A Dynamic File Organization Model

R. H. Davis and P. Coumpas
Department of Computer Science, Heriot-Watt University, Edinburgh, UK

A dynamic analytical file organization model is presented which enables the file designer to estimate file performance and cost against time and offers a quantitative solution to file organization problems. Costs, measured in terms of processing time, reorganization and storage costs, are obtained from the input data characteristics, user requirements and hardware specifications. Highly elusive costs such as those for software maintenance or system storage are not included. Six common file organizations are featured and the usefulness of the model as an operational tool is exhibited by means of a representative series of demonstrations with different file sizes and types.

1. FILE ORGANIZATIONS AND MODELS

Common levels of abstraction for the organization of data are those of logical data organization[1] and physical storage organization.[2] The former is viewed as the user's conception of data, independent of the manner in which the data are physically stored and represents the inherent structures and relationships among data items over which the file designer has no control. Physical storage organization on the other hand refers to the internal representation of data as they are stored on a device.

Models of files based on these levels of organization have been developed. Bachman introduced data structure diagrams, to illustrate logical file organizations, a model used extensively by the users of the IDS data management system. Reflecting physical organization, file structure models enable an efficient analysis of complex file structure operations to be made and they also make file design evaluation easier. Both simulation and analytical models have been used for file structure models. A file simulation model, called FOREM was introduced by Senko *et al.*[3] and extensive simulation results for the indexed-sequential access method using FOREM were reported by Lum *et al.*[4] More recently a file organization simulation model and system written in COBOL was introduced by Cardenas[5] who, without considering updating activities, ran simulations of inverted, multi-list and doubly-chained tree files on several data collections. Severence and Merten[6] used a simulation model to construct a generalized file organization from parametric modular components, good structures being selected on the basis of a file structure's evaluation in terms of space required, retrieval time and reorganization time. Finally, Siler[7] implemented a simulation model of data retrieval systems to analyse file inversion. Instead of using original database records as Cardenas or Senko had done, an abstracted form of the database derived from important database statistics is used to compute measures of system performance. This stochastic methodology is then applied to inverted, threaded and cellular list organizations.

Analytical file structure models, capable of representing several file organizations, were first introduced by Lefkovitz.[1] A list oriented structure is employed in which records with the same keyword are connected by a list or a set of lists. The file structure is divided into two parts: the list structure file contains multiple lists and the key directory contains the heads of the lists. The type of file organization represented varies from the multi-list file, to cellular partition file and inverted list file. This form of model was employed by Martin[8] to construct a model for file organization selection. A set of equations was developed for each of Lefkovitz's three configurations so that a file organization could be selected by comparing the computed costs.

Cardenas[9] in a later paper derived equations to estimate access time and storage specifically for inverted file organizations. This model is proposed to be used as a module in an established inverted file system. It could be invoked periodically to estimate access time and storage requirements for alternative inversions as the user demands and the file system contents change.

Waters[10] derived timing formulae to evaluate each processing mode of a self-indexing file and compared different processing modes to present a threshold between the serial and random updating of a predefined self-indexing file. Finally Yao and Merten[11] used a model implemented with analytical optimizing techniques to select file organizations. The output of the design system is a class of file organizations specified by their average characteristics and details of the actual file organization can be determined by simulation or other techniques.

The development of the dynamic analytical model presented here is inspired by the systematic analytical methodology of Weiderhold,[12] Schneiderman[13] and Yao *et al.*[14] It is among the few existing models which isolate and study a wide range of physical organizations. In addition the model analyses and computes the evolution of performance and cost with time, an approach not found in models previously studied.

2. METHODOLOGY FOR THE DYNAMIC ANALYTICAL MODEL

The objective is to be able to provide an integrated result in the form of total costs involved in setting up and running a particular file organization over a given period of interest. To do this, given particular file characteristics, user activity and device specifications, two main areas need consideration. First an examination of the times required to perform various operations on the range of file organizations needs to be made. Secondly the development of the costs incurred during the file life

CCC–0010–4620/84/0027–0143 $04.00

needs to be established by considering the increased user activity in parallel with file size extension and its influence on reorganization intervals and rate of increase in search costs.

The analysis is made in as general and as uniform a way as possible in order to investigate the implementation of a number of file organizations within a common frame.

