Signature Files: An Integrated Access Method for Formatted and Unformatted Databases

Deniz Aktug[*] Fazli Can[†]

[*]Miami University, commons-admin@lib.muohio.edu

[†]Miami University, commons-admin@lib.muohio.edu

TECHNICAL REPORT: MU-SEAS-CSA-1993-006

Signature Files: An Integrated Access Method for Formatted and Unformatted Databases

Deniz Aktug and Fazli Can

SIGNATURE FILES: AN INTEGRATED ACCESS METHOD FOR FORMATTED AND UNFORMATTED DATABASES[*]

by

Deniz AKTUG Fazli CAN

Systems Analysis Department
Miami University
Oxford, OH 45056

Working Paper #93-006 May, 1993

[*] This paper has been submitted for outside publication and will be copyrighted if accepted. It is being published in technical report form to expedite the dissemination of its contents. Its distribution should therefore be limited to peer communication and specific requests.

SIGNATURE FILES: AN INTEGRATED ACCESS METHOD FOR FORMATTED AND UNFORMATTED DATABASES

Deniz AKTUG **Fazli CAN***

Department of Systems Analysis
Miami University
Oxford, OH 45056

May 4, 1993

Abstract

The signature file approach is one of the most powerful information storage and retrieval techniques which is used for finding the data objects that are relevant to the user queries. The main idea of all signature based schemes is to reflect the essence of the data items into bit patterns (descriptors or signatures) and store them in a separate file which acts as a filter to eliminate the non qualifying data items for an information request. It provides an integrated access method for both formatted and unformatted databases. A comparative overview and discussion of the proposed signature generation methods and the major signature file organization schemes are presented. Applications of the signature techniques to formatted and unformatted databases, single and multiterm query cases, serial and parallel architecture, static and dynamic environments are provided with a special emphasis on the multimedia databases where the pioneering prototype systems using signatures yield highly encouraging results.

Categories and Subject Descriptors: E.4 **[Data]**: Coding and Information Theory - *data compaction and compression*; E.5 **[Data]**: Files; H.2.2 **[Database Management]**: Physical Design-*access methods*; H.3.2 **[Information Storage and Retrieval]**: Information Storage-*file organization*; H.3.3 **[Information Storage and Retrieval]**: Information Search and Retrieval; H.4.1 **[Information Systems Applications]**: Office Automation; I.7.0 **[Text Processing]**: Text Editing

General terms: Design, Performance

Additional Key Words and Phrases: Access methods, descriptors, document retrieval, dynamic file, file design, hashing, information retrieval, information theory, inverted files, multimedia data, office automation, partial-match retrieval, record signatures, signature files, superimposed coding, term discrimination power, text retrieval, word signatures

INTRODUCTION

In spite of the latest efforts to develop more powerful database management systems for attribute type data, there still is a need for an integrated access method that will be applicable for both formatted and unformatted data. In addition to those environments where text and formatted data are used side by side (like office automation systems), more complex applications require handling of various media such as image, graphics, voice, sound and video. Examples of possible applications of signatures to such cases include

* To whom all correspondence should be addressed
voice: (513) 529-5950, fax: (513) 529-3841, e-mail: fc74sanf@miamiu.bitnet

automated law and patent offices, archival systems, computerized libraries, design applications (CAD), integrated manufacturing systems (CIM), Prolog database indexing, statistical databases, DNA matching in chemical databases and multimedia document retrieval [Colomb and Jayasooriah 1986; Faloutsos 1985; Faloutsos 1988a, Ramamohanarao and Shepherd 1986; Tiberio and Zezula 1991; Zezula et al. 1991].

Using the signature approach, the essence of the data objects (messages, documents, image representations, etc.) are extracted and stored in a separate file where each object is represented as a bit string or signature. This file of abstractions reveals the information content of the original source (with some loss due to the nature of the signature extraction process) and has a smaller size (typically 10-15 % of the original file) [Faloutsos 1992; Tiberio and Zezula 1991]. Upon a retrieval request, a two stage process is applied: In the first stage, the signature of the query is created and compared against the entries of the signature file to find the qualifying signatures whose corresponding objects are to be retrieved as a response to the specified query. The second stage consists of retrieving the objects with the qualifying signatures only. The process in the first stage is much simpler than scanning the original file since only bit strings consisting of a sequence of 1s and 0s are involved rather than the original data. Besides, the outcome of the first stage acts as a filter to limit the number of the objects to be considered in the second stage since only the ones with qualifying signatures need to be accessed.

Due to the information loss that takes place during signature generation, some signatures seem to qualify the queries although the corresponding objects do not. This situation, known as a false drop or a false match, leads to unnecessary disk accesses since it cannot be resolved until the original data objects are accessed. The description of the typical signature based retrieval process is depicted in Figure 1.

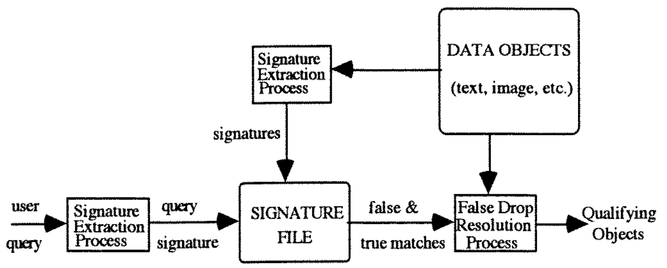


Figure 1. Description of signature-based retrieval process.

A good deal of research on signatures is devoted to the search for signature generation schemes that will minimize the occurrence of false drops. A second concern of the research on signatures is the signature file organization methods that will provide faster response without generating too much storage overhead and extra difficulty during the update operations.

The goal of this survey is to familiarize the readers with the basic idea behind information storage and retrieval using signature files and then provide a comparative overview of the methods proposed for generating and organizing signatures. We present examples to explicitly indicate how these methods are used in various applications and how the results compare. We aim to link the concepts by following a chronological order where it is possible, review different methods and emphasize the strengths and weaknesses of each method compared to the previous ones. In describing the methods, we emphasize the basic idea from which the methods originate and present an intuitive explanation instead of giving too many technical details. Interested readers are referred to the related references for a deeper understanding of the specifics of these methods.

Table of Contents

This survey assumes knowledge of the fundamental concepts of database management, and information storage and retrieval. The next section provides an overview of the basic concepts to familiarize the readers with the terminology that will be used in the following sections. The readers with a previous acquaintance with the signature file concept can skip this part and proceed to Section II which is allocated to the signature generation methods. Sections III and IV discuss single and multilevel signature file organization methods by giving specific examples for each case. Horizontally partitioned signature files are presented in Section V with a special emphasis on the incorporation of the concepts of signature extraction and signature organization. Section VI provides an overview of signature processors, discusses the use of signature files with parallel computer architecture and reviews a recently proposed combinatorial organization method. The application of signature files to multimedia databases are given in Section VII. Section VIII provides a summary and reevaluation of signature generation and organization methods. It also includes a discussion of the recent studies that compare the performance of various signature file organizations. Concluding remarks and pointers for future research are presented in Section IX.

I. BASIC CONCEPTS

The research on information retrieval (IR) and database management systems (DBMSs) has grown significantly with the extensive use of electronically stored data. Both systems aim to retrieve the information that will answer the user query where the user satisfaction depends not only on the content of the system output but also on the response time. DBMSs deal with formatted data consisting of records which are made up of fixed number of attribute values, each corresponding to an attribute [Date 1990; Ozkarahan 1986; Ullman 1988]. Information retrieval systems (IRSs), on the other hand, deal with unformatted (e.g., text) data where the database is made up of documents which are unstructured (e.g., can contain an arbitrary number of words) compared to the records [Salton and McGill 1983; Salton 1989; Van Rijsbergen 1979].

