MULTIDIMENSIONAL DYNAMIC QUANTILE HASHING IS VERY EFFICIENT FOR NON-UNIFORM RECORD DISTRIBUTIONS

Hans-Peter Kriegel* - Bernhard Seeger**

* Lehrstuhl fuer Informatik I, Universitaet Wuerzburg, D-8700 Wuerzburg, West Germany
** Institut fuer Informatik II, Universitaet Karlsruhe, D-7500 Karlsruhe, West Germany

Abstract:
Previous multidimensional dynamic hashing schemes exhibit two obvious shortcomings. First, even for uniform record distribution, the retrieval performance of these schemes suffers from several disadvantages. In a recent paper we have suggested a multidimensional dynamic hashing scheme which exhibits better retrieval performance than its competitors for uniform distribution. The even more severe second disadvantage of all known multidimensional dynamic hashing schemes is the very poor performance for nonuniform record distributions. In this paper we present the quantile method as a scheme which exhibits for nonuniform distributions practically the same performance as for uniform distributions. This is underlined by experimental runs with an implementation of our scheme. In addition to its excellent performance, our scheme fulfills all the necessary requirements to be used in an engineering database system: it is dynamic, is suitable for secondary storage devices, supports point data and spatial data objects and supports spatial clustering (proximity queries).

1. Introduction

Concerning database systems for standard applications, e.g. commercial applications, there is a large variety of index structures at the disposal of the database designer for the implementation of the physical level of a database system. As demonstrated in [Kri 84] there are efficient tree-based index structures such as multidimensional B-trees, in particular kB-trees.

For non-standard databases, also called engineering databases, used in applications such as image processing, design and manufacturing (CAD/CAM) etc. none of the tree-based index structures is suitable, since they cluster records according to the lexicographical ordering. When designed suitably, hash-based index structures cluster records which are close together in the key space. This is the type of clustering which we need in engineering databases. Furthermore, multidimensional hashing schemes can be designed to fulfill all the requirements for use in engineering database systems: to be dynamic, to be suitable for secondary storage devices and to support point and spatial objects.

However, multidimensional hashing schemes suffer from two shortcomings. In most engineering applications, objects are nonuniformly distributed in space. First, there is no multidimensional hashing scheme which exhibits practically the same retrieval performance for nonuniform distribution as for uniform distribution. Second, even for uniform distribution, the retrieval performance of all known multidimensional hashing schemes suffers either from a superlinearly growing directory or from an uneven distribution of the records over the pages of a file.

Thus we proceed in two steps. In step 1 we have designed a multidimensional dynamic hashing scheme and we have shown that it performs better than its competitors for uniform distribution. This scheme is suitably composed from the following ingredients: linear hashing [Lit 80], order preservation using bitcutoff functions and partial expansions for improving performance [Lar 80]. Therefore we call it multidimensional order preserving linear hashing with partial expansions (MOLHPE). For a description of this scheme we refer to [KS 86].

In this paper now we perform step 2 by adding on top of our scheme the quantile method which guarantees that our scheme performs for a nonuniform distribution practically as well as for a uniform distribution. As mentioned before, in most non-standard applications, objects are nonuniformly distributed in space. Recently, the following variants either of the grid file [NHS 84] or of multidimensional extendible hashing [Oto 84] have been suggested all claiming a graceful adaption of their scheme to the key distribution: these variants include the 2-level grid file [Hin 85], the multilevel grid file [KW 85], the interpolation based grid file [Ouk 85] and the balanced multidimensional extendible hash tree [Oto 86]. However, none of these structures adjusts so gracefully to the distribution that it exhibits practically the same retrieval performance for nonuniform distribution as for uniform distribution. This is due to the fact that the partitioning process in none of these structures adapts well enough to the underlying distribution.

CH2407-5/87/0000/0010$01.00 ©1987 IEEE

In the following we consider a file of d-attribute composite keys $K = (k_1,..,k_d)$. We assume the domain of the keys to be the unit d-dimensional cube $[0, 1)^d$. Obviously, this requirement can easily be fulfilled for an arbitrary domain by simple transformation.

2. The quantile method

In this section we will present the quantile method which is applicable to any multidimensional hashing scheme with or without directory. Obviously, we will apply the quantile method to MOLHPE, since the quantile method is supposed to yield practically the same retrieval performance for nonuniform distributions as for uniform distributions and MOLHPE performs better than its competitors for uniform distributions. Thus we expect practically optimal behaviour of the quantile method applied to MOLHPE for nonuniform distributions. All we have to know about MOLHPE within the context of this paper is the fact that it is an order preserving generalization of linear hashing with partial expansions to the multidimensional case. Thus MOLHPE uses no directory, allows overflow records which are organized in separate chains and guarantees that the relative load factors of all chains are between 0.5 and 1 (condition (I)). Here, the relative load factor is defined as follows:

Let r be the record to be inserted next into the file and $C_1,...,C_m$, $m \geq 1$, be a disjunct partition of the file into chains. Then the *relative load factor* $lf(C_i)$ of chain C_i, $1 \leq i \leq m$, is given by

$$lf(C_i) = \frac{Pb(r \in C_i)}{\max_{1 \leq j \leq m} Pb(r \in C_j)}$$

where $Pb(r \in C_i)$ denotes the probability with which record r is inserted into chain C_i.