2.1 Performance analysis

Several authors[1,5,9,11,12,15] have suggested parameters and formulae to estimate the timing of operations that influence the performance of file structures. Weiderhold's methodology subject to some necessary modifications and additions[16] was considered the most systematic and was adopted for the performance analysis. The structure of a file is reflected in the time required to undertake the nine operations listed below. The existence of record order according to a primary key or based on indices or upon a sequence determined by pointers, greatly influences these times, facilitating some of the operations and hindering others. Six file organizations were studied as follows: sequential, pile, index sequential, direct, inverted and multiring. For each of these, nine measures of response time were noted, namely the times required

(i)	to fetch a record from anywhere in the file	T_f
(ii)	to fetch a record with search argument a no-key attribute	T_{f0}
(iii)	to fetch the next record in the file	T_n
(iv)	to insert a record	T_i
(v)	to update a record	T_u
(vi)	to delete a record	T_d
(vii)	to read an entire file sequentially	T_x
(viii)	to read the entire file serially	T_{xs}
(ix)	to reorganize the entire file	T_y

The Appendix includes outline expressions of performance for the six organizations considered. Development of these expressions may be found elsewhere.[16]

2.2 Cost analysis

Although it is difficult to achieve the most appropriate file organization for a given situation, useful quantifiable measures for judging a file organization are the total processing and total storage costs. The initial search cost per day C'_0 is a collective term for both the initial processing and initial storage costs. In a dynamic file the search cost per day will increase as additions and deletions are made, so that the need for a reorganization of the file to reduce search costs increases with time. However since the cost of performing a reorganization is not negligible, the frequency of reorganization should be determined carefully. The dynamic reorganization criterion of Yao, Das and Teorey[14] was selected in this model to estimate reorganization intervals since reorganization intervals can be determined independently of the file life time, a feature not found in other estimations.[13]

Figure 1 depicts the way search costs may be expected to increase with increasing user activity. The total search cost without reorganization is represented by the curve (ABCDE). The approximation (AFBGCHDI) assumes linear growth at the end of time intervals $t_{i1}, t_{i2}, \ldots t_{in}$.

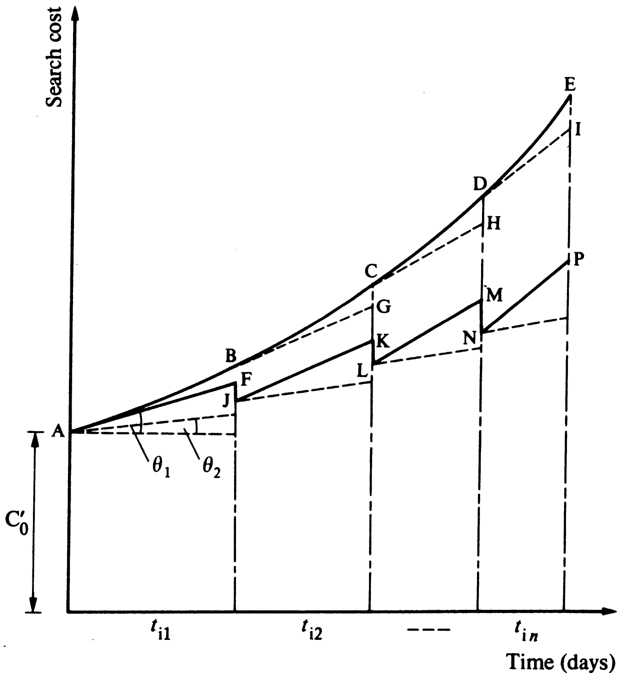

Figure 1. Cost function with increasing user activity.

This growth is dependent initially on search cost C'_0 and the increase factor θ_1 which is assumed to depend on file size. If reorganization takes place at the end of each time interval then costs fall as shown by (AFJKLMNP). The initial search cost increase factor assuming regular reorganization is now θ_2 and the initial reorganization cost C'_{r0}. For growing files, this reorganization cost will increase by a factor m_i after each time period.

According to the dynamic reorganization criterion the optimal time period, t_i, between reorganizations should be such that the gain due to improved search costs at least equals the expense of reorganizations. It can be shown[14] that

$$t_i = \{m_i + \sqrt{[m_i^2 + 4C'_{r0}(\theta_1 - \theta_2)]}\}/\{2(\theta_1 - \theta_2)\} \quad (1)$$

In general C'_0 is the sum of the products of a number of different operations (Op_i), $i = 1, \ldots n$ on the file and the corresponding initial time, $T_i(0)$, required to perform that operation multiplied the processing cost, C_{pr}, plus the initial storage cost, C'_{S0}, as follows:

$$C'_0 = \prod_{i=1}^{n} [Op_i\, T_i(0)]\, C_{pr} + C'_{S0}$$

The initial storage cost is the product of the space occupied and the cost of storage per byte and day, C_{bd}.

