Application of an Analytical Model to Evaluate Storage Structures

T. J. TEOREY

The University of Michigan, Ann Arbor, Michigan 48109

and

K. SUNDAR DAS

University of Texas at San Antonio, San Antonio, Texas

Communicated by John M. Richardson

ABSTRACT

The File Design Analyzer is a software package which evaluates well-known data base storage structures and access methods in terms of secondary storage processing time and storage overhead required to service a set of user applications. It implements a first-order analytical model to specifically evaluate sequential, indexed sequential, direct access, and inverted multilist storage structures. Interaction with the package is available in conversational mode, enabling the experienced analyst to conduct on-line sensitivity analysis.

The paper describes three extensions which converted an abstract conceptual model into a practical tool for evaluation of existing or proposed data base designs: batched transactions, multi-access interference due to shared secondary storage, and analysis of variable record size. A case study from a real system illustrates the potential of the File Design Analyzer to provide insight regarding the optimal choice of physical parameters within a specified storage structure and to effectively compare alternative storage structures for a particular set of applications.

I. INTRODUCTION

The data base design process is an important part of the work of a Data Base Administrator (DBA), and an essential part of this process is the evaluation of candidate physical designs. Previous analytical approaches to physical design evaluation require separate models for different storage struc-

The research and development for the File Design Analyzer was conducted under Defense Supply Service Contract Number MDS-903-74-C-0300 for the USAF Data Services Center (AFDSC) and Grant Number AFOSR-72-2219 for the Air Force Office of Scientific Research, Air Force Systems Command.

tures [2,4]. Simulation models are also available, but they are expensive to run relative to the analytical models [1,5]. A recent breakthrough by Yao [7,8] formulates a general analytical approach to the modeling of storage structures which can be easily adapted for sequential, indexed sequential, direct access, multilist, and inverted storage structures. The concepts of record access in this model can also be applied to network storage structures. The purpose of this paper is to describe a software package, called the File Design Analyzer, that develops and extends Yao's conceptual model into a practical tool for evaluation of storage structures, and to illustrate the usefulness of this package by describing some experimental results for an existing system.

The File Design Analyzer (FDA) evaluates and ranks the most well-known data base storage structures and access methods in terms of the secondary storage processing time (access time plus data transfer time) and storage overhead required to service a set of user applications. The package can be used by the system designer to tune an existing data base by varying certain parameters and observing the effect on system response. It can also be used as a design aid for proposed data bases for which the logical requirements are well defined.

The program implements an expected value (first-order approximation) model of simple storage structures proposed by Yao [7,8]. It accepts as input, parameters which describe the secondary storage hardware characteristics, the data base description, and the user workload in terms of simple retrieval, complex query type retrieval, insertion, deletion and modification of records, and report generation. It evaluates a static configuration but it has been implemented for conversational use to facilitate obtaining multiple data points quickly and inexpensively.

Several important extensions were made to the model to provide realistic variations of the original idealized environment. They include the analysis of multiple record sizes, multiaccess interference on secondary storage, batched processing of transactions, and data reorganization for physical deletions and overflow records. These extensions are discussed below, except reorganization, which is developed in other papers [6,9].

The FDA was written in ANS/FORTRAN and is currently operational on the Honeywell 635 and IBM 370/168. It consists of approximately 1100 source statements and requires 20K words for the object program. Execution time on the 370/168 is approximately 0.1 second per data point.