A more structured representation of a document can be obtained by using a process called indexing which generates a list of key words to represent the document contents and help distinguish it from others [Faloutsos 1985; Ozkarahan and Can 1986; Salton 1989; Van Rijsbergen 1979]. This task of indexing can be performed either manually by trained experts (manual indexing) or automatically with the aid of the computing equipment (automatic indexing) [Can and Ozkarahan 1987; Ozkarahan 1986]. A second decision concerning the use of a controlled versus uncontrolled indexing vocabulary should also be made. Some experts prefer the controlled vocabulary since they believe that the inclusion of all words can lead to ambiguity and errors. A controlled vocabulary, on the other hand, helps control of spelling and elimination of synonyms by using a unique word for each synonym group [Salton 1975; Salton and McGill 1983].

Terms like "and," "the," "or," "but," etc. which are called stopwords do not have any discriminatory power since they have no effect on document identification and differentiation. On the other hand, terms that have low occurrence frequency are usually used frequently in the queries since they have high selectivity which helps discriminating certain documents from the others. Indexing methods take the term discriminatory power values into account to accomplish high performance [Can and Ozkarahan 1987].

In IR, a document containing a query term is not necessarily relevant since relevancy is achieved only when the retrieved document is deemed pertinent by the system user. So, the concern in IR is not only existence but also relevancy and there is an ambiguity as to which documents qualify and which do not [Blair 1990; Van Rijsbergen 1979]. Since constructing a satisfactory query at the first time is a difficult task in an IRS, query modifications (by the system, user or both) take place most of the time. In Boolean systems, set numbers are attached to the retrieved document groups whose sizes are also provided. This gives the user the opportunity to create more complex queries by using the union and intersection operations on the given sets. Relevance feedback is another capability where the documents that are marked as relevant by the user at the first turn of the retrieval process are used by the system for query modification for the next turn of retrieval which is expected to produce more satisfactory outcome [Salton 1989].

The evaluation criterion for the retrieval of formatted data focuses on the efficiency concern since the retrieved objects are clearly identifiable. In contrast, document retrieval process is concerned with effectiveness and efficiency where recall and precision are used as the measures of effectiveness*. Both criteria are important since users are interested in getting as many relevant documents as they can (recall) without being overwhelmed by numerous irrelevant documents that might also be returned. As for the efficiency, the minimization of the response time coupled with an acceptable storage overhead becomes an issue and various file organization schemes are proposed to enable faster access to data without creating too much space overhead [Can 1993a; Can 1993b; Faloutsos 1985; Salton 1989; Van Rijsbergen 1979].

The complexity associated with unformatted databases increases as the database contents become more assorted to include multimedia data (image, graphics, sound, voice,

* Recall is the ratio of the number of relevant documents that are retrieved to the total number of relevant documents in the database and precision is the ratio of the number of relevant documents that are retrieved to the total number of retrieved documents.

etc.) [Ozkarahan and Can 1991]. The main difficulty is related to the indexing of the documents which contain different kinds of data. For example, most of the multimedia IR systems functionally differentiate between text and pictorial data and base the retrieval on text data by viewing the pictorial part as its attributes. Other IR systems, on the other hand, focus only on pictorial data. However, in may real life cases, the users attention is toward all relevant data regardless of the specific form and a system that will consider information contained in all parts of a document is required [Bordogna et al. 1990]. Consequently, an integrated access method that will enable easy retrieval is rigorously sought for.

Signature files can successfully be implemented as one such method that will provide access to documents composed of various kinds of data (as in multimedia applications). Besides, a substantial improvement in retrieval efficiency can be achieved for a modest storage overhead which is typically 10-15% of the original database [Faloutsos 1985]. Insertions are easier especially compared to inverted indexes [Faloutsos 1992]. The implementation is usually simple and even very large data files can be supported. Queries on parts of the words can also be handled [Faloutsos 1985]. Two weaknesses of signature files are the occurrence of false drops [Stiassny 1960] and the deterioration of performance with the increase in the size of the database. Below we provide a closer look at these two problems and in the following sections we discuss the proposed remedies.

Signatures are bit pattern representations of objects which might be documents, records or logical blocks which are defined as the parts of the stored data items. (Throughout the paper, we will use the word "term" to indicate a key word in a document, an attribute of a record, picture, pattern, etc.) Each term within an object is hashed to a bit pattern of usually fixed length to create the term signature. Next the individual term signatures belonging to one object are combined (concatenated, superimposed etc.) to form the object signature.

object	signature	generation

terms	term signatures	
object	1000 1000	
signature	0010 0100	
generation	1000 1000	
	1010 1100	<= logical block signature

query	query signature	result
database	1100 0000	no match
generation	1000 1000	true match
information	1010 0000	false match

Figure 2. Signature extraction.

Figure 2 depicts one such example that uses superimposition and assumes that the objects in the database are grouped into logical blocks. For one such logical block that contains the terms "object," "signature" and "generation," the hashing function maps each term to two (not necessarily distinct) bit positions which are to be set to 1. The logical block signature is created by superimposing (ORing) the term signatures. A query that searches for the term "generation" will have the signature 1000 1000. Comparison of the block signature against the query signature reveals that the block signature has 1s in all positions specified by the query (1st and 5th) and hence qualifies. Similarly, a query searching for the term "database" will produce the signature 1100 0000. However, no matches will be found this time since the second bit position in the block signature is not set to 1.

Due to the information loss that takes place during signature generation [Faloutsos and Christodoulakis 1987a], some object signatures seem to qualify the query whereas the objects themselves do not. Yet these objects are accessed since there is no way to detect a false drop in advance. For the above example, assume that the query signature for the term "information" is 1010 0000. When this signature is compared against the block signature of Figure 2, the block signature seems to qualify although the block itself does not include the search term. The main purpose of the signature generation methods suggested in the literature is to minimize the false drop probability, F_d, since it causes unnecessary disk accesses and an additional CPU time. An overview of some of these signature generation schemes will be provided in Section II and the applicability of the same ideas to multimedia databases is discussed in Section VII.

When the signatures are stored sequentially, the retrieval performance deteriorates severely as the database size increases since all signatures should be scanned upon query submission. This problem is a major concern of most research on signature files because very large database sizes are common in today's applications. As will be discussed in Sections III through VI, these attempts to enable efficient application of signature files to very large database sizes have been successful and numerous schemes providing different levels of trade-off among retrieval efficiency, storage overhead, ease of updating, applicability with specific computer architecture (von Neumann or parallel) have been proposed in the literature.

II. SIGNATURE GENERATION METHODS

The common concern of all signature generation schemes is to minimize the false drop probability without generating too much space overhead. Also in all methods, terms are hashed into bit patterns which are later combined to form the object signatures. We will

provide an overview of each signature generation method and mention specific studies concerned with the application of the basics of these schemes with some (if any) variations.

II.1. Superimposed Coding (SC)

The objects in the database are grouped into logical blocks. Each nontrivial term is hashed to a bit-string of fixed length to form the term signature. Term signatures for a block are then superimposed to form the block signature [Orosz and Takacz 1956; Stiassny 1960]. Similarly, a query signature is created by ORing the individual query term signatures. A block qualifies a query if all bit positions that are set in the query signature are also set in the block signature. Our previous example in Figure 2 depicts the use of SC for a hypothetical logical block with three terms where the bit-string is of length 8 and each term sets 2 bits.

Signatures of each n-letter part of the words (n-grams) can also be generated and superimposed to allow search on parts of words. When this approach is used, a user searching for "Joe **Tan** & **Son** Co.," for instance, might use the terms "Tan" and "Son" in the query. Since the set of records that are returned to the user is independent of the order in which the key values are specified in the query, the set of returned records will include the ones belonging to this company as well as those related to a "**Son** and **Tan**nenbaum Co.," assuming that such a record exists [Roberts 1979]. Retrieval of such irrelevant records can be eliminated by imposing an order dependence constraint which will mark a record as a false drop if the desired order is not followed.

The SC applications can be classified into two groups based on the way the logical blocks are created. Faloutsos and Christodoulakis suggest that each block should have the same number of unique terms after stop word and duplicate removal [Faloutsos and Christodoulakis 1985; Faloutsos 1988a]. This approach is called the fixed-size block (FSB) method [Leng and Lee, D. L. 1992]. A more recent approach is called the fixed-weight block (FWB) method where the number of the terms in a block is allowed to vary but the weight of each block signature (the number of bit positions set to 1) is controlled to a constant [Leng and Lee, D. L. 1992].

