A Dynamic Signature Technique for Multimedia Databases

*F. Rabitti and *P. Zezula*

IEI-CNR, Pisa
Via S. Maria 46, 56126 Pisa, Italy

* Computing Center of Brno Technical University
Obrancu miru 21, 60200 Brno, Czechoslovakia

ABSTRACT

A signature file acts as a filtering mechanism to reduce the amount of data that needs to be searched during query evaluation. Even though several techniques for organizing and searching signature files have been proposed in literature, they have serious limitations when applied to multimedia databases, where integrated access methods to text and image content are neeeded.
A new signature technique, called Quick Filter, is proposed in the paper. According to this technique, signatures are divided into partitions, each of which holds signatures sharing the same characteristic key. As a result, it is possible to determine if the signatures in a partition satisfy a query by merely examining the key. Partitions not matching the key need not be searched. This method is based on dynamic hashing since signatures are hashed into partitions according to the keys and the file size, computed algorithmically from the signatures. Implementation of this technique is illustrated using an example and is verified by analytical performance evaluation.
The result is a signature technique which satisfies the requirements for access methods in multimedia databases: dynamicity, with respect to insertions and updates, good query processing performance on large databases for high-weight queries.

1. INTRODUCTION

The signature file access method and its applications have received in recent years a large attention in the literature, e.g. [TSIC-83], [CHRI-84], [CHRI-86], [FALO-87], [SACK-87], [CHAN-89], [LEEL-89], etc. The advantages of signatures over inversion for text data was confirmed several times [RABI-84], [CHRI-84]. The authors agree that the signature file overhead is usually less than 10% of the size of initial data, while the inversion requires space between 50% and 300% of the size [HASK-81]. Moreover, signature techniques are much more flexible for insertion and update operations [RABI-84].

However, the increasingly sophisticated studies on signature techniques presented in literature have been mainly applied to rather simplistic application environments. Most performance studies concern the search of words in text documents, where equiprobabilistic distribution of words is assumed [CHRI-84], [FALO-87]. Other studies focus more on integrated signature techniques for data attributes and text, with the obvious difficulty in performing range queries on numerical data using the signatures [FALO-86].

There is a new area of application which is attaining increasing importance, i.e. the area of multimedia databases. A crucial point of multimedia database is the integration not only of formatted data and text but also of images [RABI-87] [RABIb-89]. Image data must be treated, in the system, with the same level of functionality as the other data (i.e. formatted data and text). This means that the system must be able not only to store images and store relationships with other data, but also to allow queries addressing the image content [RABIa-89].

Multimedia databases require access strategies which are much more complex than the access strategies for formated record systems or text document systems [THAN-90]. Present signatures techniques cannot be satisfactorily exploited as integrated access methods for multimedia database systems since they do not satisfy the operational requirements of these systems, such as the dynamicity and the query processing performance. The purpose of this paper is to propose a new signature technique, called *Quick Filter*, which fits the operational requirements as an integrated access method to multimedia databases.

In Sec. 3, we give an example of a multimedia database application concerning the retrieval of multimedia documents and we discuss the shortcomings of current signature techniques when applied to this application environment. Then, we derive the requirements for new signature techniques suitable for new application environments with these characteristics. In Sec. 4, we present the quick filter signature technique, its implementation and use in query processing. Then we evaluate its performance, mainly in relation with the requirements previously defined for the multimedia databases.

2. PRELIMINARIES ON SIGNATURE TECHNIQUES

The very idea of the *signature file access method* is to extract and compress properties of data objects and store them in a separate file. The extracted pieces of data are called *signatures*. Queries are supposed to be transformable to the signature form too. A collection of the derived signatures is called the *signature file* or the *filter* because of its role during the query processing. Signatures are connected to the data objects through unique object identifiers (OID). The function of the filter is to find all OIDs of data objects qualifying for a given query. In fact, the signature file access method allows some *false hits* on the signature file level. This is why a signature file is called "filter". Therefore, a second step of query processing, called *false drop resolution*, is needed.

The novelty of this method is the invention of the filter. Since objects are accessible by the OIDs, any direct access method for storing the objects can be used. In general, an efficient filter is a filter with very few false hits and fast filtering process. From the implementation point of view, the problem can be solved by the appropriate design of a *signature extraction method* and a *data structure* for organizing signatures.

2.1. Signature Extraction Methods

Probably the best review and analysis of the signature extraction methods is in [FALO-87]. It also contains the performance comparison which is based on the estimation of the *false drop* probability. The false drop is the situation, during query processing, in which a signature seems to qualify a query, while the corresponding object does not qualify. However, [FALO-87] does not consider data structures and their effects on the query processing performance. Such comparison can be found in [ZEZUa-90]. The basic types of the signature extraction methods are known under the names of *Word Signature (WS), Superimposed Coding (SC), Bit-Block Compression (BC), Run-Length Compression (RL).*

In SC, each data object descriptor yields a bit pattern of size f where m bits have the value "1", while the others have the value "0". These bit patterns are OR-ed together to form the object signature. The number of ones in a signature S is the *signature weight*, designated as $w(S)$. If an object signature contains ones in the same positions as the query signature does, then the object signature qualifies for the query. The time required for comparing two SC signatures is very short, in particular for query signatures with low weights.

Since we are mostly going to concentrate on SC, we will not survey the other signature extraction methods. Interested readers are referred to [CHRI-84], [FALO-86], and [FALO-87].