II. THE MODEL

The FDA inputs and outputs are illustrated in Fig. 1. The FDA output displays I/O processing time for each type of operation specified (retrieval,

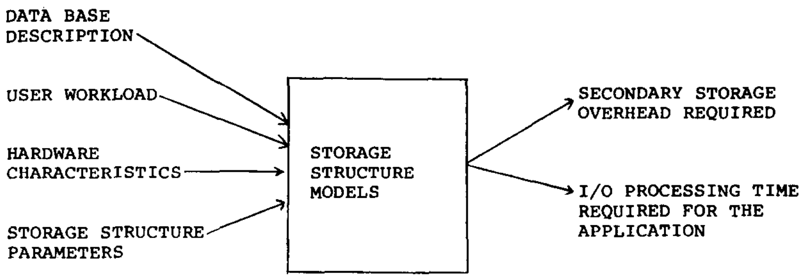
DATA BASE DESCRIPTION
USER WORKLOAD
HARDWARE CHARACTERISTICS
STORAGE STRUCTURE PARAMETERS
STORAGE STRUCTURE MODELS
SECONDARY STORAGE OVERHEAD REQUIRED
I/O PROCESSING TIME REQUIRED FOR THE APPLICATION

Fig. 1. File Design Analyzer.

insertion, etc.) the storage overhead, and the time×space product for secondary storage for the entire application.

ACCESS TREE MODEL

Access to a data base is usually made through a series of index searches or list processing. Consequently, to retrieve a certain record, only part of the data base need be searched. The access structure of data bases can be modeled as a tree as defined by Yao [7, 8], and this structure is referred to as the access tree. We illustrate the access tree concept by defining the access path for Honeywell's Indexed Sequential Processor, ISP [3]. Although the FDA currently evaluates ISP, only minor modifications are required to evaluate other indexed sequential implementations such as ISAM.

ISP permits access to records on a direct access device either sequentially or through the use of indexes. Initially, records are sequentially allocated within fixed-size pages across disk tracks and cylinders. The directory is composed of a coarse index and a fine index. The coarse index is sequentially searched until the appropriate fine index page is located. The average search length is approximately one-half the size of the coarse index. The fine index page contains the address of the page in which the required record resides. A search of the fine index is never more than one page. Thus, the search progresses from the coarse index, the fine index, and finally to the data page. In case of overflow in the data page, an overflow page must be accessed.

An access tree representation for ISP is shown in Fig. 2. Within each level the search is defined by average percentages of sequential accessing and accessing via pointers; between levels the search is always via pointers. Access paths and I/O processing time computations for other storage structures are described in Teorey and Das [6] and Yao [7].

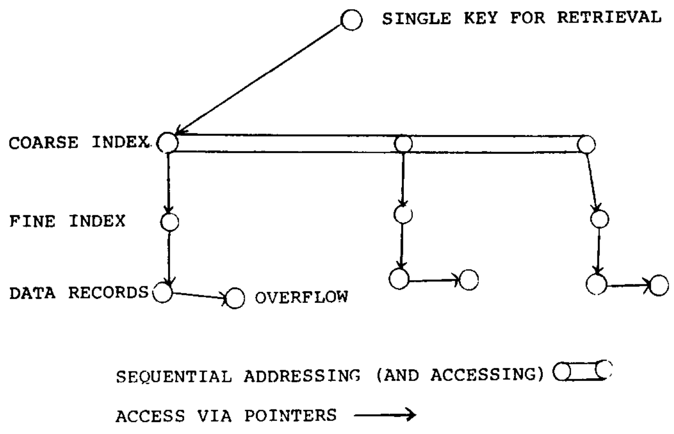

Fig. 2. The access tree representation of ISP.

ASSUMPTIONS

The model evaluates structures on the basis of a single record type. If record size is variable, several typical sizes can be chosen so the I/O processing time can be computed separately for each case. An example of this is given in the Case Study (d).

The storage medium is assumed to be a movable head device (disk), which includes fixed head devices as a special case with zero time. Individual transactions are assumed unordered and they request records which are uniformly distributed (and nonredundant) over the data base. The model assumes that contiguous storage is allocated to the data base. If the disk is dedicated to the process accessing the data base, the seeks are randomized within this contiguous storage. If the disk is shared, the seeks are assumed random over the entire disk surface.

