Multidisk File Design for Concurrently Answering Partial Match Queries

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

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

*** H. F. Lin is with the Institute of Information Engineering, Feng Chia University, Taichung, Taiwan 40724, Republic of China. E-mail: hflin@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, Lin and Chen have proposed in 1988 an MKH file design scheme which guarantees certain partial match queries of different types to be answered concurrently. In this paper, based upon the representation of an integer in a residue number system (RNS), we present a new bucket allocation method, called the RNS allocation method. It is shown that a multidisk file designed through the used of the RNS allocation method can guarantee various partial match queries of the same type to be answered concurrently.

Keywords: Mmultidisk-multiattribute file, partial match query, query response time, concurrent control, query throughput, residue number system

* The corresponding author

1. Introduction

Since files are getting more and more large in real applications, more and more queries may occur at the same time for the same file. Accordingly, the design of a large multiattribute file in a multidisk system such that the average query response time is minimized while the query throughput per unit of time is maximized is one of the most concerned information retrieval research issues in recent years [1-29, 31-33].

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. By a partial match query (PMQ), we mean an access specification of the form $q=(a_1,a_2,\cdots,a_N)$, where each a_i, $1\le i\le N$, is either a specified value belonging to the domain D_i of the i-th attribute or is "*" which denotes a don't care condition (i.e., it can be any value in D_i) [20]. For instance, q=(a, *, b) denotes a PMQ to retrieve the records with the first attribute a, the third attribute b, and the second attribute arbitrarily from some set of 3-attribute records.

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 [1-29,31-33], 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 [2,24,26-27,29,31], the multiple key hashing (MKH) file concept suggested by Rothnie and Lozano [31] has been shown to be very effective for PMQs [3,7,11,12,26,31]. Hence, almost all researches concerning the bucket allocation problem were focused on MKH files [1,5,6,9-10,14-23,25,32]. By an N-attribute MKH file with attributes $A_1,A_2,\cdots,A_N$ and corresponding domains $D_1,D_2,\cdots,D_N$, we mean an N-attribute file in which each record $(a_1,a_2,\cdots,a_N)$, $a_i\in D_i$ for $1\le i\le N$, is assigned into a bucket denoted as $[h(a_1),h(a_2),\cdots h(a_N)]$, 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 $<m_1,m_2,\cdots,m_N>$. 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].

Although there has been a great progress on the design of multidisk files for facilitating partial match queries in the past years[1-29, 31-33]; however, in almost all the previously suggested design schemes, queries can be answered only in a sequential way: i.e., only one query can be answered at one time. Hence, the query performance won't be satisfied when, in real applications, more and more queries may occur at the same time for the same file. Accordingly, it is significant to concern the problem of concurrent control of answering multiple queries at the same time to increase query throughput per unit of time and further reduce average query response time for a multidisk file.

In [14], Lin and Chen investigated the above problem of designing multidisk files for facilitating concurrent control of answering partial match queries. Based upon the observation that different queries often have significantly different clustering requirements and the use of MMI(minimal marginal increase) record clustering technique and the DM(disk modulo) bucket allocation technique, they proposed a multidisk file design scheme. Although it has been pointed out that their proposed method does guarantee efficiently concurrent control of answering partial match queries of different types (meaning those queries using different query keys); unfortunately, the concurrency of answering partial match queries of the same type (meaning those queries that use the same key but with distinct key values) is not remarkable.

Accordingly, in this paper, we are further concerned the important research issue of designing a multidisk file to further facilitate concurrent control of answering partial match queries. Based upon the representation of an integer in a residue number system (RNS) [30], we shall present a new bucket allocation method, called the RNS allocation method. It is shown theoretically and experimentally that a multidisk file designed through the used of the RNS allocation method can guarantee various partial match queries of the same type to be answered concurrently.

