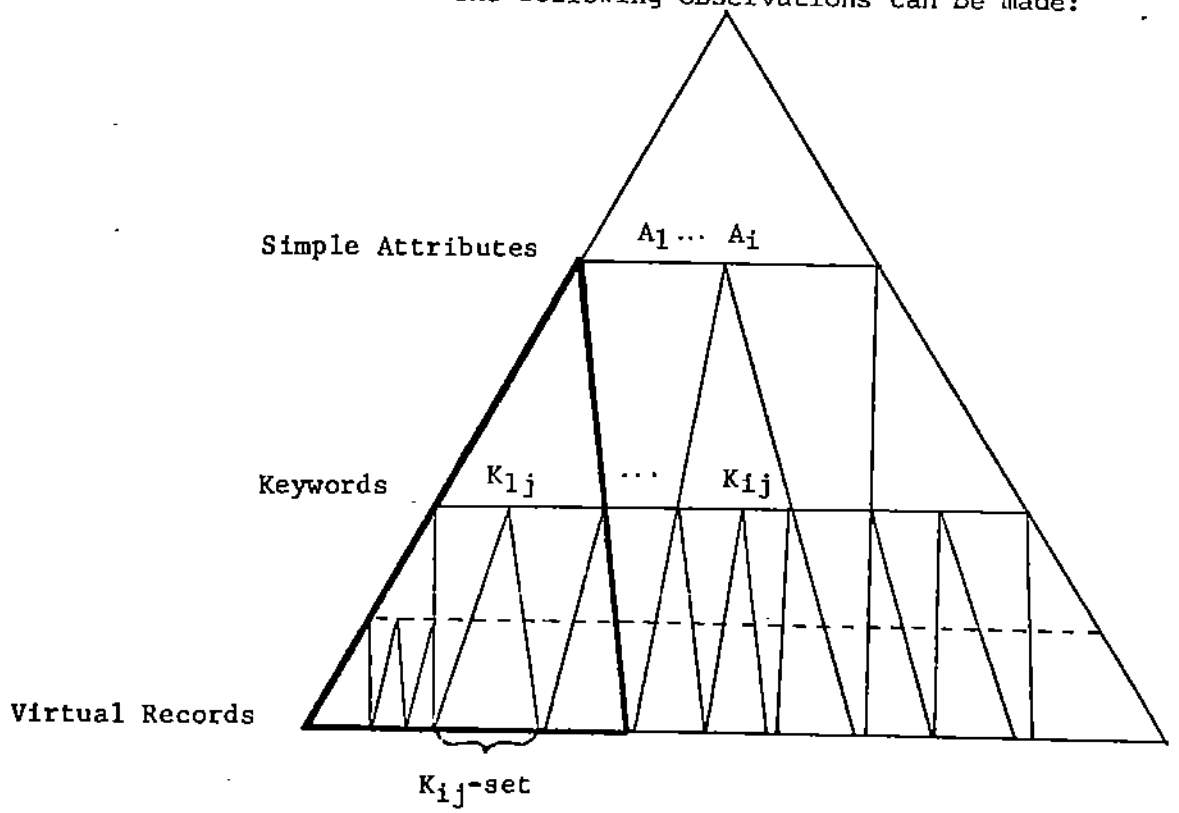

Figure 1. The access tree of a file

(1) From each attribute or keyword there are access paths leading to records containing that attribute or keyword.

(2) A record may contain more than one keyword (and hence attribute.) In order to represent the access paths by a tree, a record will be shown by as many nodes on the record level as there are access paths to it. These nodes are called virtual records.

(3) Depending on the particular file design, there may be more levels between the three identified levels. For

example, if the set of records containing a given keyword is stored as several subsets, a partition level may be introduced between the keyword level and the virtual record level. Similarly, if the keywords of an attribute is organized into a tree directory, there will be many levels between the attribute level and the keyword level.