2.2. Data Structures for Signature Filters

Even though the false drop is an important measure for comparing different signature extraction methods, the performance of signature filters depends mainly on the I/O cost, i.e. on the number of physical pages which must be accessed to evaluate a query. If the false drop is low, we can save a lot of accesses on the storage level. The efficiency of filtering is determined by the storage structures and access strategies which support the filtering process.

The most important storage structures for organizing signatures are: ***Sequential Signatures (SS)***, ***Bit-slice Signatures (BS)***.

SS is the basic organization, which is easy to implement and is space efficient. Insertions are easy to perform and exhaustive processing is efficient. Performance of query processing is not dependent on the query signature weight and the response time is linearly proportional to the size of the signature file. That is the reason why it is not convenient for very large files. Many deletions and updates may require the file reorganization.

BS, suggested in [ROBE-79], is the best organization for processing queries with low weights. Increasing the query weights requires additional block accesses and for this reason BS filter cannot be recommended for queries with very high weights. Maintenance is extremely time consuming. Th that is why BS is only suitable for stable archives, where insertions and updates are not permitted.

3. REQUIREMENTS FOR NEW APPLICATIONS

Although very sophisticated studies on signature techniques have been presented in literature, they have been usually applied to rather simplistic application environments. Historically, they were studied as access methods to formatted records (for secondary non-numerical key access) and text [FALO-86].

However, new application environments, such as office systems, multimedia databases, etc., require access strategies which are much more complex than the access strategies for formatted record systems or text document systems. In this section, we give an example of a new application environment, concerning the retrieval of multimedia documents, we present a way of defining signature for such application, and we discuss the shortcomings of current signature techniques in this context. Then, we derive the requirements for new signature techniques suitable to be applied to these new application environments.

3.1. Experience in the MULTOS Project

MULTOS (MULtimedia Office Server) is an ESPRIT Project in the area Office Systems. It supports basic filing operations, such as creation, modification, and deletion of multimedia documents, and the ability to process queries on documents. Documents are stored in databases, integrating also Optical Disk media, to allow the storage of very large amounts of data. Documents and may be shared by several users: facilities like authorization, version, and concurrency control are supported. MULTOS is based on a client/server architecture. Three different types of document servers are supported: *current server*, *dynamic server* and *archive* *server*. They allow filing and retrieval of multimedia documents based on document collections, document types and document content.

The MULTOS system must be able to answer queries at a high level of abstraction, where conditions on different document components, such as free text, formatted attributes and images, are intermixed. In order to support a fast document retrieval process, the query processor uses special access structures to the document content. In the actual system, specialized and independent access structures are defined for the different document components. B+ tree indexes are used for formatted attributes. Other B-trees are used to index objects contained in the images. Images are analyzed according to particular application domains and symbolic information, in terms of objects recognized in the image, is extracted as result of the analysis process [THAN-90]. Access to image content is then performed only using this symbolic information and not the original image (which can be a raster image or a graphical image). Image objects are characterized by the particular application domain and by the plausibility and belief of their recognition [THAN-90].

Signature techniques are used, instead, for text access. Superimposed coding of the words contained in the textual part of the document is used as text signature extraction method. As signature filter organization, sequential signatures are used in the *current server* (i.e. the server containing updatable documents, on magnetic storage, where a lot of insertions are expected) and bit-slice signatures are used in the *archive server* (i.e. the server containing stable documents, on optical storage, where only retrieval operations are allowed). The use of these signature filter organizations is consistent with the characteristics of the two MULTOS servers: sequential signatures are flexible for insertion and update operations but are slower to search, bit-slice signature are not flexible for modifications (no modification is allowed on the archive server) but are faster to search (in case of low-weight queries, see Sec. 4.4).

However, the lack of integration of the different access methods to the different document components makes the query optimization an extremely difficult problem to solve. In [BERT-88] the query processing in MULTOS is presented in detail, but only with respect to formatted attribute and text components in the documents (the extension of the system with queries involving also image document components was added later on in the second MULTOS prototype [THAN-90]) and considering only sequential signature organization (not bit-slice organization, also added later on). Query processing algorithms resulted very complex, and a similar analytical study on query optimization taking into account image access methods was never attempted (in the second MULTOS prototype organization, simpler heuristics have been used [THAN-90]).

A possible solution could be to use of a unique signature technique as an integrated access method. With this solution query processing would be much simplified, being limited to the exhaustive search of a single signature filter. In the sequel, we focus on the integration of the access structures to text and image document components into a single signature filter. We still prefer to treat formatted attribute using separated indexes (i.e. B+ trees), as in [BERT-88], due to the difficulty to treat effectively range queries using signatures.

3.2. Integrated Signature Creation

We propose here a simple approach to create a unique signature, by superimposed coding, for text and image document content description. Then, we will discuss the implications in terms of requirements on the organization of the signature filter and in terms of performance of signature search.

A unique signature block (of f bits) is allocated for a document. The signature of the textual part of the document is generated in the usual way: each word in the text sets to 1 m bits in the signature block. A scheme where each word triplet sets a bit can be used to decide which of the f bit are set by a word. This allows for search on parts of words of the text.

An image, after the image analysis process, is represented symbolically in terms of the contained objects. Each object can be in turn composed of other objects.