2. The RNS allocation method

Let there be an N-attribute MKH file F=$< m_1, m_2, \cdots, m_N >$={ $[b_1, b_2, \cdots, b_N]$ | $0 \le b_i \le m_i - 1$ for $1 \le i \le N$ }, where $m_1, m_2, \cdots, m_N$ are pairwise relatively prime integers. Let NB=$\prod_{i=1}^{N} m_i$ denote the total number of buckets, and m≥2 denote the total number of available disks. Consider a bucket $[b_1, b_2, \cdots, b_N]$ in F. Since $m_1, m_2, \cdots, m_N$ are pairwise relatively prime and $0 \le b_i \le m_i$-1 for $1 \le i \le N$, $[b_1, b_2, \cdots, b_N]$ can be served as the unique RNS representation [30, Section 4.7] of some integer, denoted as $x_{b_1 b_2 \cdots b_N}$, in the range [0,NB] by using the set of moduli $\{m_1, m_2, \cdots, m_N\}$. That is $b_i \equiv x_{b_1 b_2 \cdots b_N}$ (mod m_i) for $1 \le i \le N$. And in this way all records in F are aligned into a linearly ordered list according to the order of their integer correspondences in [0,NB-1]. Consider, for illustration, a small case where N=3, m_1=3, m_2=5 and m_3=8. Table 2.1 depicts the one-to-one correspondence between each bucket $[b_1, b_2, b_3]$ in F and the corresponding integer $x_{b_1b_2b_3}$ in [0,119].

Table 2.1 here

It is interesting to observe, from Table 2.1, that the integer correspondences of all qualifying buckets of any PMQ for F show up periodically in the range [0,119]. For instance, consider a PMQ, q=(0,0,*). Then the buckets qualified by q consist of [0,0,0], [0,0,7], [0,0,6], [0,0,5], [0,0,4], [0,0,3], [0,0,2], [0,0,1]. The integer correspondences are 0, 15, 30, 45, 60, 75, 90 and 105, respectively, which show up periodically with period $p = 3\times5 = 15$. Similarly, the integer correspondences of all buckets qualified by the query $q' = (*,*,0)$ show up periodically with period $p' = 8$. The above periodicity is in fact true in general and can be formally stated and demonstrated as the following theorem.

Theorem 2.1

For any PMQ, the RNS integer correspondences of all qualifying buckets show up periodically.

Proof: Let $q_{i_1i_2\cdots i_n}$ be a PMQ for which the i_j-th key, $1\le j\le n$, are specified and other keys are unspecified. Suppose $[b_1, b_2, \cdots, b_N]$ and $[b'_1, b'_2, \cdots, b'_N]$ are two distinct buckets qualified by $q_{i_1i_2\cdots i_n}$. Then $b_{i_j} = b'_{i_j}$ for $1\le j\le n$, or equivalently, $x_{b_1b_2\cdots b_N} \equiv x_{b'_1b'_2\cdots b'_N}$ (mod m_{i_j}) for $1\le j\le n$. Since m_{i_1}, m_{i_2}, …, m_{i_n} are pairwise relatively prime, we have $x_{b_1b_2\cdots b_N} \equiv x_{b'_1b'_2\cdots b'_N}$ (mod $p_{i_1i_2\cdots i_n}$), where $p_{i_1i_2\cdots i_n} = \prod_{j=1}^{n} m_{i_j}$. This says that the integer correspondences of the bucket s qualified by $q_{i_1i_2\cdots i_n}$ show up periodically in the range [0,NB-1] with period $p_{i_1i_2\cdots i_n}$.

Q.E.D.

Another intriguing observation from Table 2.1 is that the components of two consecutive buckets $[b_1, b_2, b_3]$ and $[b'_1, b'_2, b'_3]$ for which $x_{b'_1b'_2b'_3} = x_{b_1b_2b_3} + 1$ satisfy $b'_i \equiv b_i + 1$ (mod m_i) for $1\le i\le 3$. This property is also true in general, we formally state and prove it as the following theorem.

Theorem 2.2

Let $(b_1, b_2, \cdots b_N)$ and $(b'_1, b'_2, \cdots b'_N)$ be two records such that $x_{b'_1 b'_2 \cdots b'_N} = x_{b_1 b_2 \cdots b_N} + 1$. Then $b'_i \equiv b_i + 1 \pmod{m_i}$ for $1 \le i \le N$.

Proof: Since $b'_i \equiv x_{b'_1 b'_2 \cdots b'_N} \pmod{m_i}$ and $b_i \equiv x_{b_1 b_2 \cdots b_N} \pmod{m_i}$ for $1 \le i \le N$, $x_{b'_1 b'_2 \cdots b'_N} = x_{b_1 b_2 \cdots b_N} + 1$ implies that $b'_i \equiv x_{b'_1 b'_2 \cdots b'_N} \equiv x_{b_1 b_2 \cdots b_N} + 1 \equiv b_i + 1 \pmod{m_i}$ for $1 \le i \le N$.

