1975

Modeling and Performance Evaluation of Physical Data Base Structures

S. B. Yao

Report Number:
76-179

Yao, S. B., "Modeling and Performance Evaluation of Physical Data Base Structures" (1975). *Department of Computer Science Technical Reports*. Paper 121.
<https://docs.lib.purdue.edu/cstech/121>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

MODELING AND PERFORMANCE EVALUATION OF
PHYSICAL DATA BASE STRUCTURES

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

CSD-TR 179

Modeling and Performance Evaluation of
Physical Data Base Structures

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

ABSTRACT

A generalized file organization model and performance evaluation system is developed for estimating the performance of physical data base structures. Performance evaluation results based on the cost functions of the model and system is presented. These performance measures are compared with the results of previous simulation models for indexed sequential, multi-list, and inverted file structures. The analytic approach makes the costs of evaluation very low. Consequently, many evaluations may be performed interactively with the file designer when searching for the most suitable structure for a given application.

CR Categories: 3.72, 4.33

Modeling and Performance Evaluation of
Physical Data Base Structures

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

ABSTRACT

A generalized file organization model and performance evaluation system is developed for estimating the performance of physical data base structures. Performance evaluation results based on the cost functions of the model and system is presented. These performance measures are compared with the results of previous simulation models for indexed sequential, multi-list, and inverted file structures. The analytic approach makes the costs of evaluation very low. Consequently, many evaluations may be performed interactively with the file designer when searching for the most suitable structure for a given application.

CR Categories: 3.72, 4.33

---

1. Introduction

The design of file organization has an important effect on the performance and associated costs of a data base management system. Various techniques differ widely in their performance. Depending on the type of process involved: creation, insertion, deletion, random retrieval of records, query retrieval, or sequential retrieval of a large set of records, some techniques may out-perform others in particular situations. The problem is compounded by the large number of available file organization techniques and their variations. Designer of data base and file organizations are faced with a difficult problem when trying to determine or predict the performance of particular structures.

There are modeling attempts which try to gain understanding to the function and performance of file organizations. These modeling approaches fall into two major categories: simulation models and analytic models. One of the earliest efforts on simulation modeling is the FOREM model [6]. Detailed modeling and simulation results on ISAM were reported [5]. An important simulation model on multiple-attribute file organizations was later reported by Cardenas [2]. Simulation results on multi-list, inverted file, and doubly-chained tree organizations were presented. Simulation models that compare indexed sequential and direct

---

access methods were presented in [1]. The common approach of these simulation models is that they are all based on individual file structures models. Since different models are used for different file organizations, they do not provide a common ground for performance comparison and file organization selection.

A formal model for some popular file organizations was introduced by Hsiao and Harary [3]. This analytic model divides a file organization into two parts: a directory and a record file. Each entry in the directory corresponds to a keyword and contains head pointers to lists of records in the record file. The file organizations describable by this model are characterized by the number of lists associated with a keyword. Multi-list and inverted file can be accurately derived from the model. This file model, however, is unable to describe record sequence which is not list oriented, such as sequential or indexed sequential files. To extend the range of modeling, a two-parameter file structure model was introduced by Severance [7]. By indicating sequential/linking list connections and direct/indirection of record storage, the model was able to describe the general storage characteristics of the record file. The model is still too simplistic in description, and its scope of representation only include two additional file structures: sequential and indexed sequential organizations.

A more generalized file organization model, the Hierarchical Access Model, was introduced by Yao [9,10]. This model makes clear the distinctions between the access paths in a file and their storage representation. Using tree structure to model the access paths and using parameters similar to that of Severance to model storage representation, the Hierarchical Access Model can adequately characterize and distinguish single-attribute file organizations such as sequential, indexed sequential, B-tree and hashing, and multiple-attribute file organizations such as multi-list, bounded multi-list and inverted file. A generalized retrieval algorithm and a set of cost functions based on the Hierarchical Access Model is also developed [10].

In this paper, some performance evaluation results based on the cost functions of the Hierarchical Access Model are presented. These performance measures are further compared with the results of previous simulation models. The evaluation system is implemented in FORTRAN and is very efficient. The execution time for each evaluation is less than 0.5 second on IBM360/67. This system is also modified and adapted to run on the Honeywell 635 and IBM370/168. Additional test cases will be reported elsewhere.