Even for nonuniform distributions we want to fulfill condition (I). We will achieve this by selecting the partitioning points depending on the distribution of objects.
Obviously, the choice of the partitioning points of the j-th dimension should only depend on the distribution function of the j-th attribute. Thus condition (I) can only be fulfilled if the records follow an independent distribution. In case this independence assumption is not fulfilled the method which we will suggest will still be very efficient but not practically optimal any more. Since in most applications the distribution of objects is not known in advance, we will approximate the unknown distribution using the records presently in the file.

Burkhard was the first to use a stochastic approximation process for adapting the partitioning points to the underlying distribution [Bur 84]. We will call this process in the following quantile method. In [Bur 84] the quantile method is applied to the 1-dimensional interpolation-based index maintenance scheme [Bur 83]. Algorithms for split and reorganization of the file are only roughly sketched.

We will apply the quantile method to multidimensional order preserving linear hashing with partial expansions and we will present a rather detailed description of the algorithm for reorganization. Furthermore, we will report on experiments run with an implementation of our scheme comparing MOLHPE with and without quantile method for different nonuniform distributions.

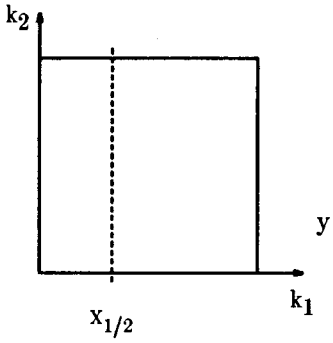
Let us demonstrate our method considering 2-attribute composite keys $K = (k_1, k_2)$ and a 2-dimensional distribution function F with marginal distribution functions f_1 and f_2. We assume that the stochastic variables k_1 and k_2 are independent, i.e.

$$F(k_1, k_2) = f_1(k_1) * f_2(k_2)$$

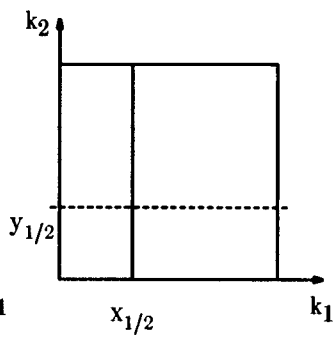
For $\alpha \in [0, 1]$ the α-quantile of the 1st (2nd) attribute is the domain value x_α (y_α) such that

$$f_1(x_\alpha) = \alpha \qquad (f_2(y_\alpha) = \alpha)$$

Now let us assume that starting from an empty file we have to partition the key space $[0,1)^2$ for the first time and we decide to partition the first dimension (axis). If f_1 is a non-uniform distribution function, we will not partition the first dimension in the middle, but we will choose the 1/2-quantile as the partitioning point. This guarantees that a new record will be stored with equal probability in the page corresponding to the rectangle $[0 , x_{1/2}) \times [0 , 1)$ or in the page corresponding to the rectangle $[x_{1/2} , 1) \times [0 , 1)$, see figure 2.1. During the next expansion we will partition the second dimension (axis) and we will choose the partitioning point $y_{1/2}$, see figure 2.2.



Figur 2.1



Figur 2.2

Figure 2.3 shows the file consisting of 16 pages, where each axis has been partitioned at the 1/4, 1/2 and 3/4 quantile. As depicted in figure 2.3, the partitioning points are stored in binary trees which can easily be stored in main memory. The *most important property* of these binary trees of partitioning points is the following: for each type of operation a nonuniformly distributed query value k_j, $j = 1,2$, is transformed into a uniformly distributed $\alpha \in [0, 1]$ by searching the corresponding binary tree of partitioning points. This uniformly distributed $\alpha \in [0, 1]$ is then used