Q.E.D.

Based upon the correspondence properties stated in Theorem 2.1 and Theorem 2.2, in the following we shall propose a new bucket allocation scheme, called the RNS bucket allocation scheme.

Algorithm 2.1 The RNS Bucket Allocation Scheme

Input: An MKH file F=$\{[b_1, b_2, \cdots, b_N] \mid 0 \le b_i \le m_i - 1$ for $1 \le i \le N\}$, where $m_1, m_2, \cdots, m_N$ are pairwise relatively prime; and m disks, m$\ge$2.

Output: The allocation of the bucket s in F onto m disks.

Steps: 1. Assign each bucket $[b_1, b_2, \cdots, b_N]$ in F to disk $x_{b_1 b_2 \cdots b_N} \bmod m$

(2.1)

Consider, for instance, the case for which m=6. Table 2.2 depicts the assignment of all buckets in F=<3, 5, 8> among 6 disks by using the RNS allocation scheme.

(Table 2.2 here)

.

3. Concurrent answering property of the RNS allocation method

Let F be an MKH file with N attributes $A_1, A_2, \cdots, A_N$. Let $Q = \{A_{i_1}, A_{i_2}, \cdots, A_{i_n} \mid 1 \le i_1 < i_2 < \cdots < i_n \le N\} \subseteq \{A_1, A_2, \cdots, A_N\}$. We say that a PMQ is of type Q, denoted as q_Q or $q_{i_1, i_2, \cdots, i_n}$, if the set of attributes specified in the query is equal to Q. Accordingly, two PMQs are said to be of the same type if the set of attributes specified, respectively, in the two queries is identical. Otherwise, they are said to be of different types. In this section, we shall show that an MKH file designed through the use of the RNS allocation method does guarantee various PMQs of the same type to be answered concurrently. Without loss of generality, we assume that $Q=\{A_{i_1}, A_{i_2}, \cdots, A_{i_n}\} = \{A_1, A_2, \cdots, A_n\}$.

Theorem 3.1

Let $F=<m_1, m_2, \cdots, m_N>$ and $Q=\{A_1, A_2, \cdots, A_n\}$ where $n \leq N$. Let m be the number of available disks and $P=\prod_{j=1}^{n} m_j$. Let $q=(a_1, a_2, \cdots, a_n, *, *, \cdots, *)$ be a PMQ of type Q and D(q) denote the set of disks where the buckets qualified by q reside under the use of the RNS allocation method. Suppose (m, P)=d, m=ad and P=bd where (a,b)=1.

(1) If $a_1 = a_2 = \cdots = a_n = 0$ then D(q) $\subseteq$ {0, d, 2d, $\cdots$, (a-1)d}. The equality holds when $a \leq \prod_{j=n+1}^{N} m_j$.

(2) In general, D(q) $\subseteq$ {r+d, r+2d, $\cdots$, r+(a-1)d}, where $r = x_{a_1 a_2 \cdots a_n 00 \cdots 0}$ mod d and $x_{a_1 a_2 \cdots a_n 00 \cdots 0}$ is the integer correspondence of the bucket $[a_1, a_2, \cdots, a_n, 0, 0, \cdots, 0]$.

Proof: (1) Since $x_{0 \cdots 00 \cdots 0}=0$, the set of integer correspondences of all buckets qualified by q is $\{kP \mid 0 \leq k < \prod_{j=n+1}^{N} m_j\}$. Accordingly, we have $D(q)=\{kP \bmod m \mid 0 \leq k < \prod_{j=n+1}^{N} m_j\}$.

Since m=ad, P=bd and (a,b)=1, we have $\{kb \bmod a \mid 0 \leq k < \prod_{j=n+1}^{N} m_j\} \subseteq \{0, 1, \cdots, a-1\}$, where the equality holds when $a \leq \prod_{j=n+1}^{N} m_j$. This implies that $D(q)=\{kP \bmod m \mid 0 \leq k < \prod_{j=n+1}^{N} m_j\}=\{kbd \bmod ad \mid 0 \leq k < \prod_{j=n+1}^{N} m_j\} \subseteq \{0, d, \cdots, (a-1)d\}$ and the equality holds when $a \leq \prod_{j=n+1}^{N} m_j$.