An example of symbolic representation of an image is the following (for simplicity we omit positional information associated with each object, which is anyway neglected in the signature generation):

$$I = O_1,\ O_2(O_4,O_5(O_6)),\ O_3(O_9,O_{10}),\ O_1,\ O_3(O_9,O_6,O_7))$$

In this example we notice that in image I five objects have been recognized: two simple objects (O_1 has been detected in two positions in the image) and three complex objects (O_2 and twice O_3). Notice that O_3 has been recognized in two different formats: once as (O_9,O_{10}) and once as (O_9,O_6,O_7). In fact, the image analysis process is based on a body of rules which may lead to many different ways in recognizing the same semantic object.

The signature for image I is obtained by superimposing the codes of the objects recognized in it. Then, the signature for I is superimposed with the signature of the text associated to I.

For each application domain, the signature of an image object is fixed as n specific bit positions in the complete signature block (f bits). The codes of all possible objects in the domain are specified in a look-up table, which may be updated to reflect the changes in the rules of image analysis process (e.g. rules for new semantic objects or rules expressing more ways of recognizing the same semantic object can be added). A simple object (e.g. O_1 in image I) will set only n bits in the image signature, as specified in the application look-up table. A complex object, instead, will set its n bit position and the bit positions associated with all the simpler objects which compose it in the symbolic representation of the image. For example, the signature of O_2, defined in I as $(O_4,O_5(O_6))$ is obtained superimposing the codes, obtained from the look-up table, of O_2,O_4,O_5,O_6.

The values of m and n must be evaluated in relation to f, taking into account the average text length, the average number of objects in each image, the number of semantic objects in the application domain, etc. to approach the target of 1/2 of ones in the resulting signature block.

Query signatures are obtained superimposing the codes of the words and image objects which the user is looking for in the database. Query processing is then performed with a unique exhaustive search of the unifies text and image signature.

We should briefly discuss here the inaccuracy introduced by this signature extraction method. False hits caused by superimposed coding in text signature has been extensively studied [CHRI-84]. In our case, a further source of false hits is due to the superimposed coding of image content. Although every object in the application domain has a unique n bit signature, coded in the look-up table, the superimposition of the codes for different objects, and their superimposition with codes of words in the text, can cause false hits. Therefore, it is necessary to perform false drop resolution on images checking the query on the symbolic representation of the image (completed with information of relative positions of the objects, plausibility and belief of recognition, etc.) linked to the image itself (either raster of graphical). Since the false drop resolution is necessary for images, in this step it is possible to check for other kind of information which is lost in the image signature but may be requested in the user query, such as presence of several instances of the same object (e.g. a house with at least four windows), relative position of the objects (e.g. a window above a door), recognition degree of the objects (e.g. a door recognized with minimum belief 0.8).

3.3. Requirements for a New Signature Technique

Let us to summarize the main requirements resulting for these signature techniques:

A) must be used as *integrated* access method to multimedia databases (i.e. addressing text and image data).

B) must be fast in the exhaustive search of *large* multimedia databases.

C) must be suitable for *dynamic* environments, with frequent insertion and update operations.

D) must be efficient for *high-weight* queries.

The last requirement is due to the fact that conditions on text and images are merged in a unique query signature and that conditions on image objects may often be transformed into equivalent queries with higher weight. Suppose that O_i is a complex image object. If it has a unique definition, in the application domain, or all its definitions are based on the same objects $O_1, \cdots, O_n$, a query on O_i can be expanded superimposing the codes of $O_i, O_1, \cdots, O_n$. In case of multiple definitions of O_i, with partially disjoint sets of composing objects, it is either possible to ask the user to choose the preferred interpretation or to limit the query signature construction only to the objects belonging to all the definitions of O_i (i.e. the intersection).

As signature extraction method, superimposed coding (*SC*) satisfies requirement A, as we have seen in the previous section on integrated signature creation. More complex is to choose an adequate filter data structure. Consider the candidates SS and BS:

SS: The sequential signature technique satisfy requirement C, is indifferent to requirement D (search speed is not dependent on query weight), but does not satisfy requirement B. In fact, since the search time is linearly proportional to the database size, the response time is bad for large databases.

BS: The bit-slice signature satisfy requirement B but not requirement D. In fact it is in general very fast in searching large databases but performance deteriorates in case of queries with high weight (see the performance comparison in Sec. 4.4). Moreover, it does not satisfy requirement C, since insertions and updates require very time consuming reorganizations.

From this discussion, we can conclude that new signature data structures are required to meet requirements B, C and D and thus be suitable for multimedia databases.

4. QUICK FILTER

We call *Quick Filter* our proposal of a new signature technique satisfying the previous requirements. This technique allows the handling of dynamic signature files and an efficient processing of high-weight queries. This technique comprises the organizational aspects of data as well as the processing procedures for efficient query execution. In the following sections we present the basic idea, a possible implementation, and performance evaluations.

4.1. The Basic Idea

The quick filter is an organization of signature strings, not slices like the Bit-slice organization. The basic units of access are pages of signatures. From this respect, quick filter is much closer to SS. However, in the quick filter similar signatures are placed in a page. The criteria for grouping signatures in pages and distributing pages within the signature file are based on hashing. This kind of organization has been presented for the first time in [ZEZU-89]. Here we present more implementation details as well as performance evaluations and comparisons with other filters.

The implementation of the idea described above concerns mainly two design problems. At first, it is the hashing function which should be applied to a signature in order to find a storage page into which the signature belongs. The second problem is the query processing algorithm with the ability to save some, preferably many, of the pages from access. Dynamic data environment is another important design assumption. Specific solutions to the problems will be discussed in the following section.