An early study using the FSB approach indicates that the optimal number of 1s set by a term, m_{opt}, can be computed as

$$m_{opt} = \frac{F\ln 2}{D}$$

where F is the signature size and D is the average number of distinct, noncommon terms in a logical block [Faloutsos 1985]. This is a crude way to compute the optimal assignment strategy that minimizes the false drop probability, F_d, since the term occurrence and query frequencies are neglected and only single term queries are considered. We will name this scheme as single m (SM) to denote that all terms set the same number of bits.

Later work attempts to account for the differences in the term occurrence and query frequencies [Faloutsos and Christodoulakis 1985]. The approach is based on the observation that the terms with lower database occurrence frequency are specified more frequently in the queries. Such terms are said to have high discriminatory power in the sense that they efficiently determine those documents that are most relevant to the queries. Since terms with high discriminatory power are more important, they should be given the privilege to set relatively more number of bits in their associated term signatures. A mapping strategy that allows terms with high discriminatory power to set more bits is expected to reduce the probability of a false drop. If terms are grouped into n_s disjoint sets based on this criteria, the number of bits set by a term in set i, m_i, can be computed as

$$m_i = \frac{F \ln 2}{D} + \frac{1}{\ln 2}\left[\ln\frac{q_i}{D_i} - \frac{\sum_{i=1}^{ns} D_i \ln\frac{q_i}{D_i}}{D}\right]$$

where

$$\sum_{i=1}^{ns} q_i = 1 \quad \text{and} \quad \sum_{i=1}^{ns} D_i = D$$

and D_i is the average number of terms in a block that are from set i and q_i is the probability that the query term is from set i [Faloutsos and Christodoulakis 1985]. This formula, however, is based on the assumption that only single term queries exist and hence gives suboptimal solutions for multiterm query environments. We will call this scheme as Multiple m based on Single term queries or MMS for short.

A new method to find the optimal assignment strategy which considers both single and multiterm queries is also proposed [Faloutsos 1987; Faloutsos 1988a]. This method will be referred to as Multiple m based on Multiterm queries or MMM for short. In fact two solutions are suggested, one based on a complicated algorithm giving an exact result and the other being an approximate one enabling a closed form representation. Using the closed formula, the optimal number of bits set by a term in set i can be computed as

$$m_i = \frac{F \ln 2}{D} + \frac{\sum_{i=1}^{ns} D_i L_i}{D \ln 2} - \frac{L_i}{\ln 2}$$

where

$$L_i = \ln\left[\frac{P_i(0)}{P_i(1)} D_i\right]$$

and $P_i(k)$ is the probability that exactly k terms will be specified from set i. However, this solution can be used only if

- m_i values are large ($m_i > 4$, for example)
- $P_i(0) \neq 0$, $P_i(1) \neq 0$ and they are of the same order of magnitude [Faloutsos 1987].

Then the approximate false drop probability is shown to be computed using

$$\ln F_d = \ln \frac{P_{null} D}{1 - P_{null}} - \frac{F (\ln 2)^2}{D} - \frac{\sum_{i=1}^{ns} D_i L_i}{D}$$

where P_{null} is the probability of a null query (retrieve all records).

Table I. Exact and Approximate Values for m_is and F_d (taken from [Faloutsos 1987]

Signature Size (F)	m_1	m_2	F_d	approx. F_d	% error
200	11.62	2.08	0.00343853	0.0027895	18.875
	12.11	2.05	0.00344999	0.0027895	19.145
250	12.43	2.72	0.00203942	0.00177288	13.069
	12.76	2.70	0.00204253	0.00177288	13.202
300	13.19	3.37	0.00123672	0.00112676	8.891
	13.41	3.35	0.00123753	0.00112676	8.951
350	13.92	4.02	0.000761838	0.000716119	6.001
	14.07	4.01	0.000762043	0.000716119	6.026
400	14.62	4.67	0.000474386	0.000455132	4.059
	14.72	4.66	0.000474436	0.000455132	4.069
450	15.31	5.32	0.000297527	0.000289261	2.778
	15.37	5.32	0.000297538	0.000289261	2.782

The m_i values together with exact and approximate false drop probabilities are computed by both the approximate and exact methods for various signature sizes and presented in [Faloutsos 1987; Faloutsos 1988a]. The results indicate that the accuracy of the approximation improves with increasing values of F. The approximate and exact values for m_is and F_d are provided in Table I. where

$D_1 = 3$	$D_2 = 50$		
$P_1(0) = 0.1$	$P_1(1) = 0.8$	$P_1(2) = 0.1$	and
$P_2(0) = 0.8$	$P_2(1) = 0.1$	$P_2(2) = 0.1$	

and the first row for each signature size shows the exact m_i values and the second row gives the approximate ones. The last column shows the % error which is computed as $((F_d\text{-approx. } F_d)/F_d)^* \, 100$.

Another study uses the information theory to find the optimal assignment method for uniform and non uniform frequency cases [Faloutsos and Christodoulakis 1987a]. It also analyzes the relationship between the false drop probability and information loss. The results show that for the uniform occurrence and query frequency case, assigning the same number of words to each signature is the optimal matching strategy. For the non uniform case, no apparent dependency between false drop probability and loss is found. Advocators of the FWB method criticize this study on information loss because it assumes that each text block contains a single term and its optimal solution is based on a lookup table (that needs to be stored and updated to reflect the database changes) and not on hashing. Also for the non uniform frequency case only a suboptimal algorithm is suggested [Leng and Lee, D. L. 1992].

In general, the FWB approach is claimed to be superior to the FSB approach because the logical blocks are created in such a way that the weight of each block signature is controlled to a constant whereas in the FSB method only the expected weight of a block signature is kept constant. The FWB approach also enables simpler analysis without the need to make unrealistic assumptions. For instance, since the weight of a block signature is a constant, it can be directly used in the expression for the false drop probability [Leng and Lee, D. L. 1992].

More specifically, the FWB method can be implemented by generating term signatures one by one and superimposing them into block signatures in a step-by-step fashion until the weight of the block signature is greater than or equal to a constant. Note that even if the weight is above the value of the constant, the deviation is slight. The resulting blocks do not necessarily contain the same number of terms.

The optimal weight assignment scheme for FWB which is based on single term queries accounts for non uniform frequencies, avoids the use of lookup tables and minimizes the false drop probability. Expected false drop probability , E_x, can be represented as

$$E_x = \sum_{j=1}^{V} \sum_{i=1}^{p_j} \frac{\binom{W_{Bi,j}}{W_{Qj}}}{\binom{m}{W_{Qj}}} P(Q_j)$$

where V is the vocabulary size that includes all terms in the database, p_j is the number of blocks that do not contain term j, $P(Q_j)$ is the query probability of term j and m is the

signature size [Leng and Lee, D. L. 1992]. Additionally, $\{B_{i,j} \mid i = 1, 2, \ldots, p_j\}$ is the set of blocks that do not contain term j, $W_{Bi,j}$ is the weight of one block signature and W_{Qj} is the weight of the query signature with term j . Note that all block signatures have the same weight and hence
$W_{B1,j} = W_{B2,j} = \ldots = W_{Bavg}$ holds for $1 \leq j \leq V$.

The problem is to minimize

$$E_x = \sum_{j=1}^{V} p_j \left(\frac{W_{Bavg}}{m}\right)^{W_{Qj}} P(Q_j)$$

such that

$$\sum_{j=1}^{V} f_j W_{Qj} = C \quad \text{and} \quad \sum_{j=1}^{V} P(Q_j) = 1$$

where f_j is the number of blocks that contain term j and C is the fixed storage overhead. The optimal assignment follows [Leng and Lee, D. L. 1992]

$$W_{Qj} = \frac{C}{\sum_{j=1}^{t} f_j} + \sum_{j=1}^{t} \frac{f_j}{\sum_{j=1}^{t} f_j} \log_d \frac{p_j P(Q_j)}{f_j} + \log_d \frac{f_j}{p_j P(Q_j)}$$

where t is the vocabulary size excluding the terms with $P(Q_j) = 0$ or $p_j = 0$ and $d = W_{Bavg}/m$.