A filial set is a set of nodes with the same immediate parent node. Any level of the access tree consists of the filial sets on that level. Let W_i denote the average size of the filial sets on level i. The entire n level access tree can be characterized by the set of variables $W_1, W_2, \ldots W_n$. This sequence of variables determines the general configuration of the access tree. It is not an exact description since the variables only represent the average filial set size on each level. From the definition of a tree, the number of nodes on level i must satisfy the relation

$$S_i = \prod_{j=1}^{i} W_j = W_1 \times W_2 \times \ldots \times W_i$$

When the size of a particular level is a given design parameter, the above identity serve as a constraint to the design of access paths.

When the access tree model is used to represent file structures, it is necessary to consider realizations of the access paths in the model. Severance [10] classified the successor connections of access paths into two types.

(1) Address sequential connection--the successor of the current node is located at the next sequential address.
(2) Pointer sequential connection--the successor node of the current node is located by a pointer field in the current node.

With this classification, the realization of the access paths on any level i of the access tree can be characterized by the pointer proportion, P_i, which is the relative proportion of type (1) and type (2) connections.

The average filial set sizes and the pointer proportions, together with the number of levels in the access tree, form the $2n + 1$ variables of the file access model:

$$n, W_1, W_2, \ldots, W_n, P_1, P_2, \ldots, P_n$$

File organizations can be represented by the file access model which is then characterized by these design variables. It was demonstrated by Yao in [12] that the access model is capable of representing most file organizations. These include sequential, indexed sequential, binary tree, double-chained tree, division hashing, multilist, cellular multilist, inverted file and variations.

The advantages of the access tree model over the previous file organization models are:

(1) The model is characterized by two sets of variables and the file design process consists of two steps: the selection of access paths (determining W_i) and the selection of realization methods for access paths (determining P_i).
(2) The model is general and can represent a wide range of file organizations with sufficient accuracy.
(3) The model makes the difference between single-attribute and multiple-attribute file organizations more explicit and hence is a more natural representation of file organizations.

4. FILE DESIGN SYSTEM

Using the generalized file access model, cost estimations of storage requirements and average retrieval and update times can be obtained. The model developed emphasizes the measurement of average storage, retrieval, and update costs. Some assumptions have been made which reflect typical aspects of the type of problem most often faced by a file designer. More precisely, these assumptions include:

(1) Single record type--only files containing homogeneous records are considered. When multiple record types occur, the file can be logically partitioned into sub-files and analyses are repeated for each sub-file.

(2) Statically formatted records--the record type is fixed for the time period under consideration. No update that modifies the record type is considered.

(3) Static data and storage characteristics--no updates that change the entire data base characteristics are considered. It is also assumed that the hardware characteristics are not changed during the time period under consideration.

(4) Well defined user activities--user activities can be measured by their types, average frequencies, and complexity.

(5) Random activities--user activities are assumed to be random and not correlated.

(6) Irredundancy--no records are stored in the data base more than once.

(7) No multiple-user interference--a file is allocated to one user for the duration of an access.

Another important assumption is that data base activities are uniformly distributed. For example, all records are assumed equal likely to be accessed. This is not always true; Knuth [4], for example, discussed the "20-80 rule" stating that 20% of a file is accessed 80% of the time. A data base, however, can be decomposed into parts such that uniformity can be assumed for each part. This can be interpreted as approximating the access distribution function by step functions.

Further assumptions that restrict the design problem are:

(1) The file design system does not address the problem of logical level data structure design nor the selection of attributes to be indexed. Logical structures and attributes are assumed given.

(2) The file structure model is characterized by average values and hence for files with large variance in structure, there will be an error in the approximation.

Figure 2 is an input/output representation of the file design system. The file design system accepts three types of inputs: data parameters, user parameters and storage parameters. The details of these parameters are shown in Table 1.

The $2n + 1$ variables of the access tree model form a $2n + 1$ dimensional space R. The access model implies a mapping $G: S \rightarrow R$ from the space of file structures S onto the space of all the combinations of model variable values, R. This mapping is usually many-to-one because the variables only represent average characteristics of the file structures, i.e. average pointer proportion and average size of filial sets of a level. At any level within the access model many different configurations of filial sets could have the same average size. The mapping is both single valued and onto. A class of file structures, corresponding to a tuple in R, is an equivalence class induced by the mapping G.