4.2. Implementation

One of the important functional requirements for the quick filter is the management of dynamic data. A dynamic hashing schema for organizing files of records having single attribute keys, called *linear hashing*, has been presented in [LITW-80]. The most important extensions of the work are *linear hashing with partial expansions* [LARS-80] and *recursive linear hashing* [RAMA-84].

4.2.1. Linear hashing

The hash function of linear hashing, let's say g, maps the keys onto the address space $\{0,1,2,...,n-1\}$ where $2^{h-1} < n \le 2^h$, for some integer h. The value of h is called *level* of the file or hashing. A variation in h induces a variation in g. In fact, the function g must be a *split function*, which means that the following condition must hold:

$$g(K,h,n) = g(K,h-1,n) \text{ or,}$$
$$g(K,h,n) = g(K,h-1,n)+2^h,$$

for any key K from the file.

Briefly, the idea of linear hashing scheme involves a set of n *primary* (addressable) pages, each with zero or more overflow pages chained to it to form a list of pages. Assuming that the file level is h, then to insert a record with key K, the page address, p, is computed as $p=g(K,h,n)$. The record is stored in the primary page p except when an overflow occurs. In this case, it is stored in an overflow page linked to p. The occurrence of an overflow triggers an expansion of the address space from n to $n+1$ primary pages.

The expansion of the address space progresses by page splitting whenever an overflow occurs in a primary page. A pointer, denoted by SP, designates the primary page that is to be split next. Suppose a collision occurs in page p, for $0 \le p < n$. The key, being inserted, is stored in an overflow page chained to p. Besides, a new primary page, number n, is allocated and the keys in the primary page SP, as well as its overflow keys, are rehashed and distributed between the pages SP and n. The number of primary pages now becomes $n+1$. The values of SP and h are synchronized as follows:

(1) the file level h is increased just before the primary page 0 is split,

(2) the pointer SP is advanced according to the assignment $SP = (SP+1) \bmod 2^{h-1}$.

4.2.2. Hashing function for signatures

Now, since an object signature, S_i, $i=1,2,...,N$ (N is the number of signatures in the signature file), is a sequence of f binary digits $b_1,b_2, \cdots ,b_f$, let's suppose them to be the keys and let n be the number of allocated pages addressed from 0 to $n-1$. A hash split function of the signatures can be defined for $h>0$ as:

$$g(S_i, h, n) = \begin{cases} \sum_{r=0}^{h-1} b_{f-r}\, 2^r, & \text{if } \sum_{r=0}^{h-1} b_{f-r}\, 2^r < n; \\ \sum_{r=0}^{h-2} b_{f-r}\, 2^r, & \text{otherwise.} \end{cases}$$

(Eq. 1) Hashing function

For the initial condition, it is $h=0$, $n=1$, we define $g(S_i,0,1)=0$.

In fact, what the hashing function g does is that it takes the h-bit, or the $(h-1)$-bit, suffix of S_i and interprets it as an integer value. The value of g is always a non-negative integer smaller than n.

The important corollary of this scheme is that N signatures can be stored in n pages in $O(n)$ page accesses and that pages with signatures can be accessed for the retrieval purposes by consecutive physical page accesses.

4.2.3. Exhaustive search

Sequential processing of the stored signatures is easy. The only thing we must do is to generate the page characteristics p in the range from 0 to $n-1$ and access the pages. If the pages are allocated on a continuous part of a dedicated disk memory, then the best thing we can do is to access them in the increasing (decreasing) order. However, sequential processing of hashed signatures is not as efficient as the processing of the sequential file. The main reasons are the overflow and a lower page load which can be for the linear hashing expected. Fortunately, the expected overflow is not high. Since signatures are usually not large, mostly less than 100 bytes, we can store many signatures in a page. For this case Litwin [LITW-80] reports practically no overflow with the page load of 50%. When the load was controlled and guaranteed to be 75%, the observed overflow was only 5%.

Furthermore, we can adopt a more efficient access strategy for the exhaustive search of signatures organized by linear hashing. All the primary pages can be accessed first in the most efficient way, i.e. with the increasing page number. Since the order of accessing signatures is not for the exhaustive search important, signatures in the overflow pages, if any, can be read in the second step, preferably in the batch access mode.

4.2.4. Search space reduction

The way how the linear hashing scheme can be exploited for a more efficient query processing is explained in detail in [ZEZU-89]. Briefly, since queries are translated into query signatures, they are also bit patterns of size f. The number of 1's in the query signatures ranges from m to $f/2$, supposing the signatures are designed as optimum. Let recall here that according to [FALO-87], optimum superimposed signatures have 50% 1's and 50% 0's. The *weight* of a specific query signature, $w(Q)$, depends on the number of terms specified in the query.

However, the typical query signature weight $w(Q)$ is usually smaller than $w(S_i)$, the weight of the i-th object signature S_i. But also in this case we can compute a characteristic key of Q as $p=g(Q,h,n)$. The value of p is in fact the smallest number of the primary page which must be accessed and its signatures tested for qualification. If $p=0$, all pages must be accessed, and exhaustive search must be used. The zero value of p also means that there are no 1's in the h-bit suffix of Q. But if there are some bits with value 1, the number of accessed pages can be decreased considerably.