Comparison of the FSB and the FWB assignments show that FWB maintains lower false drop probability. This is because, in the FSB method, there exists a non zero probability that a signature with many 1s will be generated whereas this case is eliminated by FWB. Especially when the term signatures have non uniform weights, weight of each block can vary substantially for the FSB approach, resulting in high false drop probability. Nevertheless, the storage overhead of FWB is larger when the average block weights of the two approaches are the same [Leng and Lee, D. L. 1992].

II.2. Word Signatures (WS)

Words of an object are hashed to bit patterns which are later concatenated to form the object signature. Query processing occurs in the usual way: for a single term query, for instance, the query signature is generated and compared against all document signatures to find the qualifying ones [Faloutsos and Christodoulakis 1984; Faloutsos 1985]. Words in the query signatures are hashed in the same way. The text line signatures including the query signature are retrieved. This method enables search for parts of the words where information sequencing is also maintained. Figure 3 shows an example of WS extraction for a segment of an object that contains three words.

WS do not result in a very good false drop probability compared to other methods. They perform well when used with objects defined by variable number of descriptors, if variable object signatures are allowed. Also they are known to be the only method to preserve information sequencing [Tiberio and Zezula 1991].

	hypertext	hypermedia	applications
word signature	0010	0110	1100
object signature		001001101100	

Figure 3. WS extraction.

WS can be used for formatted databases to facilitate partial-match (i.e., multiattribute) retrieval. The common problem of the simple partial-match retrieval schemes based on pure hashing is concerned with large key spaces. For the case of a static file which has 2^d pages (where $d \geq 0$ and integer), and k hashing functions, one for each field, where the i^{th} function (h_i), $1 \leq i \leq k$, maps the values from the key space of field f_i to the strings of length d_i (such that $d_1+d_2+ \ldots +d_k = d$), a particular field f_j which has a key space of 2^{c_j} creates retrieval problems if $c_j >> d_j$. This is because each value field f_j cannot create unique patterns and hence the hashing function h_j yields the same bit string for many f_j values. Due to this information loss, the pure hashing scheme creates unnecessary accesses of many irrelevant pages.

The use of WS as w bit descriptors of the records solves the problem where each field f_i is mapped to a bit string of length w_i such that $w_1+w_2+ \ldots +w_k = w$ [Ramamohanarao and Lloyd 1983]. The descriptor of a page is obtained by ORing the individual record signatures on that page and only these page descriptors are stored. A query descriptor is compared against the page descriptor (signature) to determine whether that page should be accessed.

It is also possible to improve the performance of partial-match retrieval by extending the above scheme to dynamic files, e.g., using Linear Hashing (LH) [Litwin 1980; Ramamohanarao and Lloyd 1983]. The descriptor file is also allowed to expand and shrink in accordance with the size of the LH file. When a page split occurs in the LH file, a corresponding split is initiated in the descriptor file and a new descriptor is created for the new page. Although page descriptors have to be updated when records are added or deleted, which result in additional disk accesses, the scheme justifies itself since the reduction in the total cost of answering queries is significant. Besides, since query

submissions are more frequent than database updates in most cases, the gains in query processing supersede the efficiency loss resulting from the extra disk accesses during the updates.

Notice that using WS in partial-match retrieval applications enables one to emphasize the high priority fields on which many queries are based, by allocating relatively large bit segments to them. This, in return, leads to a decrease in the false drop probability which can be further reduced by increasing the size of the record signature and/or by using appropriate hashing functions that will evenly distribute the values of the fields over the associated bit segments. The problem of finding the optimal number of bits assigned to each field is addressed in [Moran 1983]. Moran attempts to design an optimal partial-match retrieval system for an environment where each record consists of a list of attributes that are hashed to bit strings which are later concatenated to find the address of the bucket in which the record will be stored. The purpose of the study is to find the optimal number of bits set by each attribute so that the expected number of buckets retrieved per query will be minimized [Moran 1983]. The problem is shown to be NP-hard [Garey and Johnson 1979] and two heuristic algorithms are proposed neither of which is shown to be strictly better than the other.

An indexing scheme that aims to combine the virtues of WS with those of the inverted indexes has also been proposed [Burkowski 1990]. The goal is the minimization of the time to scan the database contents upon query submission and the accomplishment of easy update capability. The signature of a word is called a marker which is different from a word signature in the sense that it is generated by using an assignment strategy (instead of hashing) which guarantees uniqueness and avoids the occurrence of false drops. The marker file is divided into a large number of subsets. During the creation of this file, each marker is assigned to a subset that will be the one that stores the marker group. The marker values are unique within a group. A database dictionary which depicts the corresponding subsets of each marker value is kept. During query processing, this mapping is used to find the subset to be scanned, given the marker of the query term. The addresses of the documents containing the query term can be found next to the marker of the word. Each subset is followed by some free space to allow for expansion [Burkowski 1990]. Application of Zipf's law (which states that a few instances occur most of the time and most instances occur very seldom) [Knuth 1975; Zipf 1949] to the occurrence frequency of the terms is used to determine the free space assignment strategy. Predictions of the free space requirements of the subsets are based on the nature of the portion of the database that is initially loaded. The objective is to minimize the occurrence of the overflows. The indexing scheme that uses WS provides fast retrieval and good space utilization. The performance is evaluated as competitive to that of the inverted indexes. Additionally, updating is faster and expansion is easier.

II.3. Compression Based Signature Generation Methods

Other signature generation methods based on compression whose special features make them appropriate structures for text retrieval for the automatic message filing systems are proposed in [Faloutsos and Christodoulakis 1987b]. The environment is characterized by its dynamic nature, large database sizes, high insertion rates, low deletion and update frequencies. Also most messages are rarely retrieved once they are filed, the access frequency decreasing sharply with the age of the database item. Signatures for such an office environment can be stored sequentially and the messages can be separated into non overlapping files to create message files of manageable size so that the retrieval efficiency of the sequential signatures will not be impaired. We now present an overview of three specific methods suitable for the structure and environment described above.

II.3.1 Run Length Encoding (RL)

The objects (messages for an office environment) are divided into logical blocks as in SC. However, the signature size, F, is very large compared to SC and each term (word) is allowed to set one bit only. The resulting signatures are sparse, enabling compression. An example for this method is given in Figure 4 where the signatures of three words are superimposed to form the block signature and the L_i values, which represent the displacements between two consecutive bit positions that are set to 1, are determined and stored. The representation $[L_i]$ stands for the encoded value of length L_i.

word	**signatures**				
run	0000	0000	0000	0000	1000
length	0010	0000	0000	0000	0000
encoding	0000	0010	0000	0000	0000
Block Signature	0010	0010	0000	0000	1000
	L_1 L_2		L_3		L_4

$$\Downarrow$$

$[L_1]$ $[L_2]$ $[L_3]$ $[L_4]$

Figure 4. Run length encoding.

RL provides excellent compression but the searching is slow since the encoded lengths of all the preceding intervals (runs) have to be decoded and summed up to detect whether a bit is set to 1 or not. On the average, half of the runs are decomposed if the bits in a query

signature are set based on a uniform distribution [Faloutsos and Christodoulakis 1987b]. The false drop probability ($F_{d,RL}$) is shown to be computed as

$$\log_2 F_{d,RL} = 1.528n - \frac{F_{RL}}{D}$$

where n is the number of bits a word sets to 1, F_{RL} is the signature size for RL and D is the number of distinct noncommon words in a block [Faloutsos 1985].