Batched transactions are assumed to be ordered to maximize efficiency. The transactions, if unordered, would have randomly chosen records from the data base such that more than one transaction may occur for a given record. Thus the average number of data pages accessed to service all transactions is given by

$$\text{AVERAGE} = \text{NDP}*\left[1-\left(\frac{\text{NDP}-1}{\text{NDP}}\right)^{T}\right]$$

where NDP is the number of data pages (blocks) required for the data base, and $[(\text{NDP}-1)/\text{NDP}]^T$ is the probability that a given data page is not requested by any of T possible transactions in the batch, i.e., the result of random selection with replacement.

```
********HARDWARE PARAMETERS********

     19.  TRACKS/CYL.
404.0000  USABLE CYLINDERS PER DISK DRIVE OR PACK
 30.0000  M-SEC. AVG.SEEK TIME
 10.0000  M-SEC. NEXT CYL SEEK TIME
 16.7000  M-SEC.ROT.TIME
    1984  WORDS/TRACK
1074000.  CPS. TRANSFER RATE
      1.  WORDS/POINTER
  1.0000  BLOCK CONTROL WORD SIZE.  IN WORDS
  1.0000  RECORD CONTROL WORD SIZE.  IN WORDS
      2.  WORDS BLOCKING OVERHEAD
      6.  CHARACTERS PER WORD

********DATA BASE DESCRIPTION********

NUMBER OF RECORDS  =  24762
NUMBER OF RETRIEVAL ITEMS PER RECORD  =  1.0
NUMBER OF ACTIVE ITEM VALUES PER ITEM TYPE  =  24762.0
AVERAGE NUMBER OF RECORDS PER ITEM VALUE  =  1.0
LENGTH OF POINTER  =  1.0 WORDS.
LENGTH OF RECORD  =  8.0 WORDS.
WRITE  VERIFY CODE  =  1.0

********USER WORKLOAD********

FREQUENCY OF RECORD RETRIEVALS  =  463182   PER DAY
FREQUENCY OF QUERY RETRIEVALS  =   0.033    PER DAY
FREQUENCY OF RECORD INSERTIONS  =  0.500    PER DAY
FREQUENCY OF RECORD DELETIONS  =   0.500    PER DAY
FREQUENCY OF RECORD UPDATES  =     0.833    PER DAY
FREQUENCY OF ITEM UPDATES  =       0.0      PER DAY
FREQUENCY OF ITEM INSERTIONS  =    0.0      PER DAY
NUMBER OF QUERY CONDITIONS  =      1.000
NUMBER OF RECORD CONDITIONS PER QUERY CONDITION  =  1.0
NUMBER OF ITEM CONDITIONS PER RECORD CONDITION   =  24762.0
ESTIMATED SIZE OF QUERY RESPONSE SET = 24762.0
********STORAGE STRUCTURE PARAMETERS FOR ISP********

PERCENTAGE FILL  =  0.900
DATA PAGE SIZE  =  320.0 WORDS.
ITEM KEY SIZE  =  6.0 CHARACTERS.
INDEX PAGE SIZE  =  320.0 WORDS.
BATCHED TRANSACTION FLAG  =  0.0
AVERAGE RECORDS/BATCH  =  0.0
```

Fig. 3. FDA input parameters.

The FDA does not attempt to model all possible storage structure implementations. With the exceptions of Honeywell's ISP, the storage structure models were kept general so that most implementation-dependent parameters could be supplied via user-defined inputs (Fig. 3).

INTERFERENCE

The degree of interference of data base access can be categorized as follows:

1. Single access.
2. Single access with rotational delay.

3. Multiaccess.
4. Multiaccess with all system resource delays.

Single access is the most ideal case. It implies a dedicated disk, no delays due to multiprogramming, and CPU processing time subsumed within the time to traverse an interblock gap. Thus sequential processing of data under single access conditions results in virtually uninterrupted transmission of data. The second category assumes that access to the next contiguous block is delayed an average of half a rotation due to CPU processing and CPU wait.