The search cost after a time period t_i is given by

$$C_{ti} = C'_0 + \theta_1 t_i$$

and after reorganization by

$$C'_{ti} = C'_0 + \theta_2 t_i$$

whereas the storage cost at that point in time has become

$$C_{sti} = C'_{S0} + \theta_S t_i$$

where θ_S is the storage cost increase factor, and the cost of reorganization at the end of the same period is

$$C_{rti} = C'_{r0} + m_i t_i$$

Using equation (1) to determine t_i, the total cost during that period is

$$C_{\text{titotal}} = C'_0\, t_i + \tfrac{1}{2}\theta_1\, t_i^2 + C_{\text{tiprevious}}$$

$C_{\text{tiprevious}}$ is the reorganization cost at the end of the previous period. Thus the final cost during the file evaluation period T is the sum of the successive partial total costs at time periods t_i.

For comparison, the final total cost during the file evaluation period T without reorganization is required. This is again the sum of the successive partial total costs but, in this case, the total cost during a period without reorganization C_{tiwtotal} is given by

$$C_{\text{tiwtotal}} = C'_0 t_i + \tfrac{1}{2}\theta_1\, t_i^2$$

In addition to providing the performance parameters and costs at both the beginning and at the end of the file evaluation period, the model is designed to allow the user to choose a specific interval at which the above computations are made. Figure 2 shows a typical situation

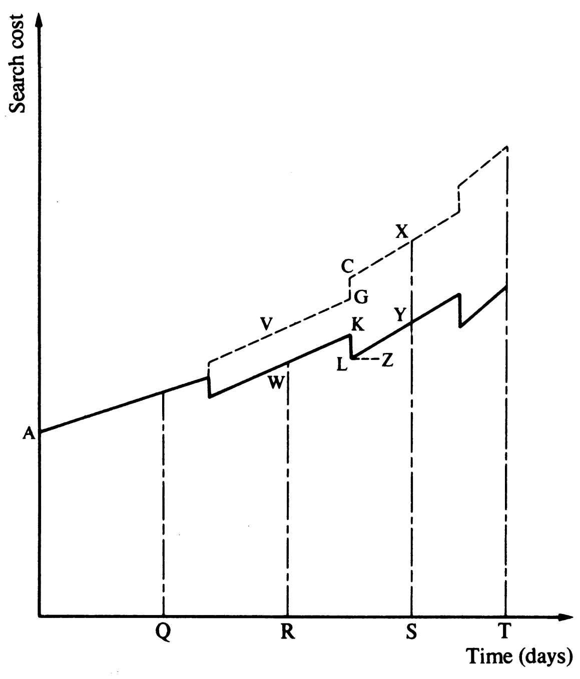

Figure 2. Cost function at regular intervals.

in which it will be noted the last time period will in general be shorter than the other regular intervals.

During the interval RS for instance, initial search costs with and without reorganization are WR and VR, respectively, the final search cost increase factor θ_1 is the angle ZLY and the number of reorganizations which took place is one at point K. The total search cost without reorganization is thus represented by area (RVGCXSR) and that after successive reorganization (RWKLYSR).

Thus having initial search and reorganization costs as well as cost increase factors, the reorganization intervals and total cuts can be calculated for any length of time.

2.3 Model description

The model has been programmed so that it may be used interactively using Pascal.[16] Owing to the extensive amount of input data that is required care has been taken to avoid asking the user unnecessary questions or requiring him to provide lengthy answers. Once the input data have been presented the performances and costs of a selection of file organizations are evaluated, the results being output in table form to assist the user in making the most of the information. The tables can be stored for subsequent print out or the input data displayed again or printed on request.

The input data are separated into three categories: file characteristics, user requirements and hardware characteristics, a specimen input data table is shown in Table 1.

Table 1. Input data

File characteristics	
No. of records	= 50,000
No. of attributes in file	= 10
Average no. of attributes in a record	= 5
Attribute value average length (bytes)	= 12
Attribute name average length (bytes)	= 12
Percentage fill of file (0.2–1)	= 0.95
User requirements	
No. of record retrievals (per day)	= 500
No. of record insertions (per day)	= 100
No. of record modifications (per day)	= 50
No. of record-key modifications (per day)	= 5
No. of attributes in record to be modified	= 2
No. of search arguments	= 10
No. of record deletions (per day)	= 40
No. of successor record retrievals (per day)	= 1000
No. of record retrievals based on no-key attribute (per month)	= 1
No. of file readings (per month)	= 2
No. of file serial readings (per month)	= 1
File organization evaluation time (months)	= 24
Length of output report period (months)	= 2
User activity increase period (days)	= 120
Hardware characteristics	
Average rotational latency (ms)	= 8.33
Average seek time (ms)	= 30
Transfer rate (bytes/ms)	= 806
Block size (bytes)	= 512
Pointer length (bytes)	= 8
No. of blocks per track	= 26
No. of tracks per cylinder	= 19
Processing cost (pence/min)	= 20
Storage cost (pence/Mbyte-day)	= 2