II.3.2 Bit-block Compression (BC)

To increase the searching speed of the RL method, BC divides the sparse block signatures into bit blocks of size b-bits, which are disjoint groups composed of consecutive bits of the block signature. Then for each bit block, a variable length signature, which consists of at most three parts, is constructed. Part 1 is zero if a bit block consists of all zeros, and equals one otherwise. Part 2 shows the number of 1s in the bit block followed by a terminating zero. Part 3 shows the offsets of the 1s from the beginning of the bit-block, where $\log_2 b$ bits are used for each 1. Figure 5 shows how the block signature of Figure 4 is compressed using the BC method when a bit-block size of 4 is used. Two representations of the compressed signature pertain to two different storage methods, one based on the concatenation of the bit-block signatures and the other based on the concatenation of the parts. The false drop probability ($F_{d,BC}$) is shown to be computed as

$$\log_2 F_{d,BC} = 1.913n - \frac{F_{BC}}{D}$$

where n is the number of bits a word sets to 1, F_{BC} is the signature size for BC and D is the number of distinct noncommon words in a block [Faloutsos 1985].

Block Signature	0010	0010	0000	0000	1000
Part 1	1	1	0	0	1
Part 2	0	0			0
Part 3	10	10			00

Storing by concatenating parts	1 1 0 0 1 ǀ 0 0 0 ǀ 10 10 00
Storing by concatenating bit-block signatures	1 0 10 ǀ 1 0 10 ǀ 0 ǀ 0 ǀ 1 0 00

Figure 5. Bit-block compression.

III.3.3 Variable Bit-block Compression (VBC)

VBC is the modified version of BC which uses an optimal bit-block size (b_{opt}) for each message based on the number of bits set to 1 in its sparse signature which has a fixed size of F for all messages. VBC aims to accomplish insensitivity to the changes in the number of words per block which will eliminate the need to split objects (messages) into logical blocks. Using VBC, the sizes of Parts 1, 2, 3 will depend on the size of the message itself. For messages of small size, for instance, where the number of distinct words per block, w, is also small, b_{opt} value will be large since it is shown to be computed as (Fln2/w). When b_{opt} is large, the number of bit-blocks gets smaller as well as the size of part 1 which equals to the number of bit-blocks. Part 2, which is of size w gets shorter. Part 3, on the other hand, shows fewer but longer offsets since each 1 is denoted by $\log_2 b$ bits.

II.4. Comparison of Signature Generation Methods

Below we provide a comparison of the SC and WS methods followed by a more general discussion on the performance of all methods discussed above,

II.4.1 Evaluation of SC against WS

A comparison of SC versus WS is provided in [Faloutsos and Christodoulakis 1984]. The research problem is to find which method gives smaller false drop probability for the same space overhead. In both methods, a document is split into logical blocks and F_d is computed for each block. The analysis is based on unsuccessful search case only which is shown to be sufficient to simulate the behavior in both successful and unsuccessful searches. For the WS method, F_d is given by

$$F_{d,WS} = 1 - \left[1 - \frac{1}{S_{max}}\right]^{D_{bl}}$$

where S_{max} is the maximum possible number of distinct word signatures and D_{bl} is the number of distinct, noncommon words per logical block.

This equation holds for arbitrary occurrence and query frequency distribution of the words as long as a_j values (which indicate the number of blocks that j^{th} word appears in) are small and the size of the database is large enough. It is interesting to note that $F_{d,WS}$ depends neither on the vocabulary size nor the database size and is not affected by the word interdependencies.

False drop probability for SC, on the other hand, is expressed as

$$F_{d,SC} = \left(\frac{1}{2}\right)^{m_{opt}} \quad \text{where} \quad m_{opt} = \frac{F \ln 2}{D_{bl}}$$

which is independent of the vocabulary and the database size and the occurrence and query frequencies of the words, provided that the above assumptions hold. Also the best performance is shown to be achieved when 50% of the bits in a signature are set to 1.

Comparative results indicate that both $F_{d,WS}$ and $F_{d,SC}$ are almost linear with the signature size F. When the size of a logical block, D_{bl}, is fixed and F is allowed to vary,

SC performs better for small signatures whereas the improvement in WS for larger signature sizes is faster. When for a constant space overhead D_{bl} is allowed to vary, the results indicate that SC supersedes WS more and more with increasing D_{bl}, if the overhead is small. For larger overhead, WS outperforms SC.

Another study comparing the signature extraction methods from various aspects acknowledges the advantages of WS and SC for partial-match queries and praises WS for preserving the sequencing of words [Tiberio and Zezula 1991]. SC allows for automatic elimination of duplicates whereas in WS, sorting should be used for the same purpose [Christodoulakis and Faloutsos 1984]. The disadvantages of WS are listed as the inability to handle queries on parts of words and on numeric fields both of which can be done with SC. WS is classified as inefficient when comparing two signatures for qualification for which SC is highly efficient [Tiberio and Zezula 1991].

II.4.2 Comparison of SC and WS Methods with RL, BC and VBC

WS and SC methods can in fact be viewed as special cases of the BC method [Faloutsos and Christodoulakis 1987b]. When the number of bits set by each term equals to m (where m = Fln2/D) instead of 1 and the sparse vector is not compressed, i.e., b = 1, BC converges to SC. On the other hand, when the number of bits set by each term is 1, b equals to the signature size and Part 3 contains the offsets of the 1s from the beginning of the signature, WS representation is obtained, the only difference being the order of the offsets which are not necessarily ascending for WS.

RL accomplishes the least false drop probability followed by BC and VBC both of which outperform SC and WS in terms of achievable false drop probability for a given signature size [Christodoulakis and Faloutsos 1986]. The results concerning the number of bit comparisons required to search a signature, which is used to reflect the CPU time, show that SC supersedes all other methods when the vocabulary size of the documents have a slight variance. In other cases, VBC requires the least CPU time. Since RL method requires decoding and adding of approximately half of the intervals it requires a longer search time. WS also requires the examination of the whole block signature but is relatively faster since no decoding is performed. VBC yields outstanding performance for documents spanning many logical blocks, for objects of variable length and queries that refer to many terms [Christodoulakis and Faloutsos 1986; Faloutsos and Christodoulakis 1987b]. In spite of these cases, SC is preferable. SC can also handle queries on parts of the words (by using the n-gram approach) and on numeric fields. Besides, SC can be used in various signature file organizations like bit-slice, S-Tree and hashed schemes whereas the other methods (except BC which can be used in bit-slice) work with sequential organizations only [Tiberio and Zezula 1991]. This provides SC a substantial advantage over the other methods since a large portion of the signature applications are based on organizations other than sequential, which provide reduction of search space and faster access to data. (Refer to Sections III to V.)

II.5. Application of Probability Theory to Problems of Signature Generation

In an early study, the combinatorial and probability theory has been used to address some of the basic issues of signature generation [Orosz and Takacz 1956]. The analysis is based on a signature generation scheme which uses a mixed model based on WS and SC where a signature of size F is composed of p segments of size F_i, such that $F_1 + F_2 + \ldots + F_p = F$. Each word is assumed to set v_i bits in segment F_i such that the vocabulary size can be computed as

$$\prod_{i=1}^{p} \binom{F_i}{v_i} = V(F,v)$$

Exact formulas for the distribution of the number of bits set when N such signatures are superimposed and for the distribution of the multiple marking of a bit position are provided. A mathematical analysis of the superimposed coding method is also provided in [Stiassny 1960] where the computation of the false drop probability and the optimal number of bits set by each term is analyzed.

