CONCURRENT CONTROL OF PARTIAL MATCH QUERIES FOR MULTIDISK MKH FILES

H. F. Lin and C. Y. Chen***

* H. F. Lin is with the Institude of Information Engineering, Feng Chia University, Taichung, Taiwan 40724, R. O. C. E-mail: hflin@fcu.edu.tw.

** C. Y. Chen is with the Department of Electronics, Feng Chia University, Taichung, Taiwan 40724, R. O. C. E-mail: chihchen@fcu.edu.tw.

ABSTRACT

Since more and more queries may occur at the same time for a large file system, in order to increase query throughput per unit of time and further reduce average query response time, in this paper, we propose a new multidisk MKH file design scheme. It is seen that an MKH file obtained from the proposed scheme does guarantee certain partial match queries of different types to be answered concurrently .

Keywords: multidisk file system, partial match query, query response time, concurrent control, query throughput

1. Introduction

Because files become more and more large in real applications, the design of a large multiattribute file in a multidisk system such that the average response time over all possible queries is minimized is one of the most concerned database research issues in recent years[1-28,30,32-33].

In an information retrieval system, a file is a collection of records, a multiattribute file is a file whose records are characterized by more than one attribute, and a query is a specification of values of the attributes which is used to retrieve the specified records from the file.

The partial match query (PMQ) is the most commonly used query type for multiattribute files. Let there be a file with N attributes A_1, A_2, Λ, A_N and N corresponding domains D_1, D_2, Λ, D_N . A PMQ is a query retrieving all records of the form ($A_1 = a_1$, $A_2 = a_2$, Λ, $A_N = a_N$), where $a_i, 1 \le i \le N,$ is either a key belonging to D_i or is unspecified (i.e., a don't care condition), in which case it is denoted by "$*$". For instance, $q = (a, *, b)$ denotes a PMQ to retrieve the records with the first attribute $A_1 = a$, the third attribute $A_3 = b$, and the second attribute arbitrary from some three-attribute file. Let $Q = \{A_{i_1}, A_{i_2}, \Lambda, A_{i_n} | 1 \le i_1 < i_2 < \cdots < i_b \le N\} \subseteq \{A_1, A_2, \Lambda, A_N\}$. We say that a PMQ is of type Q , denoted as q_Q or $q_{i_1, i_2, \Lambda, i_b}$, if the set of attributes specified in the query is equal to Q . Accordingly, there are totally 2^N different query types for an N-attribute file.

The multidisk file design problem generally consists of first organizing a given set of records into a fixed number of buckets in such a way that the average number of buckets need to be examined, over all possible queries, is minimized; and then allocating the buckets onto a fixed number of independently accessible disks in such a way that the disk access concurrence is maximized and therefore the average response time over all possible queries is minimized. It should be pointed out that both the record organization problem and the bucket allocation problem for PMQs have been shown to be NP-complete problems [13,32-33]. Hence all design schemes that have been proposed so far are all heuristics [1-28,30,32-33], meaning that they guarantee some optimalities under some

particular conditions while give near optomal or good performances in the general case.

However, among the so far proposed heuristic record organization schemes [2,23,25-26,28,30], the multiple key hashing (MKH) file concept suggested by Rothnie and Lozano [30] has been shown to be very effective for PMQs [3,7,11,12,25,30]. Hence, almost all researches concerning the bucket allocation problem were focused on MKH files [1,5,6,9-10,14-22,24,31]. By an N-attribute MKH file with attributes A_1, A_2, Λ, A_N and corresponding domains D_1, D_2, Λ, D_N, we mean an N-attribute file in which each record (a_1, a_2, Λ, a_N), $a_i \in D_i$ for $1 \le i \le N$, is assigned into a bucket denoted as $[h(a_1), h(a_2), \Lambda\ h(a_N)]$, where h_i is a hashing function from D_i to the set $\{0,1,\cdots, m_i - 1\}$ for $1 \le i \le N$ and $\prod_{i=1}^{N} m_i$ equals the total number of available buckets. An MKH file constructed above is often denoted as $< m_1, m_2, \Lambda, m_N >$. For instance, consider a simple case where N=2, $D_1=D_2=\{a,b,c,d\}$, $h_1(x)=0$ if x=a,b; 1 if x=c,d, $h_2(y)=0$ if y= a,b; 1 if y=c; and 2 if y=d. Then we have a 2-attribute MKH file <2, 3> consisting of the following six buckets: [0, 0]={(a, a), (a, b), (b, a), (b, b)}, [0, 1]={(a, c), (b,c)}, [0, 2]={(a, d), (b, d) }, [1, 0]={(c, a), (c, b), (d, a), (d, b)}, [1, 1]={(c, c), (d, c)}, and [1, 2]={(c, d), (d, d)}. Assume that both overflow and underflow problems are ignored. Then the buckets to be examined by the PMQ q=(c, *) are [1, 0], [1, 1], and [1, 2].