---

2. The Hierarchical Access Model

Consider a set R of attributes and its associated set V of values. A record is a subset of the Cartesian product $R \times V$ in which each attribute has one and only value. The keywords are the ordered pairs which are used to characterize or identify records. A K-set $S(k)$ with respect to a keyword k is a set of records containing the keyword k . Any K-set can be partitioned into w disjoint groups $G_1(k)$, $G_2(k)$, ..., $G_w(k)$. A set of records is a file if every K-set is either contained in F or disjointed from F .

Access to a data base is usually made through a series of index searches which may involve sequential 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 [9,10], and this structure is referred to as the access tree.

Figure 1 illustrates the access model. The access paths are logically partitioned into two portions. Levels 0 - r of the access tree represent the directory, while the n -th level represents the file.

The access structures in the directory can be further divided into three stages. In the first stage (levels 0 - s) the attribute name is searched. Each node on the s -th

Figure 1. The access tree of a file

level represent an attribute. Since the number of attributes in a file is not very large, a one-level search structure is usually adequate (i.e., $s=1$). The second stage (levels $s - t$) searches the keywords for a given attribute. Depends on the particular application, many level indices may be necessary. Figure 2 shows a three-level index for keywords. For each node on the t -th level, stage three searches for the groups associated with a keyword. Therefore the output of the directory search is a set of keyword groups.

In order to facilitate searching, secondary data are stored in the nodes of the access structure. Each node may contain access path pointers and a high key (or partial key). We note that the high key is defined as the highest key stored in the subtree of a node [4], while the partial key is the key component to be decoded on a level [8]. The attribute names are stored on levels $0 - s$ and the keywords are stored on levels $s - t$ of the access tree. The range of the record addresses correspond to a group is stored in a node on the group level r .

The K-sets in the file are organized into groups. From each group node in the directory, there is an access path which passes through a group of records in the file. We note that the K-sets (and hence the groups) of an attribute is usually disjoint while a record may belong to several

---

Figure 2. A two-level index structure for keywords

groups of different attributes.

A generalized retrieval algorithm is developed with the model. Consider the retrieval of records to satisfy a query. A query is defined as a proposition given by a Boolean expression of keywords in the file. A general normal form of query is

$$(K_1 \wedge K_2 \wedge \dots \wedge K_r) \vee (\dots) \vee \dots \vee (\dots).$$

The response set is the set of records satisfying a given query (records for which the query proposition is true).

Consider a conjunct of a given query

$$(K_1 \wedge K_2 \wedge \dots \wedge K_r).$$

The response set for the conjunct is given by

$$R = \bigcap_{i=1}^r S(K_i) = \bigcap_{i=1}^r \bigcup_{j=1}^{w_i} G_j(K_i)$$

The search to satisfy a conjunct can be accomplished in the following phases:

1. Search the directory and obtain a set of group nodes for each keyword in the conjunct.
2. (pre-search) Perform intersection on groups of different keywords. A group is essential if its intersection with another group is non-empty. This can be determined using the range indicator in each group node. The size of the essential group is defined as the number of records in the group.
3. Find the keyword which has the minimal total size of its essential groups.

4. Search all the records in the minimal essential groups. Examine the records retrieved and only keep those contain all the keywords in the conjunct.

An example is given in Figure 3 to illustrate the algorithm.

It was noted by Hsiao and Harary that the searching for different conjuncts of a query can be made parallel [3]. When the above algorithm is performed parallelly for all the conjuncts, redundant retrieval of records in the file can be avoided. To verify the conjunct retrieval algorithm, let us consider two extreme cases. In a multi-list file structure, there is only one group for each keyword. The records in the group are organized into a linked list. Therefore phase 2 of the algorithm is eliminated (i.e., no time is required for this phase). Phase 3 picks the shortest list to retrieve. In an inverted file, the number of groups is equal to the number of records in a K-set. Therefore the size of each group is one. After the intersection is performed, phase 4 is unnecessary since all keywords have the same essential groups. It is clear that structures between the two extremes can be searched by this algorithm.

The cost of accessing records in the file can be computed by cost functions of the algorithm. Two important costs are involved: sequential access time and random access time. These are computed based on the implementation parameters of the model and is beyond the scope of this

---

To produce $(K_1\text{-set}) \cap (K_2\text{-set})$:

- (1) Perform intersection on the group level
- (2) The one-level subtree defined by the intersection and rooted in (30-40) is searched to produce the record 30 and 35.

Figure 3. Query retrieval search algorithm

paper. Detailed cost functions can be found in [9].