For the sake of simplicity, we will suppose now that $n=2^h$. Let further assume that there is just one bit with value 1 in the h-bit suffix of Q. Then the number of accessed pages can be reduced to 2^{h-1} and the rest of the primary pages, which is also 2^{h-1}, do not have to be accessed. The reason is obvious since any h-bit binary integer with j bits having a fixed value, e.g. 1, in any specific position, can have at most 2^{h-j} different values.

According to this, if there is a query signature Q with j 1's in its h-bit suffix, $j=w(h(Q))$, it is enough to read only 2^{h-j} primary pages and their overflow areas instead of using the exhaustive search. This is the main idea of the implementation of the query processing algorithm of the quick filter. The actual numbers of read pages are decided by the Algorithm 1. The pages are accessed in the *semi-consecutive* page retrieval mode with the increasing page number, starting in the page number $g(Q,h,n)$.

Algorithm 1:

```
1. $P := g(Q,h,n)$
2. IF $h(Q) \cap P \equiv h(Q)$ THEN
       access the page $P$ to match its signatures for qualification
3. $P := P+1$
4. IF $P<n$ THEN GOTO 2.
5. END of the query processing
```

where:

P is an h-bit binary integer,
n is the number of addressable pages,
h is the level of the file
Q is the query signature,
g is the hashing function,
$h(Q)$ is the h-bit suffix of Q.

4.2.5. An example

Let's end up the implementation part of this paper with an illustration. For convenience, we use a set of six superimposed signatures. The size of each signature is f = 8. The data can be seen in the Figure 1. as a table consisting of six rows and eight columns. To demonstrate the searching ability of the quick filter we define the sample query signature Q as the following:

sample query signature Q: 00100010.

As we can see, there are just two once in our query signature, that means that the weight of the signature is w(Q) = 2.

	C1	C2	C3	C4	C5	C6	C7	C8
S1	0	0	0	1	1	1	1	0
S2	1	1	0	1	0	0	0	1
S3	0	0	1	1	1	1	0	0
S4	1	1	0	0	0	0	1	1
S5	0	0	1	1	0	1	1	0
S6	1	1	0	0	1	0	0	1

Figure 1. Sample Data of a Signature Filter

In Figure 2 we show how our sample data can be inserted into the filter and in Figure 3 how the sample query can be performed. We suppose that the capacity of a page is two signatures. The process of filling the filter is shown in seven steps. The step number 0 corresponds to the initial state when the filter is empty. The other steps show the content of the pages and the values of the file characteristics (split pointer *SP*, level of hashing *h*, and number of addressable pages *n*) always after inserting a signature. The signatures are inserted in the natural order, it is S1, S2, ... , S6.

Step 0.	**P0**: empty				*SP=0, h=0, n=1*
Step 1.	**P0**: S1				*SP=0, h=0, n=1*
Step 2.	**P0**: S1 S2				*SP=0, h=0, n=1*
Step 3.	**P0**: S1 S3	**P1**: S2			*SP=0, h=1, n=2*
Step 4.	**P0**: S1 S3	**P1**: S2 S4			*SP=0, h=1, n=2*
Step 5.	**P0**: S3	**P1**: S2 S4	**P2**: S1 S5		*SP=1, h=2, n=3*
Step 6.	**P0**: S3	**P1**: S2 S6	**P2**: S1 S5	**P3**: S4	*SP=0, h=2, n=4*

Figure 2. Inserting signatures organized by linear hashing

The resulting arrangement of the data can be seen in the Figure 3. The important feature of this organization is that all signatures in a page have the same suffix, the characteristic key, which in our case is two bits long. Considering the sample query, we can easily deduce that pages P0 and P1 cannot contain qualifying signatures, because none of their object signatures contain value "1" in the 7th position (column) as it is required by the query Q. In this way we can save page accesses and speed up the search. In our example we have to access two pages, P2 and P3, but we can save another two page accesses, namely to the pages P0 and P1.

P0: 00111100	**P1**: 11010001 11001001
P2: 00011110 00110110	**P3**: 11000011

Figure 3. Quick Filter

4.3. Performance Evaluation

It is the well known fact that every new suggestion for organizing data is not complete without appending it with qualitative evaluations. However, even though the quick filter is currently being implemented for experimental use as an access structure of MULTOS [THAN-90], no practical results are available at the moment. That is why we have decided to use modelling techniques to investigate the performance ability of quick filter.

We present two groups of performance tests, both of them based on block (page) access estimations. In the first group we study only the quick filter, namely its relationships among the type of query, size of the signature file and the efficiency of query processing. In the second group of performance analysis we compare the quick filter with the other access structures.

4.3.1. Quick filter analysis

It is not difficult to realize that the number of not accessed pages depends on the number of 1's in the h-bit suffix of Q, which in turn depends on the query weight and the size of h. Analytical formulas for estimating the number of pages which do not have to be accessed, provided the total number of pages n, level of hashing h, signature size f, and query weight w(Q) are given, have been derived in [ZEZU-89]. From this article we present here a formula, see Eq. 2, for computing the expected number of 1's in the h bit suffix of Q. This formula has been used as the basic algorithm for our block access estimations.

$$E[\dot{w}(Q,h)] = \sum_{j=1}^{\min[h,w(Q)]} \left[j \prod_{i=1}^{j} \frac{(h-i+1)\,(w(Q)-i+1)}{i\,(f-w(Q)+i)} \prod_{i=1}^{w(Q)-j} \frac{f-h-i+1}{f-i+1} \right]$$

(Eq. 2) Expected number of 1's in h-bit suffix of Q

The weight of a query depends on the application, namely on the way how queries are specified. Thus, the performance of query processing cannot be tuned by the database design.