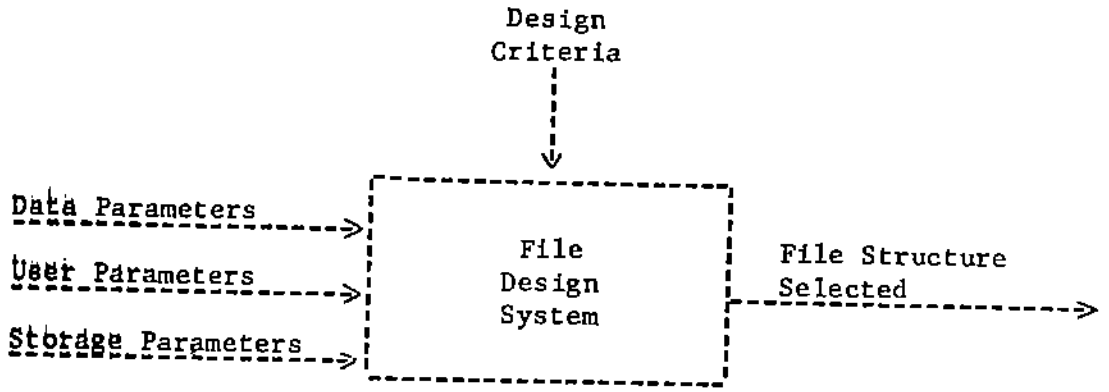

Figure 2. File design system using the access model

Table 1. File Design Parameters

(1) data parameters--
  (a) number of records
  (b) average record length
  (c) number of attributes
  (d) number of keywords
  (e) for each keyword, average number of records containing that keyword
  (f) length of indices
  (g) length of the record
(2) user parameters--
  (a) number of query retrievals per time period
  (b) average number of conjunctions in a query
  (c) average number of conjuncts in a conjunction
  (d) average size of response set for a query
  (e) average number of record insertions per time period
  (f) average number of record deletions per time period
  (g) average number of record modifications per time period
(3) storage parameters--
  (a) average random access time
  (b) average sequential access time
  (c) average transfer time
  (d) average time for comparison
  (e) length of a pointer
  (f) blocking overhead
  (g) block length
  (h) time cost
  (i) storage cost

When the cost properties of the file structures are evaluated in terms of these variables, a cost value is associated with each point in the variable space R, i.e., a function F from the variable space R into the set of real numbers R^1 is established. Hence for a given optimization criterion, the class of optimal file structure can be obtained by searching the variable space R for a point which optimizes the cost function. The result of the optimization is not a detailed file design ready for implementation but rather a guide to the file designer.

In general, the number of parameters involved in the cost function of a file is very large. For reasons of mathematical tractability, only the most pertinent

ones are selected. The data storage cost, the retrieval time to answer a typical query, and the update time can be quantified, once the characteristics of the data, the user, and the storage systems have been quantified. With these measurements, the objective function to be minimized is:

Cost per time period = $F\ (W_i, P_i, i = 1, \ldots, n) =$

$$C_s * S_t + C_t * (G_r * T_r + G_u * T_u)$$

where

C_s--unit storage cost per time period

C_t--unit access time cost

S_t--amount of secondary storage required for storing the data base

G_r--total number of retrievals per time period

G_u--total number of updates per time period

T_r--average retrieval time

T_u--average update time

The cost function estimates the performance of the file organization for a given period of time. The cost function could take the form of a product of time and storage costs such as in Sussenguth [11]. However, since the processing-time related storage (e.g., the storage of an executing program) is not the major concern, a weighted sum of time and storage costs, similar to Cardenas [1], is used. The components of the cost function, S_t, T_r and T_u are derived in terms of the design parameters in Table 1 and the $2n + 1$ file access model variables. The detailed cost function can be found in Yao [12].

With the cost function of the file access model defined, the design process is then to determine values of the design variable in order to minimize the cost function. This can be expressed as the following optimization problem.

