The Determination of the Optimum Database Maintenance Points

M. Hatzopoulos
Unit of Applied Mathematics, University of Athens, Panepistimioupolis, Athens (621), Greece
J. G. Kollias
Chair of Computer Science, School of Engineering, University of Patras, Patras, Greece

The initial database search cost deteriorates due to record additions, deletions and updates performed during the system operation. Previous studies proposed strategies for selecting the optimum maintenance points by making various assumptions about the rate of database deterioration, the database planning period, the maintenance interval, etc. However, the studies did not express the optimum maintenance points as a function of the database physical structure. The paper assumes (a) that the rate of additions to the database equals the rate of deletions for each database record type, and (b) that maintenance is performed for the entire database at fixed time intervals and it shows how the maintenance points may be determined for various primitive physical database structures. The results are subsequently ultilized to select the optimum maintenance points for arbitrary database structures.

INTRODUCTION

Database reorganization can be defined as changing some aspects of the way in which a database is arranged logically and/or physically.[1] Reorganization at the logical level is commonly termed *restructuring* while reorganization at the physical level is called *reformatting*. Some examples of restructuring are changing from a one-to-one to a one-to-many relationship and adding, deleting, splitting, combining or renaming attributes. Database reformatting includes the introduction or deletion of a secondary index and changing from hashed to indexed access.

The lowest level of database reorganization (or reformatting) is termed database *maintenance*. The reorganization at this level assumes that the database does not change either its logical or its physical structure and it is based on the observation that the initial performance of the database deteriorates with time. This is because the cost per access increases due to structural inefficiencies caused by record additions, deletions and changes. Database maintenance restores processing efficiency but requires resources to unload and reload the database. The high cost of a maintenance and the fact that the database is unavailable for the duration of the maintenance prohibit frequent maintenances.

Several research efforts have modeled the problem of selecting the optimal maintenance points by determining strategies which aim at minimizing the total cost of access and maintenance. For example, Shneiderman considered a file which is in existence a total time T and determined the optimum maintenance interval by assuming that the deterioration of the search cost grows linearly and that the maintenance cost is known.[2] Yao *et al.* determined the optimal maintenance points by dropping the assumptions of linear deterioration and known database lifetime.[3] Finally, Tuel extended Shneiderman's work and obtained the optimum maintenance interval for linearly growing files.[4]

However, all the above studies do not express the optimum maintenance points as a function of the database physical structure. (A result towards this direction is reported by the authors who determined the optimum maintenance points for the multilist file organization.)[5] The paper assumes (a) that the rate of additions to the database equals the rate of deletions for each record type, and (b) that the maintenance is performed for the entire database at fixed time intervals, and it shows how the maintenance points may be determined without considering the particular database structures. The results are subsequently utilized to demonstrate how the derived maintenance points may be reached for various primitive and for arbitrary database structures. The nature of the results obtained by the study are demonstrated through the presidential database which was introduced in the Fry and Sibley study.[6] It is worth noting that the same database has also been studied (in different context) by Taylor and Frank and Tsichritzis and Lochovsky.[7,8]

DATABASE MAINTENANCE

Consider a database consisting of n record types[7] or segments.[8] The overall planning period is denoted by T and the time between two successive database maintenances by t. Let p_i be the probability that the ith segment will be requested during $T, i = 1, 2, \ldots, n (\sum_{i=1}^{n} p_i = 1)$. Associated with each segment there exists a quantity q_i which measures the frequency that the ith segment will be used during T, $i = 1, 2, \ldots n$. As it will be shown below q_i can be determined by considering p_i and the physical database structure.

Let c_i be the initial search cost for satisfying a query using the ith segment and θ_i be the rate at which c_i deteriorates over time. By assuming that the deterioration of the search cost grows linearly over time then the quantity $c_i + \theta_i t$ is the cost of searching the ith segment

CCC–0010–4620/82/0025–0126$02.00

after a time t since the last maintenance. If we assume that the rate of additions to every database segment equals the rate of deletions then the database maintenance cost, R, may be assumed constant. Using these assumptions the cost of searching a segment of the database is the area under the graph in Fig. 1.[2] Below the optimal maintenance points are determined without considering a particular database structure.