Under user requirements, all the data relative to the type and use of the file have to be provided. Note that modifications are divided into key and no-key modifications depending on whether the file organization has a primary key (such as in the case of the sequential, indexed-sequential and direct type of organization) or implements pointers instead (such as the inverted and multiring types). The user activity increase period determines the time points at which the user activity increases and is so arranged to provide a model with the flexibility to enable the user to adjust this user activity increase period for each of the individual file organizations he wishes to consider. The user activity increase period has to be approximately that of the longest reorganization period in any modelled organization so that there is a uniform influence of user activity increase on the total cost of different organizations.

3. FILE EVALUATION

All the six modelled organizations were evaluated for the input shown in Table 1. The hardware characteristics are those of an IBM 3330 disk storage unit. The results of the sequential organization are presented in Tables 2 and 3. These results represent the output format as it might be displayed on a VDU and is the format through which the performance and evaluation of any of the six organizations can be scrutinized.

In Table 2 the nine measures of response time are provided at each of the discrete points in time noted in the first column. In Table 3 the search cost per day is shown to increase slightly irregularly due to dependence of the report time points relative to the position of the reorganization points. The results are in agreement with those shown by Yao *et al.*[14]

The bottom line of Table 3 represents the sum of the partial costs. The large number of reorganizations (467 in this case) is a consequence of having to search records in overflow. The computation of the cost measures concludes with the value of total cost of operation upon the file throughout its lifetime.

A synoptic presentation of the many further results which follow for other file organizations is adopted for convenience. Table 4, for instance, summarizes the total cost figures for the six organizations considered, and is included so that the effect of modifications on type of record, use of the file and file size can be reviewed here.

3.1 Effects of modification on record type and use of file

As an illustration of the sort of use to which the system can be put a series of three modifications to the file described in Table 1 is made. In the first modification the use to which the file is put is varied and the results of indexed-sequential and direct organizations noted. In Table 1 the 1500 retrievals per day are broken down into 500 random retrievals and 1000 successor record retrievals. These retrievals are redistributed and the results summarized in modification A of Table 5 so that there are 1400 random retrievals and 100 successor retrievals. Since this type of use takes little advantage of the sequential component of the indexed seqential organization, it can be seen that this form of organization now becomes more expensive than that of direct file organization where there is no difference between a random or a successive record retrieval.

A second modification is chosen to illustrate the occasion when a multiring file may be preferred. Here both changes of the use of the file and also changes to the type of record involved are made. These changes are summarized as follows:

(i) The frequency of record insertions is set higher to

Table 2. Performance of sequential file

Day no.	Records in file	Fetch (ms)	Fetcho (ms)	Next (ms)	Insert (ms)	Update (ms)	Delete (ms)	Read (ms)	Readser. (ms)	Reorgan. (ms)
0	50,000	527	3954	2.2	55.6	543	543	7907	7907	15,814
61	53,655	531	4243	2.2	55.6	548	548	8485	8490	16,975
122	57,389	550	4538	2.3	55.6	567	567	9076	9553	18,615
183	61,523	544	4865	2.2	55.6	561	561	9729	9877	19,602
244	65,779	556	5201	2.3	55.6	572	572	10,402	10,813	21,204
304	70,545	556	5578	2.3	55.6	573	573	11,156	11,453	22,600
365	75,356	558	5958	2.3	55.6	575	575	11,917	12,136	24,046
426	80,881	571	6395	2.3	55.6	587	587	12,791	13,317	26,093
487	86,410	569	6832	2.3	55.6	585	585	13,665	13,991	27,646
548	92,694	577	7329	2.3	55.6	594	594	14,659	15,141	29,786
609	99.029	572	7830	2.2	55.6	589	589	15,661	15,852	31,506
670	106,212	580	8398	2.3	55.6	597	597	16,796	17,125	33,911
731	113,505	578	8975	2.2	55.6	594	594	17,950	18,065	36,011