MINIMIZE $F\ (W_i, P_i, i = 1, \ldots, n)$

subject to

$$1 \leq W_i \leq N_a \qquad i = 1, \ldots, s$$

$$1 \leq W_i \leq N_u \qquad i = s + 1, \ldots, t$$

$$1 \leq W_i \leq N_k \qquad i = t + 1, \ldots, n$$

$$0 \leq P_i \leq 1 \qquad i = 1, \ldots, n$$

$$\prod_{j=1}^{s} W_j = N_a$$

$$\prod_{j=1}^{t} W_j = N_u$$

$$\prod_{j=t+1}^{n} W_j = N_k$$

where s is the attribute level, t is the keyword level, N_a is the number of attributes, N_u is the average number of keywords per attribute and N_k is the average number of records containing a given keyword.

The above optimization problem has a non-linear objective function and non-

linear constraints. Since the non-linear constraints of this problem are in the form of a product, it is possible to make a transformation of variables and obtain linear constraints. The gradient projection method [8] was employed to solve the transformed problem. This method is based on the observation that at the minimum point some of the linear constraints will be active (tight) while the rest of the constraints are inactive. Therefore, by searching in a subspace defined by the active linear constraints, the techniques of the unconstrained optimization can be used in the subspace to locate the minimum point.

5. SELECTION OF FILE ORGANIZATIONS

The cost functions and the optimization procedure described above are implemented with computer programs. Unlike other file organization selection models which compare file organizations by individual simulations [1] or exhaustive enumeration [10], this system uses non-linear programming techniques to search through the solution space. Convergence for typical file design problems was obtained in approximately 30 seconds on an IBM 360/67. The program running time is independent of the size of the file design problem, since the design parameter values affect only the cost function coefficients.

In order to demonstrate the results of this optimization process, some test cases are constructed below. These test cases are all related to the following typical file design problem:

There are 100,000 records in the data base, each containing 10 retrieval attributes. The average size of a keyword is 10 bytes, and the average record size is 200 bytes. Each attribute may have up to 10^9 possible keywords but on the average only 10,000 keywords are active. The average K-set size is 10. There are 100 expected record retrievals, insertions and updates per time period, say, a day. An average query has 4 conjuncts, each conjunct involves 4 attributes and for each attribute there are 5 keywords in the conjunct. The expected size of the query response set is 200 records. The storage device is assumed to be an IBM 2314 with full track blocking. The weights for time and storage costs are equal; i.e., a time unit (1 millisecond) and a storage unit (1 byte-month) each cost, for example, $.001. Input parameters characterizing the design problem are listed in Table 2.

The results of the optimization system are given in Figure 3. The file organization produced consists of two parts: a hybrid multidimensional-trie-tree directory [13] and a data file. The directory obtained from the optimization process has two levels. The first level is a trie structure with an expected size of 177.8 nodes. The second level is a doubly chained tree structure with expected filial set size 56.2, and its nodes are allocated sequentially. The data file is a multi-list file structure and the average list length is 10.

Table 2. Input parameters for a file design problem

(1)	Data related parameters:	
	number of actual records in file	10,000
	number of retrieval attributes in file	10
	number of retrieval attributes per record	10
	number of possible keywords per attribute	10^9
	number of active keywords per attribute	10,000
	average number of records in a K-set	10
	length of node label in file (byte)	10
	length of record (byte)	200

Table 2. Input parameters for a file design problem (concluded)

(2)	User-related parameters:	
	frequency of record retrievals	100
	frequency of query retrievals	100
	frequency of record insertions	100
	frequency of record deletions	100
	frequency of record data updates	50
	frequency of record keyword updates	50
	frequency of new keywords	100
	number of conjunction per query	4
	number of attributes per query	4
	number of keywords per query	5
	estimated size of query response set	200
(3)	Storage related parameters:	
	average random access time	90
	average sequential access time	12
	average transfer time per byte (ms)	0.0032
	average processing time per byte (ms)	0.0046
	length of a pointer (byte)	10
	blocking overhead (byte)	146
	1st level block size (byte)	7,294
	second level block size (byte)	7,294
	cost of time per ms (cent)	0.1
	cost of storage per byte-month (cent)	0.1