A more recent study on the distribution of the number of ones in the final signature after D_{bl} distinct term signatures are superimposed is based on the SM method (see Section II.1) where each term sets m number of bits regardless of its discriminatory power [Christodoulakis and Faloutsos 1984]. This problem is modeled as selecting $m*D_{bl}$ bits from a set of F bits with replacement where F is the signature size and D_{bl} is the number of distinct noncommon terms in a block. The probability distribution of the number of 1s after the selection of $m*D_{bl}$ bits is expressed as the state probability vector P_{m*Dbl} where

$$P_{m*Dbl} = (P(0, m*D_{bl}), P(1, m*Dbl), \ldots, P(F, m*D_{bl}))$$
$$P_{m*Dbl} = T^{m*Dbl} * P_0$$

where

$P(k, i)$: the probability that k bits are set after i random bit selections

P_0: the vector $(1, 0, 0, \ldots, 0)$ consisting of $(F+1)$ elements and T is the state transition matrix represented as

$$T = \begin{bmatrix} 0 & 1 & 0 & . & . & . & 0 \\ 0 & \frac{1}{F} & \frac{F-1}{F} & . & . & . & 0 \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & \frac{F}{F} \end{bmatrix}$$

Since the above expression is somewhat complex and difficult to manipulate, a simplified version for the expected number of bits after the selection of $m*D_{bl}$ bits with replacement, M, is also proposed as

$$M = F\left[1 - \left(1 - \frac{1}{F}\right)^{m*D_{bl}}\right]$$

This closed formula is then used to come up with an approximate formulation for the false drop probability represented as

$$F_d = \left(\frac{M}{F}\right)^m = \left[1 - \left(1 - \frac{1}{F}\right)^{m*D_{bl}}\right]^m$$

The above approximation can be justified intuitively by noting that the probability that a bit is set to 1 is 1/F. Then (1-1/F) represents the probability that a bit is set to 1 and (1-1/F) raised to the power $m*D_{bl}$ stands for the probability that a bit is not set to 1 by any of the $m*D_{bl}$ bit setting trials. If we call this probability P, then (1-P) is the probability that one of the m bit positions set to 1 by a term has already been set by an other term and $(1-P)^m$ indicates the probability that all selected m positions have already been set causing a false drop to occur. This approximation has been proven to give very close results to the ones obtained from the exact formulation but it can only be used for the cases where all terms set the same number of bits (m). A more general expression for the probability distribution of the number of bits set in the final signature when k term signatures each setting m_k number of bits (where m_ks are not necessarily equal) is also required to analyze the MMS and MMM cases (see Section II.1 for their definitions).

From another point of view, the problem can be reformulated as finding the distribution of the query weight, W(Q), i.e., the number of 1s in the query, when k terms are specified in a query, each setting m_k number of bits, where k is a random variable whose distribution is determined by the query characteristics of the system of concern. This problem is addressed in a recent study where the superimposition of the k term signatures to form the final signature is viewed as a k stage process [Murphree and Aktug 1992]. The number of

1s in the query signature after the completion of stage i is represented by Y_i, where $m_1 = Y_1 \leq Y_2 \leq \ldots \leq Y_k = W(Q)$. The values of $Y_1, Y_2, \ldots, Y_k$ determine the value of W(Q) and

$$P\{Y_r = a \mid Y_{r-1} = b, Y_{r-2} = c, \ldots, Y_1 = m_1\} = P\{Y_r = a \mid Y_{r-1} = b\}.$$

Noting that the random variables $Y_1, Y_2, \ldots, Y_k$ form a Markov Chain, one-step transition probabilities $P\{Y_r = a \mid Y_{r-1} = b\}$ for $r = 2, 3, \ldots, k$ are taken into account to find the distribution of $W(Q) = Y_k$ where $m_1 \leq b \leq a \leq F$ [Feller 1968]. These probabilities can be expressed as $P\{Y_r = m_1 + j \mid Y_{r-1} = m_1 + i\}$ since minimum value for both Y_r and Y_{r-1} is m_1.

P_r is an $(F-m_1+1)$ by $(F-m_1+1)$ matrix consisting of the one-step transition probabilities where the entry at the intersection of the ith row and jth column is $P\{Y_r = m_1 + j \mid Y_{r-1} = m_1 + i\}$ and all entries below the main diagonal are equal to zero. The authors show that $P\{W(Q) = s+m_1 \mid Y_1 = m_1\} = P\{Y_s = s+m_1 \mid Y_1 = m_1\}$ is given by the (0, s) entry in $P_2 P_3 \ldots P_k$, where s stands for the number of additional bit positions that can be set to 1 after the first stage. Through some matrix manipulations which make use of the fortunate fact that all P_is have the same set of eigen vectors, the conditional probability that the query weight will take a value $s+m_1$, $P\{W(Q) = s+m_1 \mid m_1\}$, is expressed as

$$\sum_{j=0}^{\min(F-m_1,\, s)} \binom{F-m_1}{j} \binom{F-m_1-j}{s-j} (-1)^{s+j} \prod_{r=2}^{k} \frac{\binom{m_1+j}{m_r}}{\binom{F}{m_r}}$$

and

$$P\{W(Q) = w\} = \sum_{m_1=0}^{w} f(m_1)\, P\{W(Q) = (w-m_1) + m_1 \mid m_1\}$$

where $f(m_1) = P\{Y_1 = m_1\}$.

A recent study that evaluates the performance of SM, MMS or MMM methods as they are applied to a dynamic signature partitioning methods in a multiterm query environment uses the derivations above to determine the distribution of the query weight [Aktug and Can 1993b]. A discussion of the findings of this study is provided in Section V.3.3.

III. SINGLE LEVEL SIGNATURE FILE ORGANIZATION METHODS

Several signature file organization schemes have been proposed in the literature, providing gains in retrieval speed, space utilization, ease of insertion/deletion, ease of use with certain hardware architecture, etc. The simplest structures are called single-level organizations

where all individual signatures (at least parts of them) should be examined during retrieval. None of these signatures are combined to create super signatures or common key values.

III.1. Sequential Signatures (SS)

Sequential Signatures (SS) organization refers to a sequential file which consists of bit-string representation of fixed-length signature records. For the case of N signatures of length F bits, the SS representation can be shown by an NxF matrix. Figure 6.a provides an example for F = 8 and N = 10. Here the symbol S_i stands for the ith signature ($1 \leq i \leq 10$). Upon query submission, all signatures are searched sequentially. It is the simplest, easy-to-implement approach which facilitates exhaustive search and enables easy insertion [Faloutsos 1992; Tiberio and Zezula 1991; Zezula et al. 1991]. However, since the retrieval performance is proportional to the size of the database, response time becomes unacceptably high for large databases.

Sequential signatures are used as an access method for text in a message file system that enables retrieval of messages according to contents [Tsichritzis and Christodoulakis 1983]. The messages are organized in general files instead of complex directories to reduce the necessity for frequent reorganization. The system uses the filtering capability of signatures to improve the performance of the sequential scan and the authors claim that since most of the time users do not provide a tight description of what they are looking for and expect to see some irrelevant messages in addition to the relevant ones, the false drops resulting from using the signature approach will not be much of their concern. If a user's expectation of irrelevant messages is, say 10%, he/she will not be much overwhelmed by another 0.5% that comes with the false drops. Naturally, false drop resolution techniques are available but they may or may not be implemented depending on preference.

A new message is appended at the end of the sequential file which stores all the messages in the database. In addition, a physical file corresponds to each logical file the user requires the message to be filed in, where the descriptor of the message (signature) is stored together with a pointer pointing to the location where the message itself is stored. This structure facilitates the overall organization of the message by enabling a message to be grouped based on all different logical files that it might relate to. Besides, since only signatures rather than multiple copies of the message are kept, the flexibility of the structure is achieved without too much storage overhead.

The design issues for such a message server facility for the office information system environment are provided in [Christodoulakis and Faloutsos 1984]. The organization consists of a two level hierarchy where the first level (access level) is made up of the sequential signatures that provide a filtering capability to limit the search space and the second level (storage level) includes the collection of the messages. A message m can be represented by a vector $(a_0, a_1, \ldots, a_n, b)$ where $a_0, a_1, \ldots, a_n$ are the attribute values of the attributes of the header and b corresponds to the body of the message which consists of text data. The corresponding signature for this message, S(m), is represented as (t, $S(a_0)$, $S(a_1)$, . . . , $S(a_n)$, S(b)) where t shows the type of the message (for example, memo), $S(a_0)$, $S(a_1)$, . . . , $S(a_n)$ are the signatures of the attributes and S(b) is the signature of the text. Attribute signature generation is highly correlated with the domain of each attribute; attributes which can take fewer values result in smaller signatures [Christodoulakis and Faloutsos 1984].

A query specifies the type of the message, some of the attribute values (optional) and also some pattern of words. The system allows for specification of single words as well as pairs of words, sequencing of the words and the word parts. It can also handle fuzzy matching, like words with possible errors and complicated expressions containing conjunctions, disjunctions, or both. Upon query submission, the access (signature) file is scanned first. If the type of a message signature does not qualify, the signatures for the corresponding attributes and the body do not need to be checked. Only when both type and attribute signatures qualify, the signature of the body is compared against the query specifications.