In the bucket allocation problem, it is assumed that the m disks can be accessed independently. Also, it is assumed that the retrieval of one bucket takes one unit of time. Therefore, the time taken to respond to a query can be simply measured in terms of the maximum number of buckets needed to be accessed on a particular disk. Accordingly, let R(q) denote the set of all qualifying buckets for a query q. Then a lower bound to the response time of q is $\lceil |R(q)| / m \rceil$, where m is the total number of available disks. A bucket allocation method that minimizes the average response time over all possible PMQs is called an optimal allocation method [19]. An allocation method that minimizes the response time of each PMQ is called a strictly optimal allocation method [19]. If, in addition, a strictly optimal allocation method also assigns all buckets uniformly among the disks, it is called a perfectly optimal allocation method [17].

Although there has been a great progress on the design of multidisk files for facilitating PMQs in the past years [1-28,30,32-33]; however, in all the previously suggested design schemes, queries can be answered only in a sequential way; i.e., only one query can be processed at one time. Since, in real applications, more and more queries may occur at the same time for the same file, it is urgently expected to concern the problem of answering multiple queries for a file concurrently to increase query throughput per unit of time in addition to reduce average query response time. Unfortunately, to our knowledge, so far the problem has not been addressed.

Accordingly, in this paper, we are concerned with the problem of multidisk MKH file design for facilitating concurrent control of PMQs. Based upon the concept given in [27] that any record clustering scheme (i.e., record organization scheme) often "biased" toward the most common queries, and different queries often have significantly different clustering requirements, hence no single clustering scheme can satisfy each query; in this paper, we propose a new multidisk MKH file design scheme in which multiple copies of the file are used and each of which is clustered differently. It is seen that a file obtained by using the proposed scheme guarantees certain queries of different types to be answered concurrently.

The proposed scheme and some discussions are given in Section 2. Section 3 contains a small example to illustrate our proposed scheme. Finally, conclusions and further research problems are presented in Section 4.

2. A New Redundant Multidisk MKH File Design Scheme

2.1 The Design Scheme

Let there be a set of N-attribute records, a set of NB buckets and m independently accessible disks. Suppose the probability distribution of all PMQ types is known. Then our file design scheme among m disks which allows certain PMQs of different types to be answered concurrently can be described as the following algorithm.

Algorithm 2.1 : A Redundant Multidisk File Design Scheme

Input: The values of N, NB and m.

Output: A multidisk file for which certain queries of different types are allowed to be answered concurrently.

Steps:1.Partition the set of all attributes $S = \{A_1, A_2, \Lambda, A_N\}$ into r disjoint subsets S_1, S_2, Λ, S_r according to the probability distribution of query types.

2. (1) Reproduce r copies of the given file.

   (2) Cluster the records of the i-th copy on attributes in S_i, $1 \le i \le r$, into an MKH file.

3. Determine, for each query type, a facile copy for processing it .

4.(1) Partition the set of disks $W = \{Disk_0, Disk_1, \Lambda, Disk_{m-1}\}$ into t disjoint subsets W_1, W_2, Λ, W_t.

   (2) Determine, for each copy, a set of disks and allocate all buckets of the copy onto the determined disks.

5. Maintain a table of size 2^N to map each query type to the copy that is used to process all queries of the type. Also maintain another table of size r to map each copy to the disks where the copy is stored.

After constructing a file system according to the above stated scheme, it is easy to determine whether a set of queries of different types for the file can be answered concurrently or not. The determination can be described as the following algorithm.

Algorithm 2.2 : The Determination of Concurrent Control

Input: A set of queries of different types for a file obtained from Algorithm 2.1.

Output: 〝1〞 denoting that the queries can be concurrently processed ; 〝0〞 denoting that the queries can not be concurrently processed.

Steps:1. Determine, for each query, the copy that is used to answer the query.

2.Determine, for each copy obtained in Step 1, the set of disks that are used to store this copy.

3. If the copies obtained in Step 1 are pairwise distinct and the sets of disks found in Step 2 are pairwise disjoint then return 〝1〞 else return 〝0〞 .