Table 3. Cost of sequential file

Day no.	Records in file	Initial search (£/day)	Search incr. (£/d^2)	No. of reorgan.	Cost of reorgan. (£)	Total proces. (£)	Total storage (£)	Total No. reorg. (£)	Total (£)
61	53,655	1.2	0.017	33	1.9	68.8	8.6	105.3	77.4
122	57,389	1.2	0.017	31	1.9	69.3	9.2	167.7	78.5
183	61,523	1.3	0.022	36	2.3	80.2	9.8	267.6	90.0
244	65,779	1.4	0.029	34	2.3	81.3	10.5	352.0	91.8
304	70,545	1.7	0.029	38	2.8	93.1	11.3	498.6	104.4
365	75,356	1.7	0.038	37	2.9	94.7	12.1	610.8	106.8
426	80,881	1.9	0.038	40	3.4	108.4	12.9	823.2	121.3
487	86,410	2.0	0.049	40	3.6	110.9	13.8	978.0	124.8
548	92,694	2.2	0.049	43	4.1	125.9	14.8	1271.6	140.7
609	99,029	2.3	0.064	43	4.4	129.5	15.9	1483.3	145.3
670	106,212	2.6	0.064	46	5.1	146.4	17.0	1887.5	163.4
731	113,505	2.6	0.084	46	5.4	151.2	18.2	2177.8	169.4
Sum				467	40	1260	154	10623	1414

Table 4. Comparison of total costs (£) for six file organizations

Organization	Day number 183	365	548	731
Indexed-sequential	27.4	34.3	43.6	51.5
Direct	42.6	52.4	63.9	83.1
Inverted	99.3	128.6	153.8	209.6
Multiring	200.6	245.7	336.0	421.3
Sequential	90.0	106.8	140.7	169.4
Pile	1903.2	2695.4	4276.7	6147.6

Table 5. Effect of 3 modifications on costs of selected file organizations

Modification	Organization	Day number 183	365	548	731
A	Indexed-sequential	123.8	160.4	201.2	244.9
	Direct	117.4	145.5	180.6	229.7
B	Inverted	260.6	268.0	286.1	292.6
	Multiring	241.1	245.4	265.5	270.5
C	Sequential	1814.5	2608.8	3893.2	5660.0
	Pile	1507.3	2215.9	3382.8	4994.1

1450 per day and random record retrievals to 50 per day to correspond to a situation where frequent reports on subsets of file information are wanted.

(ii) The frequency of record insertion is set lower to 20 per day and deletions to 10 per day to correspond to a less active file.

(iii) The average number of attributes in a record is set higher at 8 to correspond to the need to store and maintain more index blocks.

(iv) The user activity increase period is ajusted to 365 days to avoid influencing the multiring organization more than the inverted file which in this case has a period of 380 days. Modification B in Table 5 shows that in this case the multiring file has a lower total cost to that of an inverted file organization.

A third experiment conducted on the model was to vary record type and use in a way which mimics the features of a typical medical file. Here records have many attributes, there is less file activity and retrievals are required according to different attributes. Specifically there are now 50 attributes, 200 record retrievals per day (100 based on no-key attribute), and 50 successor record retrievals required per day. Results are summarized in modification C in Table 5 and show the total cost of the pile organization to drop lower than that of the sequential organization. This is due largely to the advantage of the pile organization to implement variable record size and the disadvantage of the sequential organization in requiring a considerable time to retrieve a long fixed length record, with a no-key attribute as argument.

The user of the model would be able to obtain full details of changes in the form depicted in Tables 2 and 3. Presentation in the form of Tables 4 to 6 is intended to summarize the range of use to which the model may be put.

3.2 Effects of modification of file size

Table 6 is included to indicate the way in which the model may be used to investigate appropriate organizations for differing file sizes. the investigations above concerned a 'medium' sized file of 50,000 records, but in this section a 'small' file of 5000 records and a 'large' file of 500,000 records are examined.

Table 6. Effect of file size on costs for selected file organizations

Organization	File size	Day number 183	365	548	731
Sequential	Small	19.9	25.0	31.7	40.1
	Large	3163.3	3893.3	4969.7	6152.7
Pile	Small	54.9	81.8	123.2	185.4
Indexed-	Small	7.9	9.8	12.1	15.0
sequential	Large	806.3	981.8	1228.1	1502.3
Direct	Small	12.2	15.2	19.0	23.8
	Large	1187.3	1483.2	1859.3	2338.6
Inverted	Small	23.8	30.2	38.7	49.5
	Large	3170.6	4254.3	5377.3	7503.7
Multiring	Small	42.8	54.6	70.6	91.3
	Large	6436.1	8118.9	10,672.8	13,655.7

For the small file, all six files organizations are considered, but in this case a user activity period of 75 days is chosen for the reasons outlined in Section 2.3. The user activity parameters are decreased by a factor of 10 to correspond to the decrease in file size for this range of test runs and is chosen as a reasonable, though not necessary, consequence of file size reduction. It assists in comparison with the performance of the medium file size.

The output presentation again takes the form shown in Tables 1 and 2 but for brevity is here summarized as shown in Table 6. The values of both performance and costs are much lower for the reduced file size and are of interest, when compared to the medium file size.