Therefore the relationship between the query weight and the performance is very important so that embarrassment from imperfect or bad function of the filter can be avoided.

For the demonstration purposes we define *savings* as the percentage of accessed pages from the total number of addressable pages from which the filter is formed. In Figure 4, we can see dependences between the savings and the file size expressed in the number of pages. We show several curves representing query signatures composed of different numbers of terms, provided the total number of image descriptors $D=40$, signature size $f=600$ bits, and the number of bits which each term puts to one $m=10$.

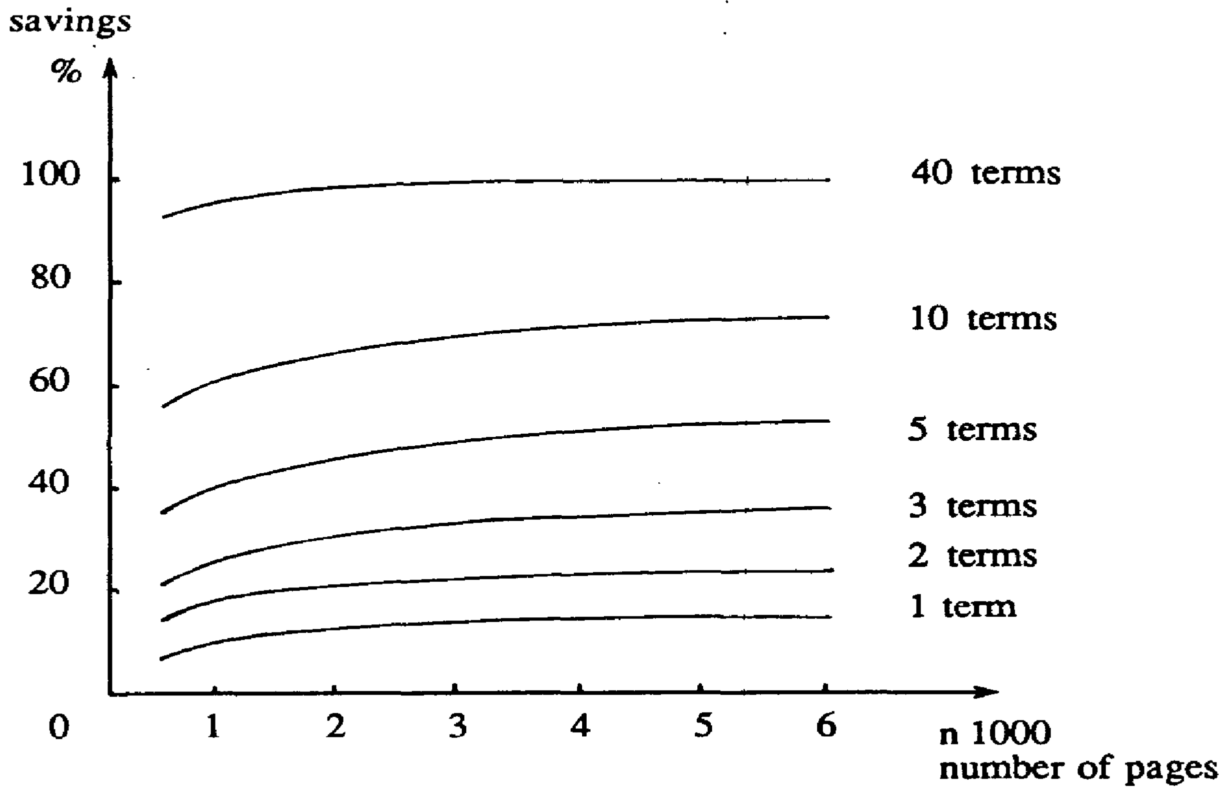

Figure 4. Percentage of savings depending on signature file size and query type
f = 600, m = 10, D =40

The figure demonstrates mainly the advantage of quick filter when processing queries containing many terms, that is to say, the high weight queries. But you can also observe that the performance is better for large rather than small files and the fact is true for any type of query.

4.4. Comparison with other access structures

In this section we would like to show the performance of the quick filter in the relation to the performance of the other access structures namely, the sequential, and Bit-slice.

As we have indicated, we used the analytical approach. We computed two performance characteristics for the quick filter, the sequential, and the Bit-slice organizations. Actually, we estimated the number of accessed pages needed for query evaluation and the space requirements, measured again in number of pages which the signature files occupied. Page size of 2K, 4 byte pointers to objects in the storage level, and the image signature weights of 120 bits, were accepted as the model assumptions.

The specific features of the modeled storage structures were built on the following additional assumptions. The pages of the sequential organization were filled with as many signatures as they could contain. All the bit-slices of the Bit-slice organization started in new pages and no page contained bits of more than one slice. Storage utilization for the quick filter was considered 75% and 5% of signatures was stored in overflow area.