4. Performance of Multiple-Attribute File Structures

In this section, some results of the performance evaluation using the access model are demonstrated and verified. Several typical file structures are evaluated. These results are compared with those obtained by existing file models.

Generalized data base systems are usually organized with multiple attributes. In addition to the alternatives for structuring the directory, a spectrum of file structures is available for organizing the records. Previous models have all been designed to evaluate particular cases in the spectrum. For example, Cardenas's simulation model evaluates the two extreme cases of multi-list and inverted file structures. The application to six real data bases are described using the time series statistics obtained [2]. The access model has been used to compute the file structure performance using these same data statistics. The results are very similar to those of Cardenas at the extreme limiting cases.

The input parameters corresponding to those used by Cardenas are summarized in Table 1. The user related parameters and the storage related parameters are the same

---

(1) User related parameters:

Fr	frequency of record retrievals	1
Fq	frequency of query retrievals	1
Fi	frequency of record insertions	1
Fd	frequency of record deletions	1
Fud	frequency of record data updates	0.5
Fuk	frequency of record keyword updates	0.5
Fki	frequency of new keywords	1
Qc	number of conjunctions per query	8
Qt	number of attributes per query	4
Qk	number of keywords per query	5
Qr	estimated size of query response set	200

(2) Storage related parameters:

Ta	average random access time	100
Ts	average sequential access time	10
Tf	average transfer time per byte (ms)	0.0032
Tp	average processing time per byte (ms)	0.0046
Sp	length of a pointer (byte)	10
Sv	blocking overhead (byte)	140
Bt	1st level block size (byte)	10,740
Bs	second level block size (byte)	10,740
Ct	cost of time per ms (cent)	1
Cs	cost of storage per byte-month (cent)	1

Table 1. The user and storage related parameters of the six test cases

for all test cases while the data bases are different. The results of the access model performance evaluation are plotted in Figure 4 through Figure 9. The number of groups w per K -set controls the type of the file structure. As w varies from 1 to the size of the K -set, the file structure varies from multi-list to inverted file. The results of Cardenas correspond to the beginning and the end points of the query retrieval curves. If the selection of file structures is based on query retrieval cost, these figures suggest that multi-list be used for data bases 1, 2, 3 and 6 while an inverted file should be used for data bases 4 and 5, which is the same conclusions as Cardenas. The access model not only gives cost estimates for the extreme cases but also for the entire range in between, and it further considers record retrieval, update, and storage costs.

An examination of these curves reveals that the performance of multi-list file depends heavily on the size of the K -sets. The performance of inverted file does not vary much for different data bases since most of its retrieval time is spent on the directory search and hence depends more on the query complexity. For large K -sets, the inverted file gives better performance. Further, for small K -sets, multi-list can be better than the inverted list, depending upon the other parameter values.

Data related parameters:

Nr	number of actual records in file	3676
Nt	number of retrieval attributes in file	6
Na	number of retrieval attributes per record	6
Np	number of possible keywords per attribute	2914
Nu	number of active keywords per attribute	2914
Rk	average number of records in a K-set	8
Sn	length of node label in file (byte)	10
Sr	length of record (byte)	87

Figure 4. Costs of data base 1

Data related parameters:

Nr	number of actual records in file	3676
Nt	number of retrieval attributes in file	4
Na	number of retrieval attributes per record	4
Np	number of possible keywords per attribute	496
Nu	number of active keywords per attribute	496
Rk	average number of records in a K-set	30
Sn	length of node label in file (byte)	3
Sr	length of record (byte)	87

Figure 5. Costs of data base 2

Data related parameters:

Nr	number of actual records in file	5239
Nt	number of retrieval attributes in file	4
Na	number of retrieval attributes per record	4
Np	number of possible keywords per attribute	483
Nu	number of active keywords per attribute	483
Rk	average number of records in a K-set	43
Sn	length of node label in file (byte)	4
Sr	length of record (byte)	93

Figure 6. Costs of data base 3

Data related parameters:

Nr	number of actual records in file	18573
Nt	number of retrieval attributes in file	4
Na	number of retrieval attributes per record	4
Np	number of possible keywords per attribute	1175
Nu	number of active keywords per attribute	1175
Rk	average number of records in a K-set	63
Sn	length of node label in file (byte)	3
Sr	length of record (byte)	84

Figure 7. Costs of data base 4

Data related parameters:

Nr	number of actual records in file	15888
Nt	number of retrieval attributes in file	6
Na	number of retrieval attributes per record	6
Np	number of possible keywords per attribute	271
Nu	number of active keywords per attribute	271
Rk	average number of records in a K-set	352
Sn	length of node label in file (byte)	4
Sr	length of record (byte)	236

Figure 8. Costs of data base 5.