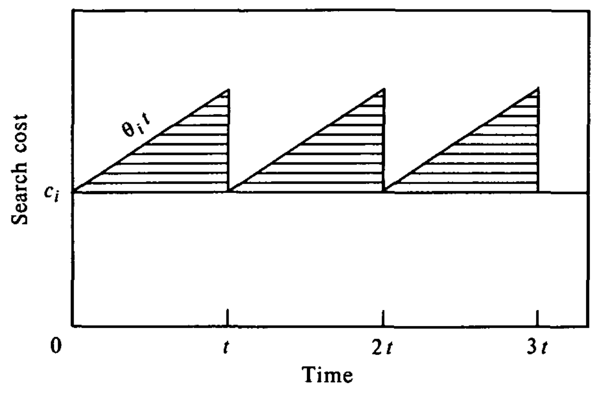

Figure 1. The cost of searching a database segment.

Let $N = T/t$ be the number of maintenances during T. The quantity to be minimized $C(T)$, is the sum of the excess search cost (shaded area in Fig. 1) and the maintenance costs:

$$\begin{aligned} C(T) &= \sum_{j=1}^{N}\left\{\sum_{i=1}^{n}\int_0^t q_i\theta_i t'\mathrm{d}t' + R\right\} \\ &= \sum_{j=1}^{N}\left(\sum_{i=1}^{n}\tfrac{1}{2}q_i\theta_i t^2 + R\right) \\ &= \sum_{i=1}^{n}\tfrac{1}{2}q_i\theta_i Tt + TR/t. \end{aligned}$$

By taking

$$\frac{\mathrm{d}C}{\mathrm{d}t} = \sum_{i=1}^{n}\tfrac{1}{2}q_i\theta_i T - TR/t^2 = 0$$

the optimal maintenance interval is determined by

$$t = \left(2R/\sum_{i=1}^{n}q_i\theta_i\right)^{1/2} \qquad (1)$$

(Note: (1) in Shneiderman can be derived from (1) by taking $n = 1$ and $q_1 = 1$).[2]

THE MAINTENANCE OF PRIMITIVE DATABASE STRUCTURES

Below, the quantities q_i in (1) are estimated for three primitive database structures, $i = 1, 2, \ldots, n$.

Hierarchy with more than one dependent level

Figure 2 illustrates (part of) the presidential database using data structure diagrams.[9] The database contains information about Presidents, Administrations and States Admitted. The segments are named and numbered. The convention employed for numbering the segments is that the more dependent segments are assigned with higher numbers. The symbol '$\longrightarrow$' signifies 1 : n relationships between record types. Therefore Fig. 2 determines a hierarchy with more than one dependent level.

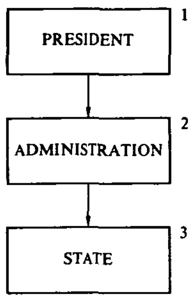

Figure 2. Data structure diagram of hierarchy.

To estimate q_i it is noted that each segment i is searched every time segments $i, i+1, \ldots, n$ are requested. This is because in a hierarchical database structure each dependent segment is always accessed through its father segment. With reference to Fig. 2: $q_1 = \sum_{i=1}^{3}p_i = 1, q_2 = p_2 + p_3$ and $q_3 = p_3$. In general if n levels are involved in the hierarchy of Fig. 2 then $q_i = \sum_{j=i}^{n}p_j$. Therefore (1) becomes $t = (2R/\sum_{i=1}^{n}\theta_i(\sum_{j=i}^{n}P_j))^{1/2}$.

Network involving more than two types of record types

The next primitive database structure is used to represent many-to-many relationships between record types. Figure 3 shows a many-to-many association in the presidential database. The names assigned to the '$\longrightarrow$' declare that two set types named CONGRESS-SERVED and PRESIDENT-SERVED exist between the PRESIDENT and CONGRESS (owners) and CONGRESS-PRESS-LINK (see also Ref. 7, p. 74).

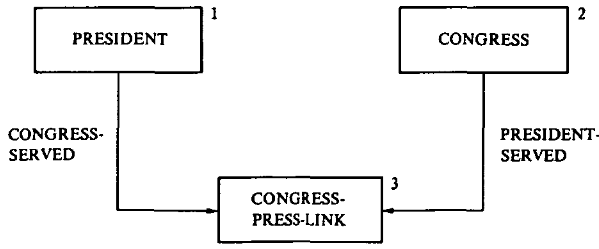

Figure 3. Many-to-many association in Presidential database.