The multiaccess case implies a shared disk in addition to multiprogramming delays so that access to blocks, which are contiguous for this process, becomes randomized by interference from other processes. The fourth category allows computation of elapsed time due to all resource delays and service time, whereas the first three categories describe only the I/O processing time (elapsed time on that disk).

The FDA implements single access and multiaccess as lower and upper bounds on I/O processing time for the given application. Operational experience with live test data indicates that single access with rotational delay is more typical of a lower bound, and the current model now includes this category. The fourth category was beyond the capability of an expected value model, but preliminary results with this category using queuing models indicate that the multiaccess interference model (category 3) is sufficient for comparative analysis of storage structures. Queuing or simulation models would be necessary to predict total elapsed time in a multiprogramming (and possibly multiprocessing) environment.

III. EXPERIMENTS WITH STORAGE STRUCTURES

The objectives of the experiments were to provide insight regarding sensitivity of performance to the values of physical parameters within a specified storage structure, to compare performance of alternative storage structures, and to provide test data for validation of the model. The results of the experiments were useful for determining the most important parameters in storage structure design.

During the calibration process several adjustments were required in the original model. The data base updating was done by setting up batches of ordered update transactions. The original version of the FDA assumed that the address of a record being processed was independent of the address of the subsequent record accessed. The modifications for batched processing resulted in the model I/O time being within 10% of live test data using accounting information from a stand alone test with this application. The model then accurately predicted the effect of block size changes on I/O processing time for sequential files.

CASE STUDY: AN INDEXED SEQUENTIAL DATA BASE ANALYSIS

a. Given Data Base Parameters

Data Base Size = 24,762 records.
Single record type.
Record size = 8 words (Honeywell 635).
Key Size = 6 characters (1 word).
Percent fill = 90%.
Page size = 320 words.
Access method: Honeywell ISP.
Workload: Two passes monthly for record updates. Random retrieval activity by 10 applications per day, ranging from 524 transactions per month for the least active and 92,619 per day for the most active, approximately linear between the two extremes.

b. Data Base Evaluation

One of the objectives of the FDA was to provide insight into the effect of different parameters on system performance. With this in mind, several experiments were run for this data base to test the effects of the most obvious physical design parameters. Figures 4 and 5 illustrate the sensitivity of I/O processing time and time × space product to index page size and data page size. The relative performance remained unchanged between single access and multiaccess.

Using I/O processing time as the performance criterion (Fig. 4), it would appear best to minimize all page sizes. However, when the time × space product is taken as the performance criterion, the current data page size of 320 words appears to be optimal. In both cases the index page size should be no larger than 64 words and possibly smaller. Because of the extremely slow rate of growth of the data base, the I/O processing time and time × space product were both minimized at 100% fill. After several months, however, the performance would start to degrade measurably, due to overflow, so that reorganization might be necessary.

The data page size of 64 words produces a larger time × space product than page size of 320 words because of the fragmentation of space in a 64 word page. The computation of the allowed number of logical records per page is as follows:

$$\text{Records per page} = \frac{\text{Page size} \times \%\text{Fill} - \text{BCW}}{\text{Effective record size}}$$

where Effective record size = stated record size + record control word size, and BCW = physical block control word size.