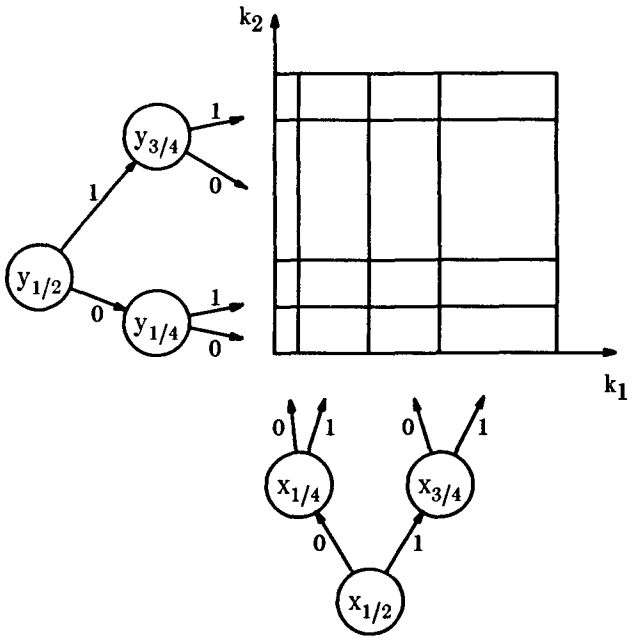


Figure 2.3

as an input to the retrieval and update algorithms of MOLHPE.

Now we will present a more formal description of the quantile method. Let F denote the d-dimensional distribution function of the d-attribute composite keys. Then f_i, $1 \le i \le d$, denotes the 1-dimensional distribution function of the marginal distribution. For $0 \le \alpha \le 1$ and $i \in \{1,...,d\}$ the α - *quantile* of the i-th attribute is the domain value $x(\alpha)$ of the i-th attribute such that

$$f_i(x(\alpha)) = \alpha.$$

In [RM 55] the quantile method has been described for the first time. A survey on the theory of quantile mehtods can be found in [KC 78]. For each f_i, $1 \le i \le d$ we will approximate the proper quantiles according to the following iterative scheme.

Let $x(\alpha)$ be the α-quantile of one of the 1-dimensional distribution functions f_i, $1 \le i \le d$. Furthermore let $x_1(\alpha)$ be our best initial guess for $x(\alpha)$. Let $\{k_i^j\}$ be the sequence of i-th component keys which have been inserted into the file. Then we define for $n \ge l$

(i) $m(n,l) = | \{ k_i^j \mid x_l(\alpha) \ge k_i^j, 1 \le j \le n \} |$

where the sequence $\{x_l(\alpha)\}$ of estimates is given by

(ii) $x_{l+1}(\alpha) = x_l(\alpha) - a_l * (\frac{m(n,l)}{n} - \alpha)$

Here $\{a_l\}$ is a sequence with

(a) $\sum_{l=1}^{k} a_l \to \infty$ for $k \to \infty$

(b) $\sum_{l=1}^{k} a_l^2$ converges for $k \to \infty$

Obviously, $m(n,l)$ denotes the number of records whose i-th component key is below the l-th estimate $x_l(\alpha)$ of the α-quantile when n records are presently in the file. The expected value of $m(n,l)/n$ corresponds to $f_i(x_l(\alpha))$ According to (ii) the $(l+1)$st estimate of $x(\alpha)$ can be computed only after at least l records are stored in the file.

The sequence $\{a_l\} = \{1/l\}$, $l \ge 1$, fulfills conditions (a) and (b). However, in order to guarantee a fast adaption to the distribution we have to choose a_l depending on the file size. In our implementation where d=2, we choose $a_l = 1/2^{(L \text{ div } d)}$, where L is the present level of the file. As $l \to \infty$ the sequence $\{ x_l(\alpha) \}$ converges to $x(\alpha)$ with probability 1, if f_i is continuous almost everywhere.

3. Application of the quantile method to multidimensional order preserving linear hashing with partial expansions

As in [KS 86], we say that the file is on level L, where $L = \sum_{i=1}^{d} L_i$, to indicate that the file has doubled in size L times. Now the i-th axis is partitioned by 2^{L_i}-1 partitioning points $pp(\alpha)$, $0 < \alpha < 1$. Each of the $pp(\alpha)$ is the newest estimate of the α-quantile, computed using the quantile method. The set of points is given by

$$P_i = \{ pp(\alpha) \mid \alpha = \sum_{j=1}^{L_i} b_j * 2^{-j}, b_j \in \{0,1\} \}$$

Obviously, each $pp(\alpha)$ can be characterized by a bitstring. Using this characterization we can associate to each partitioning point exactly one node in a binary search tree in the obvious way. For computing a new estimate of $pp(\alpha)$ we need the number of records whose i-th component key is below $pp(\alpha)$. Therefore, in each node of the binary tree, we store in addition to $pp(\alpha)$ the number of records whose i-th component key is in the interval $[pp(\alpha'),pp(\alpha)]$, where α and α' are given by

$$\alpha = \sum_{j=1}^{z} b_j * 2^{-j} \quad \text{where } z \le L_i \text{ and } b_z = 1$$

$$\alpha' = \sum_{j=1}^{z-1} b_j * 2^{-j}$$

The *most important property* of these binary trees of partitioning points is the following: for each type of operation and each attribute A_j, $1 \le j \le d$, a nonuniformly distributed query value $x_j \in$ domain (A_j) is transformed into a uniformly distributed $\alpha \in [0,1]$ by searching the corresponding binary tree of partitioning points. This uniformly distributed $\alpha \in [0,1]$ is then used as an input to the retrieval and update algorithms of MOLHPE.