(2) Let $x_{a_1 a_2 \cdots a_n 00 \cdots 0}$=cd+r where $0 \leq r < d$. That is, $r = x_{a_1 a_2 \cdots a_n 00 \cdots 0}$ mod d. Then we have, in general, that $D(q)=\{(x_{a_1 a_2 \cdots a_n 00 \cdots 0} + kP) \bmod m \mid 0 \leq k < \prod_{j=n+1}^{N} m_j\} \subseteq \{((cd+r)+jd) \bmod m \mid 0 \leq j < a\}=\{r+cd+jd \bmod ad \mid 0 \leq j < a\}=\{r+jd \mid 0 \leq j < a\}=\{r+d, r+2d, \cdots, r+(a-1)d\}$, where the equality holds when $a \leq \prod_{j=n+1}^{N} m_j$.

Q.E.D.

Since for $0 \le r_1, r_2 < d$ and $r_1 \ne r_2$, $\{r_1+d, r_1+2d, \cdots, r_1+(a-1)d\}$ and $\{r_2+d, r_2+2d, \cdots, r_2+(a-1)d\}$ are disjoint. Consequently, we have the following corollaries.

Corollary 3.1

Let q=$(a_1, a_2, \cdots, a_n, *, *, \cdots, *)$ and q'=$(a_1', a_2', \cdots, a_n', *, *, \cdots, *)$ be two PMQs of the same type Q. Suppose $x_{a_1a_2\cdots a_n00\cdots0}$ mod d $= r_1$ and $x_{a_1'a_2'\cdots a_n'00\cdots0}$ mod d $= r_2$.

(1) If $r_1 \ne r_2$ then q and q' can be answered concurrently.
(2) The maximum degree of concurrent answering queries of type Q is d.
(3) If d=1 then any two queries of type Q can' t be answered concurrently.

Since $\{ kP \bmod m \mid \mathbf{0} \le k < m \}$ is no more than a permutation of $\{0, 1, \cdots, m\text{-}1\}$ if (m, P)=d=1, we have the following conclusion.

Corollary 3.2

If (m, P)=d=1, then all qualifying buckets of each PMQ of type Q are distributed uniformly among m disks. That is, the RNS allocation method id strictly optimal for each PMQ of type Q.

Example 3.1

Consider F=<3, 5, 8>, m=6 and the RNS allocation as shown in Table 2.2. Let q=(1, *, *) and q'=(2, *, *). Then P=3, d=(m, P)=3, x_{100}=40, x_{200}=80. Since $r_1 = x_{100}$ mod d=1, $r_2 = x_{200}$ mod d=2, and $r_1 \ne r_2$, q and q' can be answered concurrently. Similarly, let $\overline{q}$=(*, 2, 4) and $\overline{q}'$=(*, 2, 7). Then P=5×8=40, d=(m, P)=2, x_{024}=12 and x_{027}=87. Since $r_1 = x_{024}$ mod d=0, $r_2 = x_{027}$ mod d=1, and $r_1 \ne r_2$, $\overline{q}$ and $\overline{q}'$ can also be answered concurrently. On the other hand, let $\hat{q}$=(*, 1, *) and $\hat{q}'$=(*, 4, *). Then P=5 and d=(m, P)=1. Thus the qualifying buckets of $\hat{q}$ and $\hat{q}'$, respectively, are distributed uniformly among m=6 disks. Consequently, they can' t be answered concurrently.

Corollary 3.1 and 3.2 can be further illustrated by Table 3.1 below which depicts, for each query type, all queries of the same type and the set of disks where the qualifying buckets of each query reside.

(Table 3.1 here)

.

4. Conclusions

In this report, we have concerned the important research issue of designing a multidisk file to facilitate concurrent answering of PMQs. Based upon the RNS (residue number system), we have first presented a new bucket allocation method, called the RNS allocation method. It has been shown that a multidisk MKH file obtained through the use of the RNS allocation method does guarantee various PMQs of the same type to be answered concurrently.

By combining the results of this paper with that of [14], we can obtain a multidisk MKH file which allows concurrent answering of various PMQs, either of the same type or different types. This will significantly increase the query throughput per unit of time and further reduce the average response time over all possible PMQs as well.