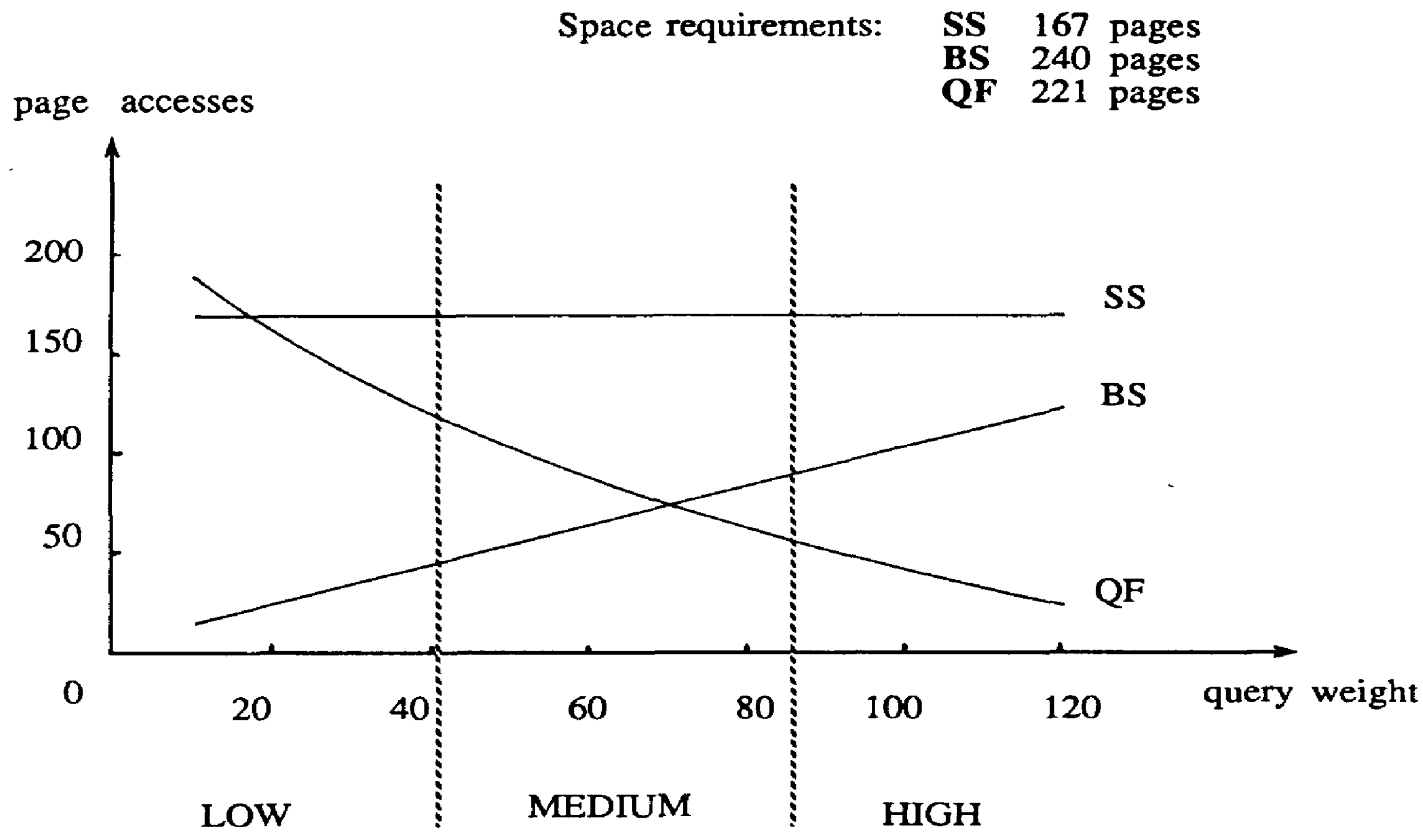

Figure 5. Comparison of signature file access structures
10,000 signatures, 2K pages, and image signature weight of 120bits

The results of the performance comparison are summarized in the Figure 5. According to these, we can say that quick filter, QF, is a complementary access structure to the Bit-slice organization. It is nearly always better than the sequential organization. The performance of quick filter for the high-weight queries is far better than the performance of any other data structure. But also for the medium weight queries, quick filter is a serious competitor even to the bit-slice organization, not to mention the sequential organization for which the query processing requires approximately twice as many page accesses than quick filter.

Unlike with the bit-sliced access structure, maintenance of the quick filter can be handled easily, because the file can both grow and shrink linearly while the load stays very high. Since the signature patterns are generated by using the random number generator, the distribution of 1's and 0's within the signatures is uniform. The overflow is low and as a consequence of this, exhaustive search is nearly as good as the exhaustive search of the sequential signatures. High weight queries, however, need to access much less blocks than any other organization and in this respect quick filter shows considerable performance improvements.

4.5. Extensions and Research Directions

As a result of our analysis we can see three ways of how to further increase the performance of the quick filter. We suggest to concentrate the effort on:
- decreasing the overflow and/or increasing the loading factor,
- increasing the selectivity of the query processing algorithm,
- clustering pages of signatures to reduce random disk accesses.

We would also like to mention another related work, [LEEL-89], which deals with partitioned signature files. The aim of the work is to provide algorithms which can divide signature files into partitions, so that both search space reduction and parallel processing can be achieved. Three different partitioning algorithms are presented and compared there. In fact, one of the partitioning algorithms, called the *Fixed Prefix Partitioning*, is quite similar to our hashing function g. The main difference is that, unlike the partitioning algorithms in [LEEL-89], the function g is the split function. It can distribute dynamic sets of signatures into partitions and guarantee reduction of search space whenever possible. We are convinced that application of the other algorithms to dynamic file organizations may result in interesting storage structure designs worth investigation.

5. FINAL REMARKS

In this paper, we have introduced a new access structure for storing and retrieving signatures, based on hashing, called quick filter. Both the hashing function and the application of linear hashing as underlying data structure are defined and illustrated using an example. We have also presented the query processing algorithm with favorable performance characteristics. The query processing performance is investigated in terms of the number of pages accesses.