Figure 3. Optimal file structure for the design problem

```
Number of levels for Multi-dimensional Index is 0
Number of levels for TRIE is 1
The expected number of nodes for M-dim/TRIE is 177.8
     Retrieval time is 35260.8
     Update time is 11753.9
     Storage requirement is 48783.3
Cost of M-dim/TRIE is 9579.8
Number of levels for TREE is 1
     The structure of TREE is
     level 1 W1 = 52.6 P1 = 0.0
     Retrival time is 37569.8
     Update time is 22003.2
     Storage requirement is 200000.0
Cost of TREE is 26016.8
Total directory cost is 35596.6
The structure of the file is
     partition level W1 = 1.0 P1 = 0.0
     record level  W1 = 10.0 P2 = 1.0
     Retrieval time is 3294648.6
     Update time is 228132.1
     Storage requirement is 30100000.0
Cost of the file is 3362278.6
     Total retrieval time is 3367479.2
     Total update time is 261889.2
     Storage requirement is 30348783.3
Total cost is 3397875.2
```

In order to investigate the effect of various input parameters to the optimal file structure, some additional test cases are constructed. Each test case varies some of the input parameters, and the effect of different parameters may be observed from the optimization result.

(1) Blocking factor

The blocking factor plays an important role in designing file structures. A smaller blocking factor usually caused an increase in the access time. When the block length is modified to 346 bytes (i.e., the blocking factor is 1) the resulting directory is more costly. It also favors more levels in the directory so that fewer blocks are required in a filial set. The doubly chained tree portion of the directory is:

Number of levels for TREE is 2
The structure of TREE is
level 1 $W_1 = 2.8$ $P_1 = 0.3$
level 2 $W_2 = 19.8$ $P_2 = 0.0$
Retrieval time is 77303.5
Update time is 37322.4
Storage requirement is 208741.1
Cost of TREE is 32336.8

Small blocks, however, require a smaller buffer, but this effect as well as other programming considerations is not measured in this optimization approach.

(2) Size of record

Increased record size has no effect on the directory structure, as indicated by the model. It does not change the structure of the file, either, since the records are unblocked and the blocking factor of record is not subjected to any change. The increased record size does have considerable effect on the cost of the data base, since more storage is required and transfer time is increased. The time spent on random access is increased too, since more blocks are required to store the records. When the record size is increased from 200 to 7,000, the total cost of file is increased in almost the same proportion. This indicates that when record size is large, the storage cost dominates the cost of the file.

(3) Density of active keywords

The density of active keywords determines the structure of the directory. When the number of active keywords is reduced from 10,000 to 200 (i.e., the density is low), the directory structure favors a TREE structure. The optimal directory structure is a two level TREE. Nodes in both levels are completely linked by pointers. At another extreme, when the number of possible keywords Np is the same as the number of active keywords Na (i.e., the density is 1) the directory structure favors a multi-dimensional index. The results of the optimization system indicate that the cost is independent of the number of dimensions used. This is because the size of the multi-dimensional index is fixed (for a given number of active keywords), and access time is always the same (one access), independent of the number of dimensions. For a moderate density the directory is usually a more general mixed structure as illustrated in the original design problem.

(4) Number of records

The number of records in a file is the most important factor in the performance of a data base. The number of records does not have as much effect on the directory as on the file. When the number of records is increased to 10,000,000, the directory is still a two level TRIE-TREE structure. The optimal structure of the file, however, changes from a multi-list file to a bounded multi-list file with a maximum average list length of 22.

(5) Size of each K-set

The size of each K-set is another important factor of file structure. When the sizes of K-sets are large, more time is spent in retrieving them for processing and hence it is more advantageous for the file to be inverted. In Figure 4 the K-set size is increased to 1,000, and the resulting optimal file structure is a fully inverted file.