Let p_1, p_2 and p_3 be the probabilities that segments 1, 2 and 3 will be requested. The segment CONGRESS-PRESS-LINK can be accessed via the set CONGRESS-SERVED or PRESIDENT-SERVED. Let p_3^1 and p_3^2 be the probabilities that the 3rd segment will be requested using the PRESIDENT and CONGRESS segments respectively. In fact network database structures allow users to 'navigate' the database using alternative access paths.[10] The qualities q_i are now: $q_1 = p_1 + p_3^1, q_2 = p_2 + p_3^2$ and $q_3 = p_3$.

The CODASYL type of data model allows networks to involve more than two types of record types (see Ref. 11,

p. 239). Specifically, if n record types are to be connected, a connection record type and n set types are introduced. Each of the n record types is made the owner of one of the set types and each connection record occurrence is made a member of exactly one of each of the n types of set and thus represents the connection between the corresponding n record types. If the connection record type is the $n + 1$ segment, then from the arguments above $p_{n+1} = p^1_{n+1} + p^2_{n+1} + \cdots + p^n_{n+1}$, where p^i_{n+1} is the probability to access the connection records via the ith record type ($i = 1, 2, \ldots, n$). The q_i quantities are: $q_i = p_i + p^i_{n+1}, i = 1, 2, \ldots, n$; $q_{n+1} = p_{n+1}$. Therefore the quantity (1) becomes for the database concerned

$$T = \left(2R/\left(\sum_{i=1}^{n} (p_i + p^i_{n+1})\theta_i + p_{n+1}\theta_{n+1}\right)\right)^{1/2}$$

Hierarchy with more than one type of record at a dependent level

Figure 3 illustrates the case by considering (part of) the presidential database which contains information about the presidents, the elections they won, the administrations they headed and the congress served.

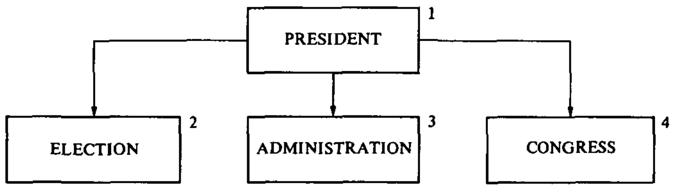

Figure 4. Part of the Presidential database.

From Fig. 4 it follows that $q_1 = 1, q_2 = p_2, q_3 = p_3$ and $q_4 = p_4$. Suppose now that a segment at level 2 is the father node of another segment. For example, Fig. 6(a) in Ref. 8 is exactly like Fig. 4 but the segment ADMINISTRATION has as a son the segment STATE ADMITTED. This last segment takes the number 5.

Let S_i be the set of the segment numbers which are sons of the ith segment, with $S_i = \emptyset$ indicating terminal segments. Then q_i is estimated by the recursive formula $q_i = p_i + \sum_{j \in S_i} q_j$. With reference to the database of Fig. 4 extended with the segment STATE ADMITTED it is noted that $S_2 = S_4 = S_5 = \emptyset$. Therefore $q_5 = p_5, q_4 = p_4$, $q_3 = p_3 + p_5$, $q_2 = p_2$ and $q_1 = p_1 + q_2 + q_3 + q_4 = \sum_{i=1}^{5} p_i = 1$.

THE MAINTENANCE OF ARBITRARY DATABASE STRUCTURES

In this section a method is presented which aims at determining the quantities q_i in (1) (and hence the optimum maintenance points) for arbitrary database structures. The method is applied using the complete presidential database as shown in Fig. 5. Individual segments may be accessed using one of the 4 entry points. (These points are indicated in Ref. 7, p. 76 using the CODASYL construct SYSTEM.)

Suppose that the n database segments are nodes of a directed graph.[12] Consider that the database possesses m

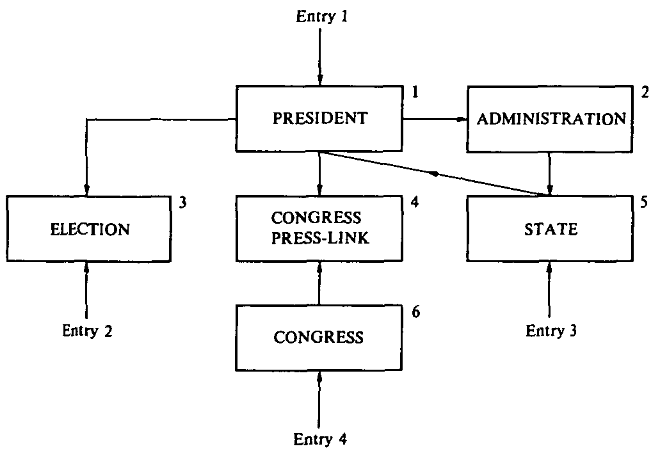

Figure 5. Data structure diagram of Presidential database.