Results show that quick filter is mainly convenient in applications where large files are to be searched, insertions and updates are frequent and user queries mostly result in high weight signature queries. These characteristics fit the requirements, discussed in Sec. 3, for signature techniques to be used as integrated access methods to text and image data in multimedia databases.

REFERENCES

[BERT-88]
Bertino E., Rabitti F., Gibbs S., "*Query Processing in a Multimedia Document System*", ACM Trans. on Office Information Systems, Vol.6, N.1, pp. 1-41, 1988.

[CHAN-89]
Chang W. W., and Schek H.J.: "*A Signature Access Method for the Starburst Database System*". Proc. of VLDB-89, Amsterdam, The Netherlands, 1989, pp. 145 - 153.

[CHRI-84]
Christodoulakis S. and Faloutsos C. "*Signature Files: An Access Method for Documents and its Analytical Performance Evaluation*". ACM Transactions on Office Information Systems, Vol. 2, N. 4, pp. 267-288, 1984.

[CHRI-86]
Christodoulakis S., et. al. "*Multimedia Document Presentation, Information Extraction and Document Formation in MINOS: A Model and a System*". ACM Transactions on Office Information Systems, Vol. 4, N. 4, pp. 345-383, 1986.

[FALO-86]
Faloutsos C. "*Integrated Access Methods for Message Using Signature Files*". Proc. of the IFIP Working Conference on Methods and Tools for Office Systems, Pisa, Italy, 1986, pp. 135-157.

[FALO-87]
Faloutsos C. and Christodoulakis S. "*Description and Performance Analysis of Signature File Methods for Office Filing*". ACM Trans. on Office Information Systems, Vol. 5, No. 3, 1987.

[HASK-81]
Haskin R.L. "*Special-purpose Processor for Text Retrieval*". Database Engineering Vol. 4, No. 1, 1981.

[LARS-80]
Larson P. "*Linear hashing with partial expansions*". Proceedings of the 6th International Conference on VLDB, 1980, pp. 212-223.

[LEEL-89]
Lee D.L. and Leng C. "*Partitioned Signature Files: Design Issues and Performance Evaluation*". ACM Transactions on Office Information Systems, Vol. 7, No. 2, April 1989. pp. 158-180.

[LITW-80]
Litwin W. "*Linear hashing: a new tool for files and table addressing*". Proc. 6th International Conference on Very Large Databases, Montreal, 1980, pp. 212-223.

[RABI-84]
Rabitti F. and Zizka J. "*Evaluation of Access Methods to Text Documents in Office Systems*". Proc. 3rd Joint ACM-BCS Symposium on Research and Development in Information Retrieval, Cambridge, England, 1984.

[RABI-87]
Rabitti F. and Stanchev P., "*An Approach to Image Retrieval from Large Image Databases*", Proc. ACM-SIGIR 1987 International Conference on Research and Development in Information Retrieval, New Orleans, June 3-5, 1987.

[RABIa-89]
F. Rabitti, P. Stanchev, "*GRIM_DBMS: a GRaphical IMage Data Base Management System*" Proc. IFIP TC-2 Working Conference on Visual Database Systems, Tokyo, April 1989, in "Visual Database Systems", edited by T. L. Kunii, North-Holland, pp. 415-430, 1989.

[RABIb-89]
F. Rabitti, P. Stanchev, "*Image Database Management Systems: Applied Theories, Tools and Decisions*" Proc. of the Third International Conference on Automatic Image Processing, CAIP-89, Leipzig, GDR, September 1989, in "Computer Analysis of Images and Patterns", edited by K. Voss, D. Chetverikov, G. Sommer, Akademie-Verlag, Berlin, pp. 208-214, 1989.

[ROBE-79]
Roberts C.S. "*Partial Match Retrieval via the Method of the Superimposed Codes*". Proc. of IEEE, Vol. 67, No. 12, pp. 1624-1642, Dec. 1979.

[RAMA-84]
Ramamohanarao K. and Sacks-Davis R. "*Recursive Linear Hashing*". ACM Transactions on Database Systems, Vol. 9,No. 3, (September 1984), pp. 369-391.

[SACK-87]
Sacks-Davis K. and Ramamohanarao A. "*Multikey Access Methods Based on Superimposed Coding Techniques*". ACM Transactions on Database Systems, Vol. 12, No. 4, December 1987.

[THAN-90]
"*Multimedia Office Filing and Retrieval: The MULTOS Approach*", edited by C. Thanos, North-Holland Series in Human Factors in Information Technology, North-Holland, 1990.

[TSIC-83]
Tsichritzis, D. et. al., "*A Multimedia Office Filing System*". Proc. of VLDB-83, Florence, Italy, Oct. 1983.

[ZEZU-89]
Zezula P. "*Linear Hashing for Signature Files*". In the book "*Network Information Processing systems*", edited by K. Boyanov and R. Angelinov, Elsevier Science Publishers (North-Holland), IFIP, 1989, pp. 243-250.

[ZEZUa-90]
Zezula P. and Tiberio P., "*Engineering Signatures for Multimedia Data*", submitted for publication.

[ZEZUb-90]
Zezula P., Tiberio P. and F. Rabitti, "*Quick Filter*", IEI-CNR Tech. Rep. B4-07, Pisa, Febr. 1990.