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] H. F. Lin and C. Y. Chen, "Concurrent Control of Partial Match Queries for Multidisk MKH Files," *Proc. ICS2000*, Chiayi, Taiwan, Dec. 2000.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

[22] 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.

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

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

[25] H. F. Lin and C. Y. Chen,”An RNS Based Perfectly Optimal Data Allocation and Declustering Scheme,” *in submission to JISE*.

[26] 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.

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

[28] 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.

[29] R. L. Rivest, “Partial-Match Retrieval Algorithms,” *SIAM J. Computing*, vol. 14, no. 1, pp. 19-50, 1976.

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

[31] J. B. Rothnie and T. Rozano, “Attribute Based File Origanization in a paged Memory Environment,” *CACM*, vol. 17, no. 2, pp. 63-69, 1974.

[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.

Table 2.1 The 1-1 Correspondence between Integers in [0,119] and Buckets in F

Integer $x_{s_1s_2s_3}$	Bucket $[s_1,s_2,s_3]$	Integer $x_{s_1s_2s_3}$	Bucket $[s_1,s_2,s_3]$	Integer $x_{s_1s_2s_3}$	Bucket $[s_1,s_2,s_3]$
0	[0,0,0]	40	[1,0,0]	80	[2,0,0]
1	[1,1,1]	41	[2,1,1]	81	[0,1,1]
2	[2,2,2]	42	[0,2,2]	82	[1,2,2]
3	[0,3,3]	43	[1,3,3]	83	[2,3,3]
4	[1,4,4]	44	[2,4,4]	84	[0,4,4]
5	[2,0,5]	45	[0,0,5]	85	[1,0,5]
6	[0,1,6]	46	[1,1,6]	86	[2,1,6]
7	[1,2,7]	47	[2,2,7]	87	[0,2,7]
8	[2,3,0]	48	[0,3,0]	88	[1,3,0]
9	[0,4,1]	49	[1,4,1]	89	[2,4,1]
10	[1,0,2]	50	[2,0,2]	90	[0,0,2]
11	[2,1,3]	51	[0,1,3]	91	[1,1,3]
12	[0,2,4]	52	[1,2,4]	92	[2,2,4]
13	[1,3,5]	53	[2,3,5]	93	[0,3,5]
14	[2,4,6]	54	[0,4,6]	94	[1,4,6]
15	[0,0,7]	55	[1,0,7]	95	[2,0,7]
16	[1,1,0]	56	[2,1,0]	96	[0,1,0]
17	[2,2,1]	57	[0,2,1]	97	[1,2,1]
18	[0,3,2]	58	[1,3,2]	98	[2,3,2]
19	[1,4,3]	59	[2,4,3]	99	[0,4,3]
20	[2,0,4]	60	[0,0,4]	100	[1,0,4]
21	[2,1,5]	61	[1,1,5]	101	[2,1,5]
22	[1,2,6]	62	[2,2,6]	102	[0,2,6]
23	[2,3,7]	63	[0,3,7]	103	[1,3,7]
24	[0,4,0]	64	[1,4,0]	104	[2,4,0]
25	[1,0,1]	65	[2,0,1]	105	[0,0,1]
26	[2,1,2]	66	[0,1,2]	106	[1,1,2]
27	[0,2,3]	67	[1,2,3]	107	[2,2,3]
28	[1,3,4]	68	[2,3,4]	108	[0,3,4]
29	[2,4,5]	69	[0,4,5]	109	[1,4,5]
30	[0,0,6]	70	[1,0,6]	110	[2,0,6]
31	[1,1,7]	71	[2,1,7]	111	[0,1,7]
32	[2,2,0]	72	[0,2,0]	112	[1,2,0]
33	[0,3,1]	73	[1,3,1]	113	[2,3,1]
34	[1,4,2]	74	[2,4,2]	114	[0,4,2]
35	[2,0,3]	75	[0,0,3]	115	[1,0,3]
36	[0,1,4]	76	[1,1,4]	116	[2,1,4]
37	[1,2,5]	77	[2,2,5]	117	[0,2,5]
38	[2,3,6]	78	[0,3,6]	118	[1,3,6]
39	[0,4,7]	79	[1,4,7]	119	[2,4,7]

Table 2.2 The Allocation of 120 Buckets among 6 Disks
by Using the RNS Allocation Scheme