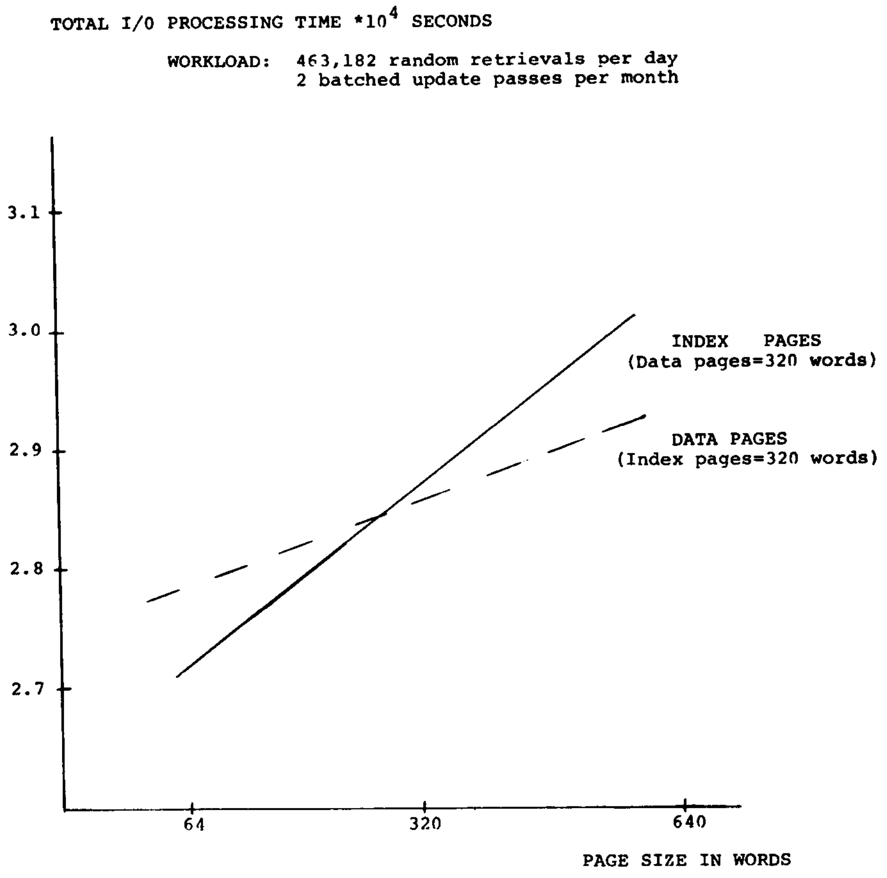

Fig. 4. ISP performance vs. block size (single access).

If block and record sizes was 1 word, percent fill was 90% and record size was 8 words; then a 64-word page would contain 6 records and a 320-word page would contain 31 records. Thus, the larger page size would reduce the total storage space required.

Alternative storage structures and access methods were tested for single access and for multiaccess with random interference. The output (Tables 1 and 2) indicated that direct access minimized I/O processing time for single access and ISP minimized I/O time for multiaccess. The direct method minimized the time $\times$ space product in both tests, indicating that it should be considered as an alternative to ISP for this application. Note that percent storage overhead for direct is less than ISP in this example, resulting in a lower time $\times$ space product.