The same series of modifications as those described in Section 3.1 is repeated for the small file size with proportional changes in input data. The modifications A, B and C had a similar effect with the small file size as were noted with the medium file. These modifications describe situations where there is a cost value reversal achieved by choice of organization.

Table 6 also shows total costs incurred when a large file size was organized in five of the six ways discussed earlier. The pile organization is omitted since it is inadequate for a file this size. Again the above results are consistent with those for the medium file size and can be fully explained.[16] The user activity parameters are increased by the same factor of 10, as was the file size, in order to assist the comparison with the medium file size.

Simular cost value reversals as were noted before are observed when the modifications A and B described in Section 3.1 were repeated for the large file size.

These experiments are of course far from exhaustive but none the less demonstrate realistic model behaviour and how performance and costs can be analysed under varying situations.

4. CONCLUSION

A dynamic analytical model has been developed to serve as an aid in quantifying and comparing the performance and cost of file organization. Quantification is necessary because general guidelines for file organization are not easy to establish and each situation needs to be examined individually.

Several file models are demonstrated with varying levels of analysis, input and output data. They all show situations where a particular file organization represents an advantage over others and is consistent with the work of others.[5,8,17,7,11] There is no universally optimal file organization nor any general rule available for selecting an optimal file organization for a particular application.

The model described[16] is among few existing models which isolate and study a range of physical organizations, analyse and compute the evolution of performance and cost parameters with time, and present these figures in a detailed descriptive way.

The dynamic approach is not found elsewhere, and enables total processing and storage costs to be estimated throughout the file's lifetime. The problem of reorganization is treated in detail and optimal reorganization point computed. The effect of user requirements on the number of reorganizations and on the cost parameters is analysed. The ability to apply different types of user activity increase provides a flexibility of adjustment in keeping with the real situation. Interaction with the model is available in conversational mode enabling the evaluator to conduct an on-line physical organization analysis. It is an easily used and interpreted operational tool, that facilitates the process of file organization, evaluation and selection.

REFERENCES

1. D. Lefkovitz, *File Structures for On-Line Systems*, Sparton Books, New York (1969).
2. T. W. Olle, Data structures and storage structures for generalised file processing. *Proc. of the File 68 IAG Conf. on File Organisation*, Copenhagen (1968).
3. M. E. Senko, V. Y. Lum and P. J. Owens, A file organisation evaluation model (FOREM). *Info. Processing 68*, pp. 514–519, North-Holland (1968).
4. V. Y. Lum, H. Ling and M. E. Senko, Analysis of a complex data management access method by simulation modelling. *Proc. 1970 Fall Jt. Comp. Conf.*, pp. 211–222, AFIPS (1970).
5. A. F. Cardenas, Evaluation and selection of file organisations—a model and system. *CACM* **16** (9), 540–548 (1973).
6. D. G. Severance and A. G. Merten, Performance evaluation of file organisations through modelling. *Proc. 27th Nat. Conf. ACM*, pp. 1061–1072 (1972).
7. K. F. Siler, A stochastic evaluation model for database organisations in data retrieval systems. *CACM* **19** (2), 84–95 (1976).
8. L. D. Martin, A model for file structure determination for large on-line data files. *Proc. of the File 68 IAG Conf. on File Organisation*, Copenhagen (1968).
9. A. F. Cardenas, Analysis and performance of inverted data base structures. *CACM* **18** (5), 253–263 (1975).
10. S. J. Waters, Analysis of self-indexing disc files. *The Computer Journal* **18** (3), 200–205 (1974).
11. S. B. Yao and A. G. Merten, Selection of file organisation using an analytic model. *Proc. of Int. Conf. on VDLB*, pp. 255–267, *ACM* (1975).
12. G. Weiderhold, *Database Design*, McGraw-Hill, Kogashuka, (1977).
13. B. Schneiderman, Optimum data base reorganisation points. *CACM* **16** (6), 362–365 (1973).
14. S. B. Yao, K. S. Das and T. J. Teorey. A dynamic database reorganisation algorithm. *ACM Trans. on Database Systems* **1** (2), 159–174 (1976).
15. P. Hammad and J. Raviart, Formulation of choice criteria for file organisations, *Information Systems*, **3**, 123–130 (1978).
16. P. Coumpas, Performance and cost of file organisations using a dynamic analytical model. *M.Sc. Thesis*, Heriot-Watt University (1981).
17. M. E. Senko, Semi-operational evaluation of file modelling techniques, final report. IBM San Jose Research Lab., California, February (1971).