The main emphasis of the above study is on the generation of the signatures of the bodies whose space requirements are significantly higher than that of the attribute signatures. The body b of a message m is divided into u logical blocks $(b_1, b_2, \ldots, b_u)$ using the FSB approach (see Section II.1) where each block consists of a fixed number of noncommon words. The signature of a block, $S(b_i)$ is generated by letting all the noncommon words in this block set m number of bits and the block signatures are superimposed to form the signature of the body. The signature size, F, and the number of bits set by each term, m, are design parameters whose optimal values are determined by using the block signatures rather than the message signatures as units since the use of the block signatures allow for a better choice of the values of the parameters. Had these values been based on an average size message file, the resulting performance would have dropped for messages with different sizes.

The total cost, C_{tot}, representing the total amount of accesses required to answer a single word query is computed as

$$C_{tot} = M_{bl}\left(\frac{F}{BF}\,(CS) + F_d\,(CT)\right)$$

where

M_{bl}: total number of blocks in the text file

D_{bl}: expected number of distinct common words in a block

BF: number of bits in a block of the access file

CS: cost of accessing a single block of the access file

CT: cost of accessing a single block of the storage file

and the false drop (Fd) is computed using the formula in Section II.5. The problem is to find the optimal value for m that minimizes this cost function. The results of the study indicate that equation ($m = F\ln 2/D_{bl}$) can be used to obtain an approximate solution. The resulting values of m are proven to be very close to the exact values which minimize the cost function [Christodoulakis and Faloutsos 1984].

It has also been examined that the access frequency of a message decreases exponentially with time and is also dependent on the type of the message. What is more, in some instances, only attribute values are used to determine whether a specific record should be retrieved or not without the need to access the signature of the body. These observations suggest that an organization based on the retrieval frequency of the signatures of the bodies will improve the system performance. The frequency of use of the signature of a logical block b_i, denoted by f_i, is assumed to decrease only from one reorganization of the messages to the other. At every reorganization point, the new signature $S'(b_i)$ is created by eliminating the last n bits of the signature in the previous reorganization, $S(b_i)$. The storage file remains intact but since the last n bits of $S(b_i)$ have been discarded, the cost of the sequential scan of the access file decreases. The optimal value for n for a block signature is shown to be computed as

$$n = \frac{r*F - M}{F - M}$$

where

$$r = \left\lceil \frac{a*F^2}{f*m*(F - M)} \right\rceil \qquad a = \frac{(CS)}{(CT)*(BF)}$$

and f is the frequency with which the block signature is accessed (= corresponding f_i value) and M is the expected number of ones in the block signature. As the frequency f decreases, the value of n increases together with the percent savings. Table II. shows the resulting percent savings (%sav) for particular values of CT/CS, F and m [Christodoulakis and Faloutsos 1984].

Table II. Percent Savings for Different Levels of f and n
(taken from [Christodoulakis and Faloutsos 1984])
CT/CS = 21, F = 578, m = 10

f	0.8	0.4	0.3	0.1	0.08	0.04	0.03	0.01	0.008	0.004	0.003	0.001
n	0	36	56	138	156	215	241	347	370	446	479	578
%sav	0	1.3	3.0	12.8	15.3	23.9	27.8	44.1	47.7	59.5	64.7	85.1

III.2. Vertical Partitioning

Since the sequential signature organization accesses every bit in every signature when processing a query, the response time gets very slow, especially for large databases. Vertical partitioning aims to alleviate this problem by accessing only those bits of the signatures that are set in the query signature.

III.2.1 Bit-sliced Signatures (BS)

The main idea is to store the signature matrix columnwise so that only k columns have to be accessed for a query with weight k, where k refers to the number of 1s in the query signature. For the case of N signatures of length F bits, the BS representation can be viewed as an FxN matrix where typically N>>F. The method is efficient for low query weights but the number of disk accesses increases with the query weight. Maintenance, on the other hand, is very costly and time consuming hence this organization is suggested for stable files, archives or for systems with typically low weight queries [Tiberio and Zezula 1991]. Figure 6.b shows how 10 signatures of length 8 can be organized using BS. For a query with signature 1010 0000, where k is 2, only two column accesses are necessary corresponding to the first and third bit positions. (Even in this simple example, the improvement in retrieval efficiency relative to SS can be observed.) The comparison of the first and third rows of the matrix in Figure 6.b reveals that the only signatures which have 1s in both the first and the third bit positions are S_3 and S_4.

S_1:	0001 1100	S_6:	1001 1000
S_2:	0110 0001	S_7:	0011 1000
S_3:	1010 0010	S_8:	0000 1110
S_4:	1010 0001	S_9:	1100 0010
S_5:	0010 0011	S_{10}:	0001 0011

Signatures

$$\begin{array}{c|c}
\Leftarrow F \Rightarrow & \\
\hline
0\,0\,0\,1\,1\,1\,0\,0 & \wedge \\
0\,1\,1\,0\,0\,0\,0\,1 & \| \\
1\,0\,1\,0\,0\,0\,1\,0 & \| \\
1\,0\,1\,0\,0\,0\,0\,1 & \| \\
0\,0\,1\,0\,0\,0\,1\,1 & \| \\
1\,0\,0\,1\,1\,0\,0\,0 & N \\
0\,0\,1\,1\,1\,0\,0\,0 & \| \\
0\,0\,0\,0\,1\,1\,1\,0 & \| \\
1\,1\,0\,0\,0\,0\,1\,0 & \| \\
0\,0\,0\,1\,0\,0\,1\,1 & \vee
\end{array}$$

Figure 6.a *SS Organization.*

$$\begin{array}{c|c}
 & \Leftarrow N \Rightarrow \\
\hline
\wedge & 0\,0\,1\,1\,0\,1\,0\,0\,1\,0 \\
\| & 0\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\
\| & 0\,1\,1\,1\,1\,0\,1\,0\,0\,0 \\
F & 1\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\
\| & 1\,0\,0\,0\,0\,1\,1\,1\,0\,0 \\
\| & 1\,0\,0\,0\,0\,0\,0\,1\,0\,0 \\
\| & 0\,0\,1\,0\,1\,0\,0\,1\,1\,1 \\
\vee & 0\,1\,0\,1\,1\,0\,0\,0\,0\,1
\end{array}$$

Figure 6.b *BS Organization.*

Figure 6. SS and BS organization examples.

In [Faloutsos and Chan 1988], three methods based on the BS organization (Compressed Bit Slices (CBS), Doubly Compressed Bit Slices (DCBS) and No False Drops (NFD)) are proposed where ease of insertion of signature files is combined with the fast retrieval of the inverted files. Of these three methods, CBS stores the position of the 1s to compress the bit-sliced signature file. The bit files are stored in buckets of size Bp (where Bp is a design parameter) and are linked to each other by pointers. A directory with F pointers is used, where F is the signature size, and each pointer corresponds to one bit slice. A hashing function maps each term to a bit slice. The set of all compressed bit files is called the postings file which contains pointers to the appropriate documents that contain the term. DCBS method modifies this structure by adding an intermediate file to it and attempting to distinguish between the synonyms by using a second hashing function. The NFD method, on the other hand, aims to eliminate all false drops by storing a pointer to the word in the text file.

All three methods require small overhead (20-30% of the original file), give fast responses and require no rewriting. They can work well on both magnetic and optical disks. Interested readers are referred to [Faloutsos and Chan 1988] for detailed performance evaluation formulas for these methods.

III.2.2 Frame-sliced Signatures (FS)

The underlying motivation of the method is to improve the virtues of the BS organization without sacrificing too much from insertion time and space overhead [Lin and Faloutsos 1992]. Since disk access time is dominated by the seek time, the method aims to reduce the number of random disk accesses. This new approach views the bit-slice for a signature as k frames of s bits each. To create a term signature, two hashing functions are used; the first one determining the frame the term is going to use, and the second function giving the m bit positions to be set by the term in that particular frame. Figure 7 provides an example for this method. When the signature matrix is stored frame-wise and each frame is stored in consecutive disk blocks, only one frame is accessed for a single word query and n for an n-word query [Lin and Faloutsos 1992].