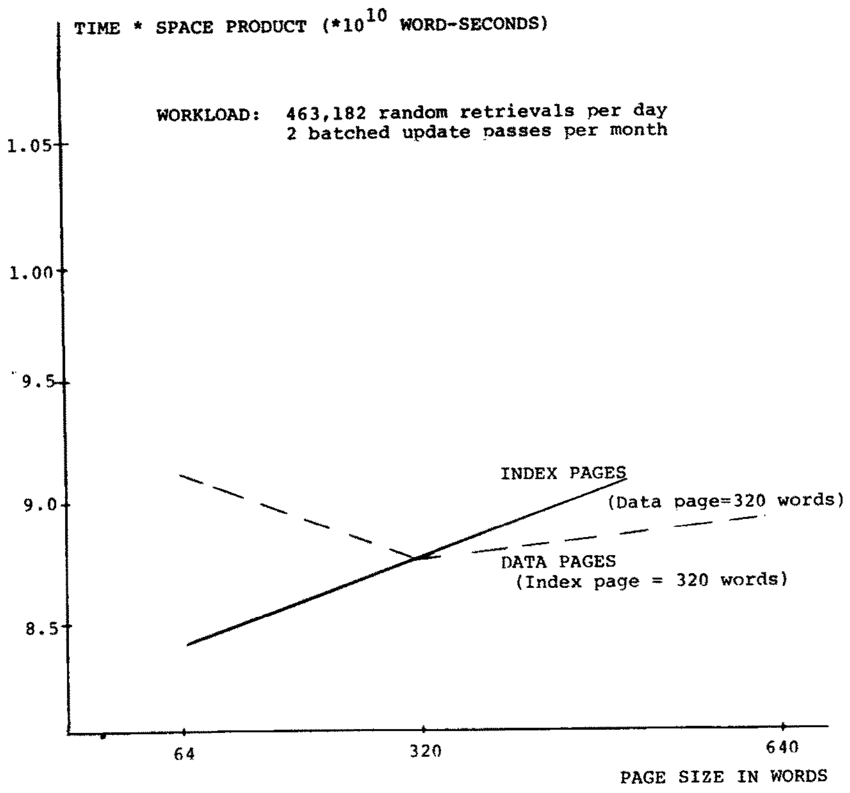

Fig. 5. ISP performance vs. block size (single access).

TABLE 1
Comparative Performance Statistics for Case Study (Single Access)

	Total I/O Time (sec) $\times 10^4$	Time Rank	T×S Product (Word-sec) $\times 10^{10}$	T×S Rank	% Storage Overhead
SEQ	32.6	5	7.4	5	12.6%
Direct	1.05	1	.3	1	25.7%
Inv.	8.49	4	2.7	4	38.6%
Multi.	8.486	3	2.7	3	38.6%
ISP	2.83	2	.9	2	35.6%

TABLE 2
Comparative Performance Statistics for Case Study (Multiaccess)

	Total I/O Time (sec) $\times 10^4$	Time Rank	T$\times$S Product (Word-sec) $\times 10^{10}$	T$\times$S Rank	Time Degradation over Single Access
Seq	659	5	149	5	20.2
Direct	6.2	2	1.6	1	5.9
Inv	147	3	47.6	3	17.3
Multi	147	3	47.6	3	17.3
ISP	5.6	1	1.7	2	2.0

c. *Direct Access as an Alternative to* ISP

The optimal design for direct access is superior to all other storage structures for this application. Table 3 summarizes the sensitivity of the direct access method to the number of buckets for hashing and the number of physical records specified per bucket. The performance (in I/O processing time) appears to be highly sensitive to these parameters. Note that records within a bucket are assumed unblocked but are stored in contiguous locations. Storage overhead variation is due to extra pointers for chained overflow, and is quite sensitive to the distribution of bucket size (as well as the distribution due to the hashing function, but this application is assumed to be uniformly distributed). The degradation in I/O processing time due to multiaccess interference increases as the data base becomes more sequentially stored (i.e., larger bucket size).

TABLE 3
Performance of Direct Access for Case Study

Inputs		Single Access			Multi-Access	
Number of Buckets	Records/ Bucket	I/O Processing Time (sec) $\times 10^4$	T$\times$S Product $\times 10^{10}$	% Storage Overhead	I/O Processing Time (sec) $\times 10^4$	Time Degradation over Single Access
24762	1	.70	.29	41.5%	1.45	2.1
12381	2	.92	.36	34.5%	2.55	2.8
8254	3	1.00	.39	30.9%	3.52	3.5
4127[a]	6	1.05	.40	25.7%	6.23	5.9

[a]Configuration for Tables 1 and 2.

d. Effect of Variable Record Size

The assumption of single record type (and fixed size) in storage structures is a serious limitation for many applications. However, the FDA does allow one to specify several record sizes, run them separately, and analyze the I/O processing time for each case. An experiment was attempted with a hypothetical application that involved every operation allowed (retrieval, insertion, etc.), and three sets of records, each with the same average size, were modeled assuming ISP. The results are presented in Table 4.

In each case the block size of page size was 320 words and the block control word and record control word (overhead) were one word each. In all three cases, there existed significant differences in I/O processing times, showing how significant the distribution of record size (and the discrete nature of fitting records into blocks) can be with respect to system performance.