2.2 Some Discussions

Let Q be a nonempty subset of the set S of all attributes. Observe that if a file is clustered on the attributes in Q then there is only one bucket need to be examined for each query of type Q. This suggests each copy ought to be clustered on a set of attributes for which the corresponding query type occurs more frequently. Accordingly, the partition $\{S_1, S_2, \Lambda, S_r\}$ of S we made in Step 1 of Algorithm 2.1 is the one for which $\sum_{i=1}^{r} P(q_{S_i})$ is the maximum, where $P(q_{S_i})$ denotes the occurrence probability of a query of type S_i. Usually, the probability distribution of query types for a file can be known by collecting statistics on the various query types when the file has been used for a certain period of time or, by estimating the expected usage of the various query types when the file is being designed.

Since it has been shown in [16,27,31] that the MMI(minimum marginal increase) method which allocates some units to a set of variables, one at a time, in the direction of minimum marginal direction is very effectively for clustering a set of records into an MKH file

for answering PMQs, in Step 2 of Algorithm 2.1, the MMI method is suggested to be used for clustering each copy into an optimal or good MKH file to reduce the number of buckets qualified by each query. For the limitation of space, we don't give a detail introduction for the MMI method here. The interested reader may consult [16,27,31].

The task to determine a facile copy to answer a query in Step 3 of Algorithm2.1 is easy. It should only to select the one which minimizes the number of buckets qualified by the query. Suppose, after clustering each copy C_i on S_i, we have $C_i = <m_1^{(i)}, m_2^{(i)}, \Lambda, m_N^{(i)}>$ for $1 \leqq i \leqq r$ and let q be a query of type Q. Then the number of buckets qualified by q can be computed as $\prod_{A_j \in S-Q} m_j^{(i)} \prod_{A_j \in S-Q} m_j^{(i)}$.

In order to increase the degree of parallel processing for different query types, the number of partitions of all disks is suggested to be equal to the number of partitions for all attributes. Further, in order to reduce the average response time of all queries, the number of disks in each partition for storing a copy file is suggested to be proportional to the maximum number of buckets qualified by the queries answered by the copy.

Although so far there has no general method been proposed for allocating an MKH file optimally on a multidisk system; however, a number of heuristic methods that guarantee optimal allocation performance in certain conditions and provide near optimal or good performance in general case have been suggested [1,5-6,9-10,14-22,24,32]. For instance, consider an MKH file F= $<m_1, m_2, \Lambda, m_N>$ and m disks. If m∈ {2,.3} or m_i mod m∈ {0, 1, m-1} for each $1 \leq i \leq N$, the DM (Disk Modulo) allocation method given by Du and Sobolewski [19] guarantees strictly optimal performance for F. If m_1, m_2, Λ, m_N are pairwise relatively prime, the RNS (Residue Number System) allocation method suggested newly by Lin and Chen [24] guarantees perfectly optimal performance for F. Accordingly, depending on various conditions on the values of N, m and m_i, $1 \leq i \leq N$, we can select a facile method for allocating each copy onto the designate disks to facilitate all queries processed by that copy.

Finally, when the size of the table obtained in Step 5 of Algorithm 2.1 is too large, we may, instead of maintaining the table, store r sets U_1, U_2, Λ, U_r, where U_i is the set of query types processed by the i-th copy C_i. And we can know the copy for a query q by finding which set of $\{U_1, U_2, \Lambda, U_r\}$ containing q.

3. A Small Example

In this section, a small example is given to illustrate how certain queries of different types for a file obtained from Algorithm 2.1 can be answered concurrently.

Let $N = 3$, $NB = 30$ and $m = 3$. Suppose the probability distribution of all PMQ types is known as follows: $P_1 = 0.23$, $P_2 = 0.08$, $P_3 = 0.12$, $P_{12} = 0.06$, $P_{13} = 0.11$, $P_{23} = 0.31$, and $P_{123} = 0.09$. We proceed to construct a file according to Algorithm 2.1 as follows.

Step 1:Since $P_{23} > P_1 > P_{12} > \cdots > P_{12}$ and {2,3} ∪ {1}={1,2,3}, S={ A_1, A_2, A_3 }is partitioned into $S_1 = \{A_1\}$ and $S_2 = \{A_2, A_3\}$

Step 2: Reproduce two copies of the file and cluster them on S_1 and S_2, respectively, into two MKH files $C_1 = <30,1,1>$ and $C_2 = <1,5,6>$ by using the MMI method.

Step 3: Determine, for each query type, a copy that minimizes the number of qualified bucket as shown in Table 3.1.