Data related parameters:

Nr	number of actual records in file	1296
Nt	number of retrieval attributes in file	4
Na	number of retrieval attributes per record	4
Np	number of possible keywords per attribute	466
Nu	number of active keywords per attribute	466
Rk	average number of records in a K-set	11
Sn	length of node label in file (byte)	4
Sr	length of record (byte)	404

Figure 9. Costs of data base 6

It should be pointed out that these six real data bases are relatively small in size and do not provide a sufficient basis to establish a design guideline. They are used here only for comparisons. Arbitrary large data bases can be evaluated once their design parameters are input into the system.

5. Performance of Single-Attribute File Structures

The distinction between single-attribute and multiple-attribute file structures were never made clear in previous models. The fact that single-attribute files are special cases of multiple-attribute files is clearly demonstrated by the access model. Since there is only one attribute, the access tree structure above the attribute level is unnecessary. On the keyword level, all of the keywords have the same attribute. Since these keywords act as identifiers, the structures between the keyword level and the record level are degenerated.

To demonstrate the evaluation of a single-attribute file structure, the indexed sequential file will be used. This structure has been commercially programmed and implemented in a variety of ways. The Indexed Sequential Access Method (ISAM) of IBM is chosen for demonstration. ISAM permits access to records of a disk file in either a

---

sequential or direct manner through the use of indices. Initially, records are sequentially allocated on various tracks of cylinders. The cylinder index consists of a highest key for each cylinder. There is one track index for each cylinder. Each entry in the track index contains a highest key and a highest key in its overflow area.

The input parameters for the evaluation are shown in Table 2. The storage parameters are based on the IBM 2314 disk storage device. Define

W_1 = number of entries in a cylinder index

W_2 = number of entries in a track index

W_3 = number of records in a prime track and
its overflow

Some constraints on the design parameters W_1 , W_2 , and W_3 are implied by the ISAM structure. Given a record level blocking factor $Bf=1$, the number of records in a prime track is

$$U = [Bm / (Sv + Bf * Sr)] = [7294 / (128 + 600)] = 10$$

where $[x]$ denotes rounding off and Bm denotes the maximal allowable block length. Let P be the percentage of overflow and $U = (1-P) * W_3$, we have

$$W_3 = U / (1-P) = 10 / (1-P)$$

Since a track can contain U records, the number of tracks required for a prime track and its overflow is $1/(1-P)$. There are 20 tracks per cylinder. The number of prime tracks in a cylinder (number of entries in a track index) is

(1) Data related parameters:

Nr	number of actual records in file	50,000
Nt	number of retrieval attributes in file	1
Na	number of retrieval attributes per record	1
Np	number of possible keywords per attribute	50,000
Nu	number of active keywords per attribute	50,000
Rk	average number of records in a K-set	1
Sn	length of node label in file (byte)	10
Sr	length of record (byte)	600

(2) User related parameters:

Fr	frequency of record retrievals	1
Fq	frequency of query retrievals	0
Fi	frequency of record insertions	1
Fd	frequency of record deletions	0
Fud	frequency of record data updates	0
Fuk	frequency of record keyword updates	0
Fki	frequency of new keywords	1
Qc	number of conjunctions per query	0
Qt	number of attributes per query	0
Qk	number of keywords per query	0
Qr	estimated size of query response set	0

(3) Storage related parameters:

Ta	average random access time	90
Ts	average sequential access time	12
Tf	average transfer time per byte (ms)	0.0032
Tp	average processing time per byte (ms)	0
Sp	length of a pointer (byte)	10
Sv	blocking overhead (byte)	128
Bt	1st level block size (byte)	7294
Bs	second level block size (byte)	7294
Ct	cost of time per ms (cent)	1
Cs	cost of storage per byte-month (cent)	1