entry points ($m < n$). Let p^j_i be the probability of using the ith segment when the database has been entered through the jth entry point ($i = 1, 2, \ldots n; j = 1, 2, \ldots, m$). Let also Q^j_i be the frequency that the ith segment is requested by application programs via the jth entry. By definition $p_i = \sum_{j=1}^{m} p^j_i$ and $q_i = \sum_{j=1}^{m} Q^j_i$ (2). From (1) and (2) it follows that the optimum maintenance points can be determined if the quantities Q^j_i can be estimated.

To calculate Q^j_i the arguments in the preceding section are generalized as follows: Associated with each segment i there exists a set $\mathbf{D}_i$ which contains the number of the segments which can be directly reached via segment i. The variables Q^j_i are defined from the recursive formula $Q^j_i = p_i + \sum_{r \in \mathbf{D}_i} Q^j_i$(3).

By applying formula (3) in the database structure of Fig. 5 the following results are obtained: $Q^1_1 = Q^1_2 + Q^1_3 + Q^1_4$ (because there exist arcs which connect segment 1 with segments 2, 3 and 4) $= p^1_3 + p^1_4 + p^1_2 + Q^1_5$ (because there is an arc from segment 2 to 5 and because there is not an outgoing arc from segments 3 and 4) $= p^1_3 + p^1_4 + p^1_2 + p^1_5$. Similarly, $Q^1_2 = p^1_2 + p^1_5$, $Q^1_3 = p^1_3$ (because the 3rd segment is a terminal node), $Q^1_4 = p^1_4$ and $Q^1_5 = p^1_5$. With reference to the second entry point one gets $Q^2_1 = p^2_1 + p^2_3 + p^2_4 + p^2_2$, $Q^2_2 = p^2_2$, $Q^2_3 = p^2_3$, $Q^2_4 = p^2_4$ and $Q^2_5 = p^2_5 + p^2_1 + p^2_2 + p^2_3 + p^2_4$. Finally, $Q^3_3 = p^3_3$, $Q^4_4 = p^4_4$ and $Q^4_6 = p^4_6 + p^4_4$.

Using the above quantities in (2) we get $q_1 = Q^1_1 + Q^2_1$, $q_2 = Q^1_2 + Q^2_2$, $q_3 = Q^1_3 + Q^2_3 + Q^3_3$, $q_4 = Q^1_4 + Q^2_4 + Q^4_4$, $q_5 = Q^1_5 + Q^2_5$ *and* $q_6 = Q^4_6$. Substituting the six quantities in (1) the optimal maintenance points are obtained.

CONCLUSION

Previous studies on database maintenance emphasized on different maintenance strategies without considering the particular database physical structure employed. The present study demonstrated through a strategy, borrowed from a paper by Shneiderman,[2] how the database structure affects the optimum maintenance points. The formulae reported may be utilized by database administrators to determine the appropriate time interval for performing the database maintenance. The results can be easily modified to extend also the strategies suggested by Yao *et al.*[3] and Tuel[4] to select the optimum maintenance points for arbitrary databases.