Now for each axis j, $1 \le j \le d$, we organize the partitioning points within a binary tree which may be stored explicitly (using pointers) or implicitly (stored within an array

using no pointers). These d binary trees have a total storage requirement of $0(d/b * n^{1/d})$ where n is the number of records in the file and b is the capacity of a data page. The storage requirement of the d binary trees is comparable to that of the linear scales in the grid file. Thus these binary trees can easily be kept in central memory. The algorithms for retrieval and expansion are basically the same as for MOLPHE without quantile approximation. Therefore, they are not presented here. Let us only mention, that the file is cyclically expanded or contradicted depending on the number of records stored in the file. Instead, we will treat the important problem of reorganization extensively.

4. Reorganization of the file

None of the previous multidimensional dynamic hashing schemes gracefully adapts to the distribution of the records in key space. Using the suggested quantile method, we *reorganize the file* by adapting the partition of the key space to the present distribution. This is done incrementally by computing a new estimate for a partitioning point $pp(\alpha)$ using (ii) and adjusting the partition of the key space to the new partitioning hyperplane. We say that we move a partitioning hyperplane. We call the rule which determines whether the file will be reorganized or not the *control function for the reorganization of the file.* We will consider the following possible control functions:

(C1) The file is reorganized before each expansion (or contraction).

(C2) Let α be the quantile to be reorganized next and $pp(\alpha)$ be the present estimate of the α-quantile. Then we compute the confidence interval I for α. If $\alpha \in I$, then $pp(\alpha)$ is accepted as α-quantile. Otherwise a new estimate is computed for $pp(\alpha)$ according to equation (ii), and the file is reorganized by moving the $pp(\alpha)$-hyperplane. Then α is advanced to the α-value which will be reorganized next. This step is performed after a predefined number of insertions and deletions.

(C3) Let $i_0 \in \{1,...d\}$ be the axis to be considered next for reorganization. Compute the number of records with i_0-th component key between two neighboring partitioning points $pp(\alpha_L)$,and $pp(\alpha_R)$, $pp(\alpha_L) < pp(\alpha_R)$. Substract $(\alpha_R - \alpha_L)*n$ which is the corresponding expected value. Take the maximum of this difference over all possible α_R's on axis i_0. Compute a new estimate for this $pp(\alpha_R)$ and advance the axis to be reorganized next cyclically.

Control function (C1) has the following property if we assume that the expansion of the file is controlled by storage utilization. The partition of the key space remains invariant if after an expansion of the file continuously a record is inserted and a different record is deleted. Thus the distribution of records in key space may change without adapting the partition of the key space.

Such undesirable behaviour is prevented by control function (C2). Furthermore (C2) exhibits the following desirable property: before computing a new estimate for a quantile approximation we can decide whether the old estimate already suffices.

Control function (C3) determines the worst estimate over each axis. Thus it requires computation for all partitioning points and searching for the maximum difference of present estimate and expected value. Obviously, this needs intensive computation. Summarizing we can say that (C2) and (C3) are adequate control functions for the reorganization of the file.

As mentioned before, during reorganization one partitioning hyperplane is moved. If this move is done in one step, it requires $0(n^{1-1/d})$ pages accesses for performing the reorganization. Although this may sound very high, the same number of page accesses is required by the grid file when adding a new partitioning hyperplane to the grid directory. If moving hyperplanes is done in one step, the quantile method (and the same is true for the grid file) looses its dynamic character. Since reorganization only improves retrieval performance and is not as crucial as restructuring in the grid file, we will amortize the time required for moving one partitioning hyperplane over a sequence of insert and delete operations. Thus, step by step, we will adjust a pair of chains separated by the hyperplane to be moved to the new estimate. We call a reorganization *local*, if the expected value for the number of page accesses during reorganization is constant. The corresponding control function for reorganization of the file is called *linear*. Obviously, local reorganization is only possible for hashing schemes without directory. Similar as for the expansion of a file, we define the following variables: ra $\in\{1,...d\}$ denotes the axis (dimension) whose partitioning points will be recomputed next. We call this axis reorganization axis.

The pair of chains to be reorganized next is specified by d reorganization pointers $rp_1,...,rp_d$. Now the chain $G(rp_1,...,rp_d)$ and its right neighbor chain with respect to axis ra will be adjusted during the next local reorganization. The address $G(rp_1,..,rp_d)$ is computed in two steps. For every j, $1 \leq j \leq d$, rp_j is computed using an one dimensional order preserving hashing function. Then the d-dimensional index $(rp_1,..,rp_d)$ and the address function G [KS 86] is used to compute the address.