Step 4:Partition the set of disks $W = \{Disk_0, Disk_1, Disk_2\}$ into $W_1 = \{Disk_0\}$ and $W_2 = \{Disk_1, Disk_2\}$

. Allocate C_1 onto disks in W_1 and C_2 onto disks in W_2, respectively, by the DM allocation method [19].

Note that the use of the DM method is because it has strictly optimal response time performance for each PMQ in a two-disk system [19].

Step 5: The query-copy mapping table and copy –disk mapping table are shown as follows.

Table 3.1

Query types	Used copy	Number of qualified buckets	Response time
q_1	C_1	1	1
q_2	C_2	6	3
q_3	C_2	5	3
q_{12}	C_1	1	1
q_{13}	C_1	1	1
q_{23}	C_2	1	1
q_{123}	C_1	1	1

Table 3.2

Copy	Stored disks
C_1	D_0
C_2	D_1, D_2

Suppose, according to the probability distribution of query types, there are ten queries of various types $q_1, q_1, q_{23}, q_{23}, q_{23}, q_3, q_{13}, q_{123}, q_2$ and q_{12} which occur at the same time for the above obtained file. Consider a query q in $U_1=\{q_1, q_{12}, q_{13}, q_{123}\}$ and a query q^* in $U_2=\{q_2, q_3, q_{23}\}$. Since q and q^* are answered by different copies C_1 and C_2, respectively, which are stored on disjoint sets of disks $\{Disk_0\}$ and $\{Disk_1, Disk_2\}$, they can be processed concurrently. Accordingly, from Table 3.1, we have the total response time for all queries is 9 units of time. Therefore, the query throughput is $10/9 \approx 1.1$ while the average response time is $9/10. \approx 0.9$. Suppose the above queries are answered by a single copy and strictly optimal sequential processing system. The total response time would be 14 and the query throughput and average response time would be $10/14 \approx 0.71$ and $14/10 \approx 1.4$, respectively. It is seen that our proposed method can indeed increase query throughput per unit of time and reduce average response time.

4. Conclusions

In this paper, we have proposed a new multidisk MKH file design scheme. It has been seen that our method does guarantee that certain queries of different types can be answered concurrently. Therefore, the query throughput per unit of time is significantly increased. Further, since redundant copies of the file are used, the average response time over all possible queries is also reduced. However, it is difficult to have a mathematical mode for analyzing the performance on various related parameters. On the other hand, only queries of different types can be parallelly processed by the proposed method. In order to maximize query throughput per unit of time, it is challenging to develop more powerful design schemes that would also guarantee concurrently control for various queries of the same type. This remains to be our further research problem.

REFERENCES

[1] K. A. S. Abdel-Ghaffar and A. El. Abbadi, "Optimal Disk Allocation for Partial Match Queries," ACM Trans. Database Systems, vol. 18, no. 1, pp. 132-156, 1993.

[2] A. V. Aho and J. D. Ullman, "Optimal Partial-Match Retrieval When Fields Are Independently Specified," ACM Trans. Database Systems, vol. 4, no. 2, pp. 168-179, 1979.

[3] A. Bolour, "Optimality properties of Multiple Key Hashing Functions," J. Assoc. Computing, vol. 26, no. 2, pp. 196-210, 1979.

[4] W. A. Burkhard, "Partial Match Hash Coding: Benefits of Redundancy," ACM Trans. Database Systems, vol. 4, no. 2, pp. 228-239, 1979.

[5] M. Y. Chan, "Multidisk File Design: An Analysis of Folding

Buckets to Disks," BIT, vol. 24, pp. 262-268, 1984.

[6] M. Y. Chan, "A Note on Redundant Disk Allocation," IPL, vol. 20, pp. 121-123, 1985.

[7] C. C. Chang, "Optimal Information Retrieval When Queries Are Not Random," Information Sciences, vol. 34, pp. 199-223, 1984.

[8] C. C. Chang, "Application of Principal Component Analysis to Multidisk Concurrent Accessing," BIT, vol. 28, pp. 205-214, 1988.

[9] C. C. Chang and C. Y. Chen, "Gray Code as a Declustering Scheme for Concurrent Disk Retrieval," Information Science and Eng., vol. 13, no. 2, pp. 177-188, 1987.

[10] C. C. Chang and C. Y. Chen, "Symbolic Gray Code as a Data Allocation Scheme for Two-disk Systems," The Computer J., U. K., vol. 35, no. 3, pp. 299-305, 1992.