(6) Frequency of retrieval

The structure for a heavily retrieved file is investigated by modifying the frequency of retrieval from 100 to 5,000. The directory is a two level TRIE-TREE. The expected size of TRIE is 1,000 and the expected filial set size for TREE is 10. Compared to the optimal structure for the original problem, this structure uses a larger TRIE level, which results in a reduction of retrieval time. The file structure is also shifted from a multi-list file to a bounded multi-list with an average list length of 3.

```
Number of levels for Multi-dimensional Index is 0
Number of levels for TRIE is 0
Number of levels for TREE is 2
      The structure of TREE is
      level 1 W1 = 9.8 P1 = 1.0
      level 2 W2 = 10.2 P2 = 1.0
      Retrieval time is 75553.4
      Update time is 26208.2
      Storage requirement is 3186.4
Cost of TREE is 10494.8
Total directory cost is 10484.8
The structure of the file is
      partition level W1 = 100.0 P1 = 0.0
      record level W2 = 1.0 P2 = 0.0
      Retrieval time is 6331012.1
      Update time is 224660.2
      Storage requirement is 33683157.4
Cost of the file is 4023882.9
      Total retrieval time is 6406565.5
      Total update time is 250868.4
      Storage requirement is 33686343.8
Total cost is 4034377.7
```

Figure 4. The effect of list length

(7) Frequency of insertion

There is a trade-off between the retrieval and the insertion frequencies. Structures favoring record retrieval usually penalize record insertion. When the frequency of insertion is high, a linked list is a better structure than an array. When the frequency of insertion is increased to 5,000, the optimal directory structure is a two-level TRIE followed by a one-level TREE structure. The file is a multi-list file, which usually facilitates efficient insertions.

6. CONCLUSION

It has been observed that all access mechanisms are gradually refining processes and that they hierarchically decompose the search space. Within a data base, the access paths of most file structures can be modeled by an access tree. The access model is described by two sets of variables. One set of variables describes the configuration of the tree and specifies an access level structure. The other set of variables determines the realization of the branches of the tree and specifies an actual level structure. The file design, then, is to determine the values of these two sets of variables.

A file design system is developed based on the access model. This design system takes its design requirement parameters for the data collection, user activities, and storage characteristics. The output of the design system is a class of file structures specified by its average characteristics and the details of the actual file structure can be determined by simulation or other techniques.

REFERENCES

1. Cardenas, A. F., "Evaluation and Selection of File Organization-- A Model and System", Comm ACM, 16, 9 (Sept. 1973), pp. 540-548.
2. Dodd, G. G., "Elements of Data Management Systems", Computing Surveys, 1 (1969), pp. 117-133.
3. Hsiao, D. and Harary, F., "A Formal System for Information Retrieval from Files" Comm ACM, 13, 2 (Feb. 1970), pp. 67-73; "Correction" Ibid., 13, 4 (April 1970), p. 266.
4. Knuth, D. E., The Art of Computer Programming, Vol. 3: Searching and Sorting, Reading, Mass., Addison-Wesley, 1973.
5. Lefkovitz, D., File Structures for On-Line Systems, New York, Spartan Books, 1969.
6. Lum, V. Y., H. Ling and M. E. Senko, "Analysis of a Complex Data Management Access Method by Simulation Modeling," Proc. 1970 Fall Joint Computer Conference, pp. 211-222.
7. Martin, L. D., "A Model for File Structure Determination for Large On-Line Data Files", Proc. of the FILE 68 International Seminar on File Organization,pp. 793-834.
8. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming, Part I, Linear Constraints", J. Soc. Indust. Appl. Math., 8, 1960, pp. 181-217.
9. Senko, M. E., V. Lum, and P. Owens, "A File Organization Evaluation Model (FOREM)", Proc. International Federation of Information Processing Societies, 1968,pp. C19-C23.
10. Severance, D. G., Some Generalized Modeling Structures for Use in Design of File Organizations, Ph.D. Dissertation, The University of Michigan, 1972.
11. Sussenguth, E. H., "Use of Tree Structures for Processing Files", Comm ACM, 6, 5 (May 1963), pp. 272-279.
12. Yao, S. B., Evaluation and Optimization of File Organizations Through Analytic Modeling", Ph.D. Dissertation, The University of Michigan, 1974.
13. Yao, S. B., "Tree Structure Construction Using Key Densities", Proceedings 1975 ACM National Conference, (Oct., 1975).
14. Waters, S. J., "File Design Fallacies", Computer J., 15 (Jan. 1972), pp. 1-4.