After inserting a record we will either expand the file or perform a local reorganization of the file. More presisely, after each insertion which does not require an expansion, a local reorganization will be carried out. This strategy is supported by the following observation obtained from experimental runs on our implementation: For most nonuniform distributions the expected value of page accesses for insertion is considerably lower with quantile method than without quantile method. Since this phenomenon is mostly unexpected, it justifies our strategy even more.

We will now present an algorithm for local reorganization using control function (C2):

Algorithm for local reorganization

1. For chain Next = $G(rp_1,...,rp_d)$ compute the right partitioning point $pp(\alpha_R)$ on axis ra;
2. IF $rp_i = 0$ for all $i \in \{1,...,d\}$ - {ra} THEN
   compute a confidence interval I for α_R and $pp(\alpha_R)$;
   IF $\alpha_R \epsilon I$ THEN
   $rp_{ra} := rp_{ra}+1$;
   IF $rp_{ra} = 2^{L_{ra}}-1$ THEN
   ra := ra MOD d + 1
   END;
   $rp_i := 0$ for all $i \in \{1,...,d\}$
   ELSE
   compute a new estimate for $pp(\alpha_R)$ using equation (ii)
   END;
3. Adjust chain Next and the right neighbor chain with respect to axis ra to the new estimate for $pp(\alpha_R)$.
4. Advance $(rp_1,...,rp_d)$ to the next value.

Step 3 of the algorithm physically performs the local reorganization. Steps 3 and 4 are illustrated using the following example.

Example:
Let us consider the situation in Figure 2.3 and let us assume that the estimates of the quantiles are stored in the nodes of the binary trees. Thus we have the following parameters: d=2, L_1=2, L_2=2, L=4. Furthermore we consider rp=(0,1) and ra=2 and assume that the file will not be expanded during the following four insertions. This implies that the reorganization algorithm is executed after each of these insertions.

An insertion of a record triggers a computation of a new estimate $y_{1/2}$ of the 1/2-quantile with respect to the second dimension (Fig. 4.1). Then we adjust the page Next = G(rp) to the new estimate by removing all corresponding records from the upper neighbor page. We advance the reorganization pointer to the next page which is still limited by the old estimate $y_{1/2}$ (Fig. 4.2). After two more insertions we have adjusted the partition of two more pairs of pages to the new estimate (Fig 4.3,4.4). Eventually, after one more insertion, we adjust the last pair of pages and reorganize the binary tree of the second dimension (Fig. 4.5).