Table 2. Input parameters for the
evaluation of an ISAM file

$$W2 = 20*(1-P)$$

The total number of records in a cylinder is

$$W2*W3 = 200$$

If there are Nr records in the file, the total number of cylinders (number of entries in cylinder index) required is

$$W1 = Nr/200$$

The retrieval and insertion times per record are plotted against the percentage of overflow in Figure 10. The solid curves are for full track blocked files and the dashed curves are for unblocked files (i.e., $Bf=1$). The cost for files with full track blocking is much lower than without blocking, but the price for this is a much larger buffer for deblocking. In both cases, the retrieval time increases with overflow and the insertion time decreases with overflow. Optimal total performance may be reached when there is a certain amount of overflow in the file. Assuming the same frequencies for all types of transaction, the total cost is plotted in Figure 11. The optimal overflow in this case is about 30%. This coincides with the finding of Lum, et al. [5] who concluded by running the FOREM simulation model that the performance of ISAM with cylinder overflow and equal retrieval and insertion frequencies is optimum with 25-45 percent of overflow.

it is generally believed that a master index (an index to the cylinder indices) is useful for a large file but not

Figure 10. Influence of overflow on the retrieval and insertion times

for a small file. IBM suggested that a master index be used when the cylinder index occupies more than four tracks [4]. Lum, et al. also made the same suggestion for full track blocked files with "reasonable key sizes". When the file is not blocked, the suggested decision point is two tracks.

Using the access model cost functions for full track blocking and no overflow records, the retrieval and insertion times are plotted against varying file sizes in Figure 12. The times with master index are plotted in dashed curves. The flatness of the dashed curve is clearly due to the fact that increased file size only affects the size of the master index, which is relatively small. The two sets of curves intersect at 300,000 and 400,000. Since there are 200 records for each cylinder, these require 1500 and 2000 cylinders. These correspond to 4.1 and 5.5 tracks of cylinder indices since each track of cylinder index contains

$$\begin{aligned} & [(Bm - Sv) / (Key\ length + Sp)] \\ & = [(7294 - 146) / (10 + 10)] = 357\ entries \end{aligned}$$

The decision point of using master index lies somewhere between these intersections depending on the relative frequencies of retrieval and insertion. The same set of curves are plotted for unblocked files in Figure 13. The decision point is about 14,000 records which corresponds to 70 cylinders and to 1.6 tracks of cylinder indices since each track can contain $[(Bm / (Sv + Key\ length + Sp))] = [7294 /$

Figure 11. Total access time for an ISAM file

Figure 12. Use of master index for
a full track blocked ISAM file

Figure 13. Use of master index for an unblocked ISAM file

$(146+10+10)] = 43$ entries.

6. Conclusions

The modeling concepts and the file organization evaluation system represent a significant step towards understanding the design of physical data bases. The analytic approach makes the costs of performance evaluation very low. Consequently, many evaluations may be performed interactively with the file designer when searching for the most suitable structure. The results of this efficient tool is validated by comparing with the previous expensive and complex simulation models.

It should be noted that this evaluation system computes only first order average values for processing time and storage requirement. The system is currently being extended to include more detailed probabilistic measures and other considerations such as multi-programming and channel interferences. The model is also being extended to represent logical access paths in hierarchical and network structures.

REFERENCES

1. Behymer, J.A., Ogilvie, R.A. and Merten, R.G. Analysis of indexed sequential and direct access file organizations. Proc. SIGMOD 1974 Workshop, (May 1974) 389-417.
2. Cardenas, R.F. Evaluation and selection of file organization -- A model and system, Comm ACM, 16,9 (Sept. 1973) 540-548.
3. Hsiao, D. and Harary, F. A formal system for information retrieval from files, Comm ACM 13,2 (Feb. 1970) 67-73; "Correction" Ibid., 13,4 (April 1970) 266.
4. IBM Corp. Introduction to IBM system 360 direct access storage devices and organization methods (C20-1649). White Plains, N. Y.: IBM Corp., 70 pp.
5. Lum, V.Y., Ling, H and Senko, M.E. Analysis of a complex data management access method by simulation modeling. Proc. 1970 Fall Joint Computer Conference, 211-222.
6. Senko, M.E., Lum, V. and Owens, P. A file organization evaluation model (FOREM). Proc. International Federation of Information Processing Societies, 1968, 514519.
7. Severance, D.G. A parametric model of alternative file structures. Information Systems, 1,2 (1975) 51-55.
8. Sussenguth, E.H. Use of tree structures for processing files. Comm ACM 6,5 (May 1963) 272-279.
9. Yao, S.B. evaluation and optimization of file organizations through analytic modeling. Ph.D. Dissertation, The University of Michigan, 1974.
10. Yao, S.B. A hierarchical access model for data base organizations. Technical Report TR-177, Computer Sciences, Purdue University, Lafayette, Indiana 47907. (Feb. 1976).