TABLE 4
ISP Performance vs. Record Size Distributions

Case	Expected Record Size (Words)	Record Size Distribution	I/O Processing Time (sec)	% Increase over Case A
A	100	Constant	14240	—
B	100	$\frac{1}{3}$ 10 wds. $\frac{1}{3}$ 100 wds. $\frac{1}{3}$ 190 wds.	14640	3%
C	100	20% 20 wds. 10% 50 wds. 70% 130 wds.	20390	43%

IV. CONCLUSIONS AND RECOMMENDATIONS

The experiments described in this paper are condensed from a large collection of experiments which attempted to determine the most significant storage structure parameters.

No attempt was made to rank the parameters in order of significance because of the application-dependent nature of the experiments. Furthermore, the following list is not meant to be exhaustive, but should be considered as a minimal set of parameters to be tested for sensitivity in the physical data base design process.

	Parameter	Storage Structure Applicable
1.	Physical block size (data)	All
2.	Physical block size (index)	ISP, Multilist, Inverted
3.	Percent fill	ISP
4.	Bucket size	Direct Access
5.	Physical ordering	Sequential, ISP, Direct Access

In conclusion, the File Design Analyzer (FDA) program represents a significant step toward understanding physical data base design. It is an operational tool that is easily utilized and easily interpreted. Processing costs are low, enabling the experimental designer to test sensitivity of I/O processing time on many parameters at a single sitting.

On the other hand, the FDA should not be used without an awareness of its limitations. It is a first-order model of I/O processing time and does not include the waiting time in the queues for the system resources. It does not model the probabilistic nature of dynamic storage allocation in virtual storage systems, but allows the user to estimate page residence probabilities from statistical data collected on their own.

The FDA performance criteria, I/O processing time and secondary storage space, account for only part of total cost of managing data. Other costs include the main storage for buffers and data management software, CPU time for record processing and system overhead, personnel costs, development costs, and data base creation and reorganization costs. Because I/O processing time is usually a good measure of elapsed time to process a data base application, it also represents the time delays for channels, control units, main storage, and waiting time at time-sharing terminals. It would be straightforward then to estimate at least the hardware costs from rental cost figures. Data base creation and reorganization costs can be computed by formulating the individual data base operations required in terms of those already included in the FDA.

Plans for extending the FDA for hierarchical and general network structures are currently under way. It is believed that a generalized version of this model can eventually be linked to the logical data base design process, thus providing a more useful design aid over a wider range of alternative logical data structures.

The authors wish to acknowledge with deep appreciation the guidance and dedication of those who reviewed the computations and experiments: John De-Heus, Betty Finney, Jim Fry, Alan Merten, and Bob Tufts.

REFERENCES

1. A.F. Cardenas, Evaluation and selection of file organization—A model and system, *Comm. ACM* 16, 9 540–548 (1973).
2. A.F. Cardenas, Analysis and performance of inverted data base structures, *Comm. ACM* 18, 5 253–263 (1975).
3. Honeywell Information Systems, Inc. *Indexed Sequential Processor*, DA37, August 1973.
4. D. Lefkovitz, *File Structures for On-Line Systems*, Spartan Books, New Jersey, 1969.
5. M.E. Senko, V. Lum, and P. Owens, A file organization evaluation model (FOREM), *Proc. IFIP 1968*, C19–C23.
6. T.J. Teorey and K.S. Das, *Detailed Specifications for the File Design Analyzer*, SEL Tech. Report No. 87, Department of Electrical and Computer Engineering, The University of Michigan, July 1975.
7. S.B. Yao, *Evaluation and optimization of file organizations through analytic modeling*, Ph. D. Thesis, The University of Michigan, 1974.
8. S.B. Yao and A.G. Merten, Selection of file organization using an analytic model, *Proc. International Conference on Very Large Data Bases*, Framingham, Mass., Sept. 22–24, 1975, pp. 255–267.
9. S.B. Yao, K.S. Das, and T.J. Teorey, A dynamic reorganization algorithm, *ACM Trans. Database Systems* 1, 2 159–174 (1976).

Received January, 1977