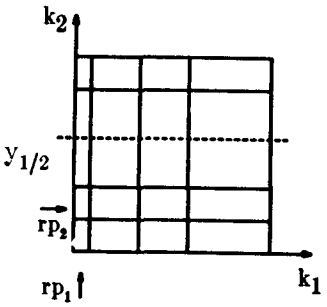


Figure 4.1

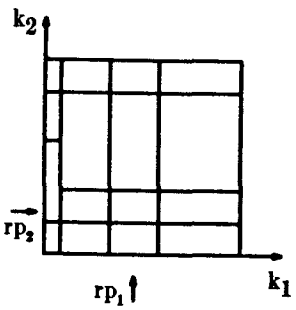


Figure 4.2

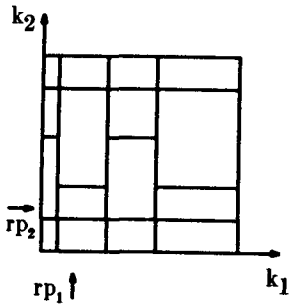


Figure 4.3

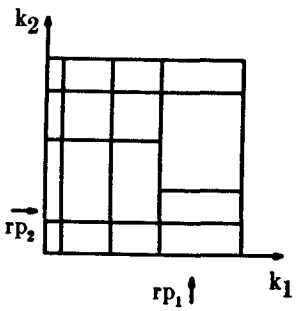


Figure 4.4

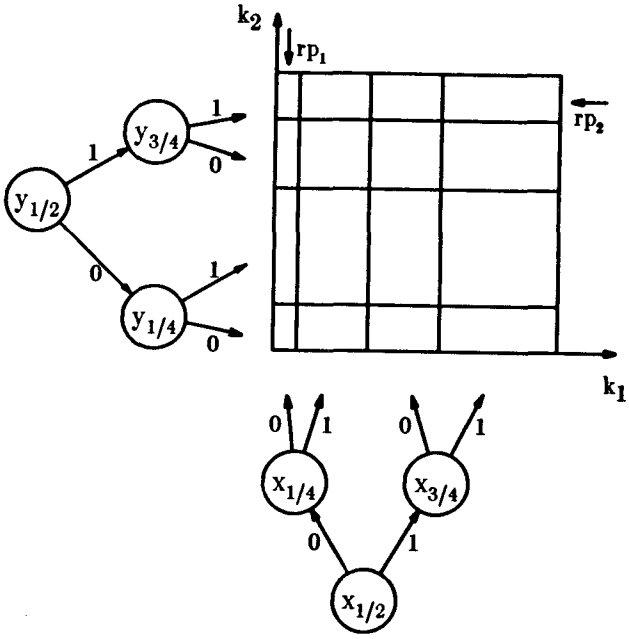


Figure 4.5

5. Experimental results

In order to demonstrate the performance of the quantile method for nonuniform distributions, we have implemented it on top of MOHLPE in MODULA-2 on an Olivetti M24 PC with 640 KB RAM and with hard disk. In MOLHPE the following parameters were chosen: the control function for the expansion of the file is expansion after 28 insertions, the capacity of a primary page is 31 records, and that of a secondary page is 7 records. From all the experiments, we select the following two for demonstrating the graceful adaption of the quantile method to a nonuniform distribution.

Experiment 5.1:
For $k_i \in [0,1]$, i =1,2, we have:
$Pb(k_i \leq 1/4) = 9/16$
$Pb(k_i \leq 1/2) = 7/8$
where Pb denotes the probability with which k_i is in the specified interval, i = 1,2. However, we only accept component keys k_i which follow the distribution and are not in [0.6, 0.7]. The application of the quantile method yields the partition of the key space depicted in figure 5.1. In figure 5.2 we have plotted the average number of page accesses in a successful exact match query with and without using the quantile method as a function of the level of the file. Figure 5.3 depicts the average storage utilization with and without quantile method for various levels of the file.

Let us remark that level = 9 corresponds to 15,000 records in the file and that the implemented version of MOLHPE uses p = 1 partial expansions. The implementation for $p \geq 2$ is currently on the way.

The advantage of the quantile method is obvious. The average number of page accesses in a successful exact match query is reduced from more than 4 to less than 1.5, the average storage utilization is improved from approxi-

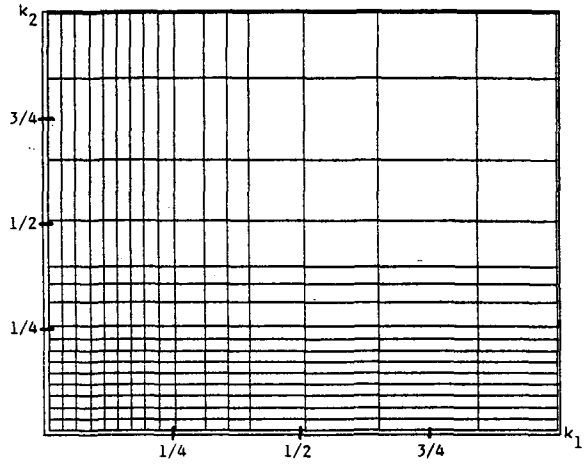


Figure 5.1: partition of the key space

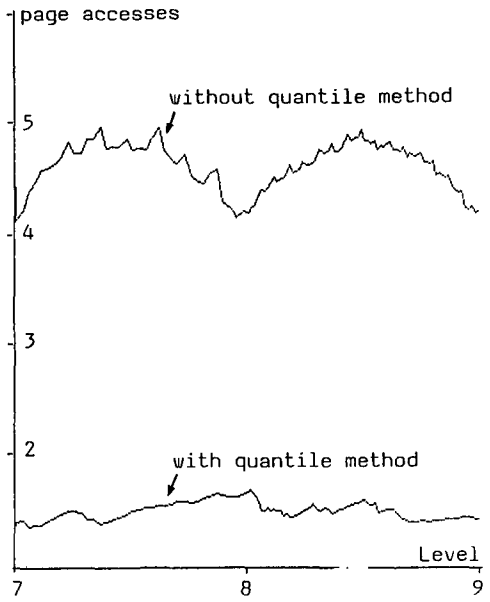


Fig.5.2: Average number of page accesses in a successful exact match query as a function of the level of the file

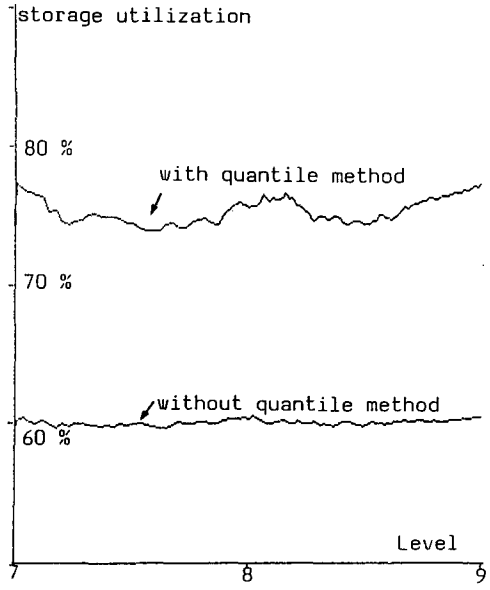


Fig.5.3: Average storage utilization as a function of the level of the file

mately 60% to more than 75%. Let us remark that although the grid file is tailorcut for best exact match performance it is easily outperformed by the quantile method. It will be interesting to observe the very superlinear growth of the grid directory for this nonuniform distribution.