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
[0,0,0]	[1,1,1]	[2,2,2]	[0,3,3]	[1,4,4]	[2,0,5]
[0,1,6]	[1,2,7]	[2,3,0]	[0,4,1]	[1,0,2]	[2,1,3]
[0,2,4]	[1,3,5]	[2,4,6]	[0,0,7]	[1,1,0]	[2,2,1]
[0,3,2]	[1,4,3]	[2,0,4]	[0,1,5]	[1,2,6]	[2,3,7]
[0,4,0]	[1,0,1]	[2,1,2]	[0,2,3]	[1,3,4]	[2,4,5]
[0,0,6]	[1,1,7]	[2,2,0]	[0,3,1]	[1,4,2]	[2,0,3]
[0,1,4]	[1,2,5]	[2,3,6]	[0,4,7]	[1,0,0]	[2,1,1]
[0,2,2]	[1,3,3]	[2,4,4]	[0,0,5]	[1,1,6]	[2,2,7]
[0,3,0]	[1,4,1]	[2,0,2]	[0,1,3]	[1,2,4]	[2,3,5]
[0,4,6]	[1,0,7]	[2,1,0]	[0,2,1]	[1,3,2]	[2,4,3]
[0,0,4]	[1,1,5]	[2,2,6]	[0,3,7]	[1,4,0]	[2,0,1]
[0,1,2]	[1,2,3]	[2,3,4]	[0,4,5]	[1,0,6]	[2,1,7]
[0,2,0]	[1,3,1]	[2,4,2]	[0,0,3]	[1,1,4]	[2,2,5]
[0,3,6]	[1,4,7]	[2,0,0]	[0,1,1]	[1,2,2]	[2,3,3]
[0,4,4]	[1,0,5]	[2,1,6]	[0,2,7]	[1,3,0]	[2,4,1]
[0,0,2]	[1,1,3]	[2,2,4]	[0,3,5]	[1,4,6]	[2,0,7]
[0,1,0]	[1,2,1]	[2,3,2]	[0,4,3]	[1,0,4]	[2,1,5]
[0,2,6]	[1,3,7]	[2,4,0]	[0,0,1]	[1,1,2]	[2,2,3]
[0,3,4]	[1,4,5]	[2,0,6]	[0,1,7]	[1,2,0]	[2,3,1]
[0,4,2]	[1,0,3]	[2,1,4]	[0,2,5]	[1,3,6]	[2,4,7]

Table 3.1The Set of Disks Where All Qualifying Buckets of Each Query Reside

Query type	**Query**	**Stored disks**
$(A_1,*,*)$	$(0,*,*)$	D_0 , D_3
	$(1,*,*)$	D_1 , D_4
	$(2,*,*)$	D_2 , D_5
$(*, A_2,*)$	$(*, a_2,*)$ $a_2 =$**0,1,2,3,4**	D_0 , D_1 , D_2 , D_3 , D_4 , D_5
$(*,*, A_3)$	$(*,*,a_3)$ $a_3 =$**0,2,4,6**	D_0 , D_2 , D_4
	$(*,*,a_3)$ $a_3 =$**1,3,5,7**	D_1 , D_3 , D_5
$(A_1, A_2,*)$	$(0, a_2,*)$ $a_2 =$**0,1,2,3,4**	D_0 , D_3
	$(1, a_2,*)$ $a_2 =$**0,1,2,3,4**	D_1 , D_4
	$(2, a_2,*)$ $a_2 =$**0,1,2,3,4**	D_2 , D_5
$(A_1,*, A_3)$	$(0,*,a_3)$ $a_3 =$**0,2,4,6**	D_0
	$(0,*,a_3)$ $a_3 =$**1,3,5,7**	D_3
	$(1,*, a_3)$ $a_3 =$**0,2,4,6**	D_4
	$(1,*, a_3)$ $a_3 =$**1,3,5,7**	D_1
	$(2,*, a_3)$ $a_3 =$**0,2,4,6**	D_2
	$(2,*, a_3)$ $a_3 =$**1,3,5,7**	D_5
$(*, A_2, A_3)$	$(*, a_2, a_3)$ $a_2 =$**0,1,2,3,4** $a_3 =$**0,2,4,6**	D_0 , D_2 , D_4
	$(*, a_2, a_3)$ $a_2 =$**0,1,2,3,4** $a_3 =$**1,3,5,7**	D_1 , D_3 , D_5