Received August 1982

APPENDIX

List of symbols

a	Number of different attributes in the file
a'	Average number of attributes in a record
a_u	Number of attributes to be modified in a record
A	Average space required for attribute name
b	Number of blocks per track
B	Block size
c	Computational time required for hashing operation
C_{bd}	Cost of storage per byte and day
C'_0	Initial search cost per day
C_{pr}	Cost of processing per unit time
C'_{r0}	Initial reorganization cost
C_{rki}	Reorganization cost at a point in time
C'_{S0}	Initial storage cost
C_{sti}	Storage cost at a point in time
C_{ti}	Search cost at a point in time
C'_{ti}	Search cost at a point in time after reorganization
d	Number of record deletions per day
I	Number of record insertions per day
k	Number of tracks per cylinder
m	Number of available slots for records
m'	Number of available slots after reorganization
M	Number of record modifications per day
M_k	Number of record-key modifications per day
n	Number of records in file
n'	Number of records in file after reorganization
n_i	Number of records in file plus record modifications after some time
n'_i	Number of records in file plus record modifications after reorganization
n_{ki}	Number of records in file plus record-key modifications after some time
N_c	Number of seeks required for the cylinders occupied by the index
N'_c	Number of seeks required for the cylinders occupied by the index after reorganization
N_1	Number of blocks required for a 1st level index
o	Number of records in overflow
o_i	Number of records in overflow after some point in time
o'_i	Number of records in overflow after reorganization
p_b	Probability that an attribute lies in a new first level index block

p_{ov}	Probability that a record is in the overflow area
p	Space required for a pointer
r	Rotational latency time
R	Record space
s	Average seek time
S	Storage space for index
S'	Storage space for index after reorganization
t	Transfer rate
t'	Bulk transfer rate
t_i	Time in days
T_d	Record deletion time
T_f	Record fetch time
T_{f0}	Record fetch time for other than the key attribute
T_i	Record insertion time
T_n	Next record retrieval time
T_u	Update record time
T_{uk}	Update record-key time
T_x	Read entire file time
T_{xs}	Read serially entire file time
T_y	Reorganization time
V	Average space for value part of an attribute
x	Number of index (or ring) levels
$\lfloor\,\rfloor$	Integer part of decimal quantity
$\lceil\,\rceil$	Next higher integer
θ_1	Search cost increase factor without reorganization
θ_2	Search cost increase factor assuming continuous reorganization
θ_S	Storage cost increase factor
μ	Percentage of rings in which attributes are unordered
σ	Number of standard deviations on either side of mean

Sequential file organization

The data records are ordered into a specific sequence according to the key attribute and the individual records contain all the data attribute values in the same order and position. Sequential organization occupies the least possible space but leads to difficulties with record insertion and deletion.

Performance

$T_f = (s + r + B/t)\log_2(nR/B) + (I + M_k)Rt_i/2t'$ where $R = aV$

$T_{fo} = [n + (I + M_k)t_i]R/2t'$

$T_n = (r + B/t)/\lfloor B/R\rfloor + (1 - 1/\lfloor B/R\rfloor)(I + M_k)^2 t_i^2 R/(2t'[n + (I + M_k)t_i])$

$T_i = s + 3r + B/t$

$T_u = \mathrm{T}_f + 2r$

$T_{uk} = T_f + 2r + T_i$

$T_d = T_f + 2r$

$T_x = [n + (I + M_k)t_i]R/t'$

$T_{xs} = (2Rt_i/t')(I + M_k)\log_2[(I + M_k)t_i] + T_x$

$T_y = (n + n')R/t' + (I + M_k)(Rt_i/t')[1 + 2\log_2(I + M_k)t_i]$

Pile file organization

The records are of variable length and need not have similar data files. The pile records are arranged sequentially in order of arrival and since records differ, each record contains identification of the meaning of its elements.

Performance

$T_f = n_iR/2t'$ where $n_i = n + (I + M)t_i$ and $R = a'(A + V + 2)$

$T_n = T_f$

$T_i = s + 3r + B/t$

$T_u = T_f + 2r + T_i$

$T_d = T_f + 2r$

$T_x = n_iR/t'$

$T_{xs} = T_x + 2n_i(R/t')\log_2 n_i)$

$T_y = T_x + n_i' R/t'$ where $n_i' = n + (I - d)t_i$

Indexed-sequential file organization

The index for the data records is parallel in sequence to the file. There are several index levels the first level of which is kept on the same cylinder with the data area and the overflow area. The first level of index has one entry for every data block and the other levels one entry per index block for the immediately lower level. In order to avoid modifications of the index between reorganizations due to insertions, only the pointer to the inserted record is placed in the predecessor record; the inserted record itself is placed in the overflow area.