Experiment 5.2:
For each k_i, i = 1,2, the component keys follow a Gaussian distribution with expected value 0.5 and variance 1. Furthermore, the component keys are multiplied with the factor 1/8 and the composite keys are in $[0,1]^2$. The application of the quantile method yields the partition of the key space depicted in figure 5.4 . This partition gives an idea of the graceful adaption, depicted for level = 10. In figure 5.5 we have plotted the average number of page accesses in a successful exact match query with and without the quantile method for levels = 8,9,10. Here level = 10 corresponds to 30,000 records. Since the performance of our scheme without quantile method is cyclic and stationary, we did not extend this experiment beyond level = 10. However, figure 5.6 clearly shows the positive trend for the performance of the quantile method with increasing file size. We do not depict storage utilization. Using the quantile method it exceeds 70% for levels larger than 9.

The improvement in page accesses by using the quantile method is even more drastic in this experiment than in the previous one. For level = 9 the number of page accesses is improved from approximately 8 to less than 2. Comparing exact match queries is clearly to the advantage of the grid file. But even for this type of query the quantile method outperforms the grid file for large enough files. For file sizes around 10^6 we expect the quantile method to perform successful exact match queries in practically 1 disk access. The most drastic advantage of the quantile method over the grid file for nonuniform distributions will turn out for complex queries such as range queries. Here the possibly

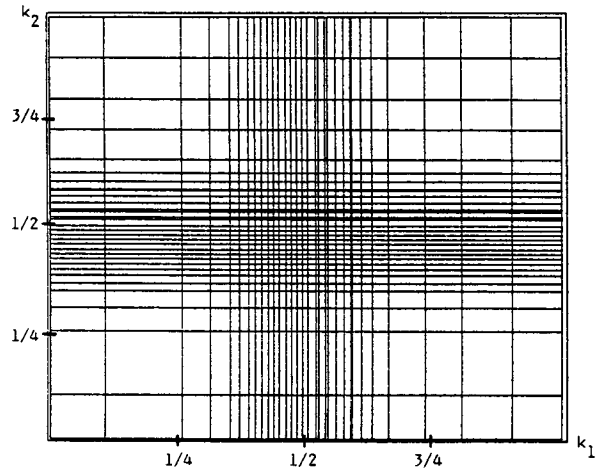


Figure 5.4: partition of the key space (Gaussian distribution)