[11] C. C. Chang, M. W. Du, and R. C. T. Lee, "Performance Analysis of Cartesian Product Files and Random Files," IEEE Trans. Software Eng., vol. 10, no. 1, pp. 88-99, 1984.

[12] C. C. Chang, R. C. T. Lee, and H. C. Du, "Some Properties of Cartesian Product Files," Proc. ACM-SIGMOD Conf., pp. 157-168, 1980.

[13] C. C. Chang and J. C. Shieh, "On the Complexity of File Allocation Problem," Proc. Int'l Conf. Foundation of Data Organization, Kyoto, Japan, pp. 113-115, May 1985.

[14] C. Y. Chen and H. F. Lin, "Optimality Criteria of the Disk Modulo Allocation Method for Cartesian Product Files," BIT, vol. 31, pp. 566-575, 1991.

[15] C. Y. Chen, H. F. Lin, R. C. T. Lee and C. C. Chang, "Redundant MKH Files Design among Multiple Disks for Concurrent Partial Match Retrieval," J. Systems and software, vol. 35, pp. 199-207, 1996.

[16] C. Y. Chen, C. C. Chang and R.C.T. Lee, "Optimal MMI File Systems for Orthogonal Range Retrieval," Information Systems, vol. 18, No. 1, PP. 37-54, 1993.

[17] C. Y. Chen, H. F. Lin, C. C. Chang and R. C. T. Lee, "Optimal Bucket Allocation Design of K-ary MKH Files for Partial Match Retrieval," IEEE Trans. Knowledge and Data Engineering, vol. 9, no. 1, pp. 148-159, 1997.

[18] H. C. Du, "Disk Allocation Methods for Binary Cartesian Product Files," BIT, vol. 26, pp. 138-147, 1986.

[19] H. C. Du and J. S. Sobolewski, "Disk Allocation for Cartesian Product Files on Multiple Disk Systems," ACM Trans. Database Systems, vol. 7, no. 1, pp. 82-101, 1982.

[20] C. Faloutsos and D. Metaxas, "Disk Allocation Methods Using Error Correcting Codes," IEEE Trans. Computers, vol. 40, no. 8, pp. 907-914, 1991.

[21] M. F. Fang, R. C. T. Lee, and C. C. Chang, "The Idea of Declustering and Its Applications," Proc. 12th Int'l Conf. VLDB, Kyoto, Japan, pp. 181-188, Aug. 1986.

[22] M. H. Kim and S. Pramanik, "Optimal File Distribution for partial Match Retrieval," Proc. ACM-SIGMOD Conf., pp. 173-182, 1988.

[23] R. C. T. Lee and S. H. Tseng, "Multikey Sorting," Policy Analysis and Information Systems, vol. 3, no. 2,pp.1-20, 1979.

[24] H. F. Lin and C. Y. Chen,"An RNS Based Data Allocation Method as a Perfectly Optimal Multiattribute Declustering Scheme," in submission to IEEE Int'l Conf. CLUSTER 2000.

[25] W. C. Lin, R. C. T. Lee, and H. C. Du, "Common Properties of some Multi-Attribute File Systems," IEEE Trans. Software Eng., vol. 1, SE-5, no. 2, pp. 160-174, 1979.

[26] J. H. Liou and S.B. Yao, "Multi-Dimension Clustering for Database Organizations," Information Systems, vol. 2, no. 2,pp. 187-198, 1977.

[27] K. Ramamohanarao, J. Shepherd, and R. Sacks-Davis, "Multi-Attribute Hashing with Multiple File Copies for High Performance Partial-Match Retrieval," BIT, vol. 30, pp. 404-423, 1990.

[28] R. L. Rivest, "Partial-Match Retrieval Algorithms," SIAM J. Computing, vol. 14, no. 1, pp. 19-50, 1976.

[29] K. H. Rosen, Elementary Number Theory and Its Applications, 3rd ed., Addison Wesley, 1993.

[30] J. B. Rothnie and T. Rozano, "Attribute Based File Organization in a paged Memory Environment," CACM, vol. 17, no. 2, pp. 63-69, 1974.

[31] J. L. Saaty, Optimization in Integers and Related Extremal Problems, Mc-Graw-Hill, New York, 1970.

[32] Y. Y. Sung, “Performance Analysis of Disk Allocation Method for Cartesian Product Files,” IEEE Trans. Software Eng., vol. 13,no. 9, pp. 1,018-1,026, 1987.

[33] C. Y. Tang, D. J. Buehrer, and R. C. T. Lee, “On the Complexity of Some Multiattribute File Design Problems,” Information Systems, vol.10, no. 1, pp.21-25, 1985.