Performance

$T_f = s + 2(r + B/t) + p_{ov}(r + B/t)(1 + p_{ov}/2$, when $x \leqq 2$

$\quad = 2s + x(r + B/t) + p_{ov}(r + B/t)(1 + p_{ov}/2)$, when $x > 2$

$T_{f0} = [n + (I + M_k)t_i]R/2t'$

$T_n = (r + B/t)/\lfloor B/R\rfloor + 2p_{ov}(r + B/t)(1 - 1/\lfloor B/R\rfloor)$

$T_i = T_f + 5r + B/t$

$T_u = T_f + 2r$

$T_{uk} = 2T_f + 7r + B/t$

$T_d = T_f + 2r$

$T_x = [n + (I + M_k)t_i]R/t'$

$T_{xs} = nR/t' + (I + M_k)(r + B/t)t_i$

$T_y = T_{xs} + [n + (I - d)t_i]R/t' + [S + (I - d)t_i y/\lfloor B/R\rfloor]/t'$

Direct file organization

Records are of fixed length, with m the number of record slots available for use, greater than n the number of records. Collisions are put into an overflow area similar to that of the indexed sequential file, and chained to records in the primary area.

Performance

$T_f = (s + r + B/t)(1 + n_{ki}/2m)$ where $n_{ki} = [n + (I + M_k)t_i]$

$T_{f0} = (m + o_i)R/2t'$ where $o_i = n_{ki} - m[1 - \exp(-n_{ki}/m) + \sigma n_{ki}/\sqrt{m}]$ and $R = aV + P$

$T_n = T_f$

$T_i = (s + 3r + B/t)[2 - \exp(-n_{ki}/m)]$

$T_u = T_f + 2r$

$T_{uk} = T_f + 2r + T_i$

$T_d = T_f + 2r$

$T_x = (m + o_i)R/t'$

$T_{xs} = T_x + (2R/t')n_{ki}\log_2 n_{ki}$

$T_y = T_x + cn_i + (2R/t')n_i' + (m' + o_i')R/t'$ where $o_i' = n_i' - m'[1 - \exp(-n_i'/m')] + \sigma n_i'/\sqrt{m'}$

Inverted file organization

There is no sequentiality in this organization based on primary key attribute and hence a gain in flexibility is obtained. If accesses based on several attributes are frequent then the inverted file is to be preferred to the indexed-sequential organization. An index for an attribute consists of an entry, i.e. attribute plus value pointer, for every record having that attribute. Each index may require multiple levels.

Performance

$T_f = (1 + N_c + p_b)s + (1 + x)(r + B/t)$
where $N_c = \lceil Sbk/B \rceil$ if $bkS/B \leqq 2$
$= \lceil (Sbk/B - N_1)/bk \rceil + 1$ otherwise
and $p_b = (I - d)t_i/[o + (I - d)t_i]$

$T_n = s + r + B/t$

$T_i = s + 3r + B/t + a'[s(N_c + p_b) + x(r + B/t)$
$+ (I - d)(s + 2r + 2B/t)t_i/o + 2r]$

$T_u = T_f + 2r + a_u[s(N_c + p_b) + x(r + B/t)$
$+ (I - d)(s + 2r + 2B/t)t_i/o + 5r + B/t]$

$T_d = T_f + 2r + a'[s(N_c + p_b) + x(r + B/t) + 2r]$

$T_x = [n + (I - d)t_i][s + r + B/t'][a'(A + V + 2)/B]$

$T_{xs} = n_i'(s + R + B/t')R/B + (2n_i'R/t')\log_2 n_i'$

$T_y = as(N_c + N_c') + a(r + B/t)(S + S')/B$
$+ aSt_i(I - d)(s + r + B/t)/Bo$
where $S' = S[1 + (I - d)t_{i/n}$
and N_c' corresponds to N_c with S' for S throughout.

Multiring file organization

The basic characteristic of this organization is the efficient processing of those subsets which contain some common attributes. In management information systems where a great percentage of the total number of operations involves summarizing and reporting, multiring files are widely used.

Performance

$T_f = (s + r + B/t)(xn_i'/2)\exp(1/x)$

$T_n = s + r + B/t$

$T_i = a'(T_f + 2r) + (s + r + B/t)$
$\times [1 - a'(\mu/2)n_i' \exp(1/x)] + 2r$

$T_u = a_u\{2(s + r + B/t)n_i' \exp(1/x) + 4r\}$
$+ \{1 - a_u(\mu/2)n_i' \exp(1/x)\}(s + r + B/t) + 2r$

$T_d = a'\{(s + r + B/t)n_i' \exp(1/x) + 2r\} + s + 3r + B/t$

$T_x = n_i'(s + r + B/t)$