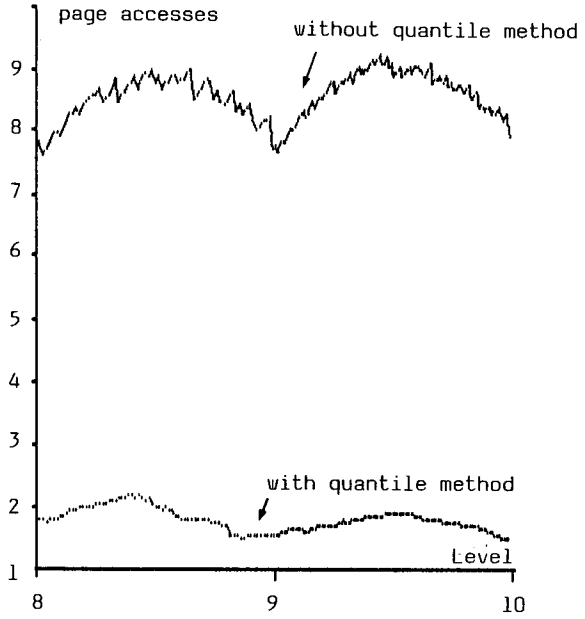


Fig.5.5: Average number of page accesses in a successful exact match query

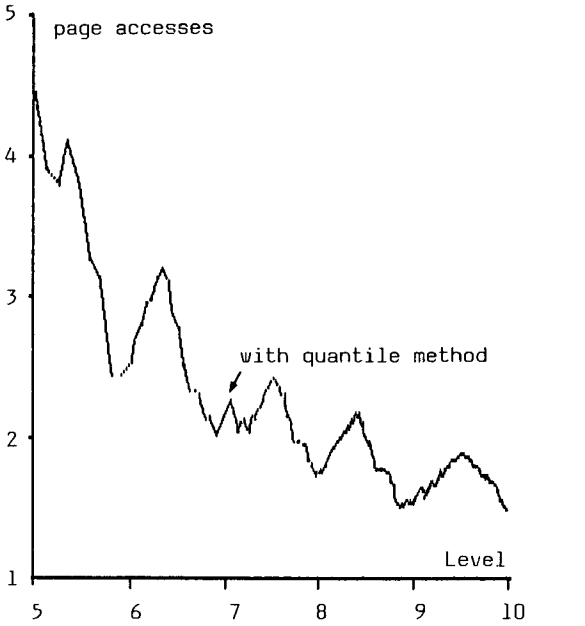


Fig.5.6: Average number of page accesses in a successful exact match query as a function of the level of the file

exponential growth of the grid file will result in very poor query times. Experiments to compare very efficient grid file versions and the quantile method for range queries are under way.

6. Conclusions

Our goal was twofold. First, for uniform record distribution we have suggested multidimensional order preserving linear hashing with partial expansions (MOLHPE) [KS 86] which performs better than its competitors. Second, for nonuniform record distributions all known multidimensional dynamic hashing schemes exhibit very poor performance. By presenting the quantile method in this paper and applying it to MOLHPE we suggest a scheme which exhibits for nonuniform distributions practically the same ideal performance as for uniform distributions. First experimental runs of an implementation of our scheme underline this fact and show how graceful our scheme adapts to nonuniform distributions. Exact match queries can be performed in almost one disk access even for very nonuniform distributions. We expect an even clearer advantage of our scheme over the various grid file versions in case of more complex queries such as range queries. Experiments for this comparison are under way.

References

[Bur 83] Burkhard, W.A.:'Interpolation - based index maintenance', BIT 23, 274 - 294 (1983)

[Bur 84] Burkhard, W.A.:'Index maintenance for nonuniform record distributions', Proc. 3rd ACM SIGACT/SIGMOD Symp. on PoDS, 173 - 180 (1984)

[Hin 85] Hinrichs, K.:'The grid file system: implementation and case studies of applications',Ph.D. Dissertation, Swiss Federal Institute of Technology, Zürich (1985)

[Kri 84] Kriegel, H.P.: 'Performance comparison of index structures for multi-key retrieval', Proc. 1984 ACM/SIGMOD Int. Conf. on Management of Data, 186 - 196

[KS 86] Kriegel, H.P., Seeger, B.: 'Multidimensional order preserving linear hashing with partial expansions',Proc. ICDT'86, International Conference on Database Theory, Rome, Sept 8-10,1986, Proceedings to appear in the Lecture Notes in Computer Science series

[KW 85] Krishnamurthy, R., Whang, K.-Y.: 'Multilevel grid files', Draft Report, IBM Research Lab., Yorktown Heights (1985)

[KC 78] Kushner, H.J., Clark, D.S.:'Stochastic approximation methods for constrained and unconstrained systems', Applied Mathematical Sciences No. 26, Springer-Verlag (1978)

[Lar 80] Larson, P.-A.: 'Linear hashing with partial expansions', Proc. 6th Int. Conf. on VLDB, 224 - 232 (1980)

[Lit 80] Litwin, W.: 'Linear hashing: a new tool for file and table addressing', Proc. 6th Int. Conf. on VLDB, 212 - 223 (1980)

[NHS 84] Nievergelt, J., Hinterberger, H., Sevcik, K.C.: 'The grid file: an adaptable, symmetric multikey file structure', ACM TODS, 9, 1, 38 - 71 (1984)

[Oto 84] Otoo, E.J.: 'A mapping function for the directory of a multidimensional extendible hashing', Proc. 10th Int. Conf. on VLDB, 491 - 506 (1984)

[Oto 85] Otoo, E.J.: 'Symmetric dynamic index maintenance scheme', Proc. of Int. Conf. on Foundations of Data Org., 283 - 296 (1985)

[Oto 86] Otoo, E.J.: 'Balanced multidimensional extendible hash tree', Proc. 5th ACM SIGACT/SIGMOD Symp. on PoDS, (1986)

[Ouk 85] Ouksel, M.: 'The interpolation-based grid file', Proc. 4th ACM SIGACT/SIGMOD Symp. on PoDS, 20 - 27 (1985)

[Rob 81] Robinson, J.T.: 'The K-D-B-tree: a search structure for large multidimensional dynamic indexes', Proc. 1981 ACM/SIGMOD Int. Conf. on Management of Data, 10 - 18 (1981)

[RM 55] Robbins, H., Monro, S.: 'A stochastic approximation method ', Annals of Mathematical Statistics 22, 400-407 (1955)