REFERENCES

1. G. H. Sockut and R. P. Goldberg, Database reorganisation—principles and practice. *Computing Surveys* **11** (No. 4), 371-395 (1979).
2. B. Shneiderman, Optimum database reorganisation points. *Communications of the ACM* **16** (No. 6), 362-365 (1973).
3. S. B. Yao, K. S. Das and T. J. Teorey, A dynamic database reorganisation algorithm. *ACM Transactions on Database Systems* **1** (No. 2), 159-174 (1976).
4. W. G. Tuel, Optimum reorganisation points for linearly growing files. *ACM Transactions on Database Systems* **3** (No. 1), 32-40 (1978).
5. M. Hatzopoulos and J. G. Kollias, The reorganisation of the multilist file organisation. *Angewandte Informatik* No. 10/80, 424-426 (1980).
6. J. P. Fry and E. H. Sibley, Evolution of Database Management Systems, *Computing Surveys* **8** (No. 1), 7-42 (1976).
7. R. W. Taylor and R. L. Frank, CODASYL Database Management Systems. *Computing Surveys* **8** (No. 1), 67-103 (1976).
8. D. C. Tsichritzis and F. H. Lochovsky, Hierarchical Database Management. *Computing Surveys* **8** (No. 1), 105-123 (1976).
9. C. W. Bachman, The evolution of storage structures. *Communications of the ACM* **15** (No. 7), 628-634 (1972).
10. C. W. Bachman, The programmer as navigator. *Communications of the ACM* **16** (No. 11), 653-658 (1973).
11. C. J. Date, *An Introduction to Database Systems*, The Systems Programming Series, Addison-Wesley, Reading, Massachusetts (1975).
12. D. E. Knuth, *The Art of Computer Programming*, Vol. 1, Fundamental Algorithms, Addison-Wesley, Reading, Massachusetts (1968).

Received February 1981

Book Reviews

Continued from p. 160

James L. PETERSON
Petri Net Theory and the Modeling of Systems
Prentice-Hall, New York, 1981. 290pp. £16.05.

This book's preface claims that 'the presentation and organization is (*sic*!) suitable both for individual study by the practicing professional and for organized graduate study in computer science'. I write from the point of view of the former but also as someone who has been waiting for a good book on the subject to turn up.

The book has satisifed part of my needs. It is a readable introduction to what Petri nets are and to the formal theory related to them. It contains a large annotated bibliography (although only up to 1979) which would be of use to anyone studying the subject further. It has exercises and topics for further study, which I presume would be of use for classes, but there are no answers, which I find annoying. The most serious deficiency is that the examples of using Petri nets are too brief and somewhat dated. While some 40 pages are devoted to modelling, one thorough example would have been worthwhile. I am left with the impression that Petri nets have become just an interesting topic in pure mathematics, for which the book serves as good text. That may be true; otherwise someone needs to write a book on how to use Petri nets to model and analyse systems.

PETER RADFORD
Chelmsford

C. E. VANDONI (ED.)
Eurographics 80
North Holland, Amsterdam, 1980. 340pp. $44.00.

This is the complete proceedings of the Eurographics 80 Conference and Exhibition held at the University of Geneva in September 1980. I always approach such a volume with some concern as I know that I will be faced with countless different styles of presentation, different type faces, poorly reproduced figures, etc. Eurographics 80 did not let me down—indeed there are examples of *hand drawn* graphics in the text, quite incredible at a graphics conference.

Some twenty-nine papers presented are subdivided under seven subject headings: Man-Machine Interaction based on Core System, Algorithms, Special Purpose Graphics Packages, Interdisciplinary Applications, General Purpose Graphics Packages, Distributed Graphics and Raster Graphics Systems. There is also a section entitled 'Industrial Seminar' which was obviously linked to the exhibition and should have been omitted as it serves no useful purpose in this volume. It is impossible to review all the papers but as is always the case there are several useful papers and a few which manage to say very little in anything up to eight pages of text.

In the Algorithms section the paper by K. Altimimi (Perkins Engines) describes an interesting technique for three-dimensional Mesh Generating using a Turnkey graphics system (Computervision). Under Graphics Packages P. Comninos and G. Webster describe CGAL currently under development at Teesside Polytechnic—an example here of bad proof reading with viewport and viewpoint being confused. In the Interdisciplinary Applications DRAW (K. Bicknell, Rothamsted) shows how the apparently trivial exercise of producing chemical formulae is in fact a very interesting piece of work. On the other hand J. Weber, G. Bernardinelli, J. J. Combremont and M. Roch (University of Geneva) convince me even more that three-dimensional ball and stick representation of molecules can only be satisfactorily displayed with grey scales pictures—their line plots just do not work. Simpleplot makes its customary appearance in the General Purpose Graphics Packages, this time in Algol68 guise. The surprisingly small section in Raster Graphics has a couple of interesting papers—particularly Computrol® (S. E. Ranjbaran, R. J. Swallow of Advance Technology Systems, New Jersey).

Eurographics 80 is a record of what appears to have been a useful conference—but I do wish publishers would spend a little less on producing hard back (paper back would be sufficient) covers and more on standardizing the presentation.

K. P. TEARE
Salford

Continued on p. 83