Term	**Frame 1**	**Frame 2**	**Frame 3**
signature	0000	0000	1100
file	0000	1001	0000
organization	0000	1100	0000
Document signature	0000	1101	1100

Figure 7. FS generation: F=12, k=3, s=4, m=2.
The term "signature" is hashed to the third frame whereas the terms "file" and "organization" are hashed to the second frame. The document signature is formed by superimposing the term signatures.

FS requires a small overhead (12-15% of the original file) and no rewriting like other signature methods. No reorganization has to be performed upon insertion. FS is faster than BS on insertion. Further gains are also possible if a more generalized model, called Generalized Frame-sliced Signatures (GFS), is used. This time, a word is mapped into n distinct frames and is allowed to set m bits in each. Note that GFS converges to FS when n equals to 1 and to BS when both k and n are 1. GFS has been shown to outperform these two organizations which are in fact its special cases. It is suitable for magnetic disks, CD-ROMs, write-once optical disks (WORMs) and erasable optical disks, since it provides fast response and low space overhead [Lin and Faloutsos 1992].

III.3 A Hybrid Organization for Text Databases Using BS

The motivation to create an index over a large number of terms for a large number of documents has lead to the design of a hybrid index organization for text databases [Faloutsos and Jagadish 1992]. The objectives are minimization of the storage overhead of the index and the retrieval time. A secondary concern is the efficiency of updates for dynamic environments.

The study makes use of the Zipf's law which states that a few instances occur most of the time and most instances occur very seldom [Zipf 1949]. When applied to the index terms, this law suggests that assuming equal occurrence frequency for the index terms in the documents is not realistic because an imbalance is very likely to occur. Hence none of the indexing techniques (inverted indexes, signature files, etc.) alone will perform best in all situations. Therefore, a hybrid method that combines the advantages of each can most probably give better performance [Faloutsos and Jagadish 1992].

The new organization treats frequent terms in a different way. The traditional inverted index, which consists of the sorted list of the terms (usually represented as a B-tree [Tharp 1988]) and the postings file, is modified in such a way that the same structure is kept for rare terms only and the postings list is stored as a bit vector for the frequent terms. Use of a bit-slice representation connotes a signature file like approach. Changes to this basic structure is possible depending on the properties of the environment.

The results for both static and dynamic environments indicate that it is possible to achieve improvement in space, search and insertion time over the inverted index method. For dynamic environments, the hybrid technique is suggested to be modified so that it becomes closer to a signature file approach rather than an inverted index. This shift aims to take advantage of the superiority of the signature approach over inverted indexes for insertion time.

III.4. Document Ranking Using Single Level Signature File Organizations

Text data is dynamic (especially in terms of additions), variable in length and consists of a wide variety of tokens. Text retrieval methods have to cope with these undesirable features which lead to efficiency problems. Secondly, effectiveness turns out to be another issue since text data has poorly defined semantics and finding the match is not sufficient to retrieve the document. Signature files have been criticized to address only the efficiency problems and to neglect the effectiveness issues. The basic retrieval technique supported by the signature files is evaluated as weak because it does not handle the ranking of the documents.

The study reported in [Croft and Savino 1988] attempts to implement probabilistic ranking strategies for sequential and bit-slice organizations, with little cost reflected in efficiency. Variations of probabilistic ranking algorithms are discussed and it is concluded that a signature based implementation should at least take term weights into account which will bring a 10-50 % gain in precision. If term significance weights are also taken into account with the extra cost of storing the within-document frequencies (which indicate the frequency with which a term appears in a particular document), an additional 10-20% gain is also accomplished. The performance of each case is compared against the corresponding (sequential or inverted) term-based organization. The results indicate that for the same level of effectiveness, as for the sequential structures, term-based file is somewhat more efficient in I/O time and storage overhead. Signature organization is faster for short queries but gets slower for larger ones. For the inverted structures, the term-based file requires fewer I/O operations, gives faster response time and does not demand a large storage overhead [Croft and Savino 1988]. Note that though appreciated, the results of the study should not be overgeneralized to include all signature file schemes since the associated experiments are based only on single level signature file organizations. Effectiveness issues will be discussed more in Section V.2.

IV. MULTILEVEL SIGNATURE FILE ORGANIZATION METHODS

Multilevel organizations require the construction of one or more other signature levels (or an index) in addition to the single level signatures to establish a filtering mechanism that will limit the search space and improve retrieval efficiency. The following is a discussion of the major multilevel signature file organizations.

IV.1. Tree Structures

IV.1.1 Applications Using Signature Trees

In signature trees, individual signatures are divided into groups and signatures in each group are superimposed to form the super signature for the next higher level. Hence a signature tree is constructed from bottom to top. The process of creating an upper level continues until too few signatures are left to form a super signature. The number of signatures in a group can vary with the level of the tree. Besides, the signature size should be chosen large enough for trees with many levels [Tharp 1988]. Figure 8 indicates one example of a signature tree that can be used for formatted data where the Level 1 signatures are formed by the concatenation of three bit segments each of which corresponding to the signature of one field/attribute value (f_i). The size of the bit segments allocated to each field depends on the range of values and the relative importance of the contents field [Moran 1983]. When a query of the form 10000 00000 001 is submitted, the upper level signature is checked to see that it qualifies. Of the three signatures in the second level only the third one qualifies (marked with * in Figure 8) and hence it is sufficient to consider the lower level signatures on this branch only. Among these five signatures on Level 1, only the third one qualifies (marked with **).

```
                                                   f1     f2     f3
                                                  00010 001000 010
                                                  00010 100000 100
                        00110 101001 110          00100 100000 010
                                                  00100 000001 100
                                                  00100 001000 100

                                                  00010 100000 010
                                                  00100 010000 010
 10110 111001 111       00110 111001 011          00010 001000 001
                                                  00010 001000 010
                                                  00100 000001 001

                                                  10000 100000 100
                                                  10000 010000 100
                       *10100 111000 101        **10000 001000 001
                                                  00100 001000 001
                                                  00100 001000 100
     Level 3                Level 2                    Level 1
```

Figure 8. Signature tree.

As also demonstrated by the above example, signature trees provide a pruning feature which decreases the number of signatures searched for a query. The query signature is initially compared with the signature at the highest level of the tree and only the branches with the qualifying signatures are traced. The structure works well for queries with high weights since the pruning mechanism becomes more selective. If the signature size is not made large enough for the database size, the upper level signatures easily get cluttered and the performance of the organization is seriously degraded. Besides, grouping similar signatures decreases the amount of cluttering in the upper levels [Kotamari and Tharp 1990]. Also as opposed to the inverted indexes where the decision as to invert a term is

binary, signature trees provide more flexibility by allowing the assignment of different storage requirement to each attribute based on it associated weight. The resulting storage overhead is 5-40% for signature files whereas it is possible to have overhead percentages around and above 100% for the inverted files [Tharp 1988].

The same idea prevails in [Pfaltz et al. 1980] where upper levels of signatures (called the block descriptors) are created by superimposing a group of the lower level signatures that are assigned to a block. The number of signatures from level i that are superimposed to form the block descriptor at level (i+1), where ($i \geq 1$), is called the packing factor p(i) which is a design parameter and may vary for different levels of the tree. The structure is called indexed descriptor files.

Two file creation structures are proposed: In the "bottom-up" approach, the data records are sorted lexicographically on the subfields of their signatures and then the first p(0) signatures are selected to form the first block and the associated block descriptor is made to be the first entry (super signature) of level 1. Once all level 1 signatures are generated this way, the first p(1) signatures are superimposed to form the first entry of level 2 and so on. The new record signatures are just inserted in the last partially filled block and the associated block descriptor together with those upper level block descriptors that are affected by this insertion are modified. This bottom-up structure has a storage overhead cost which is less that 10% of the original file and can be used for static and moderately dynamic files [Pfaltz et al. 1980].

The second structure, called the "top-down" approach applies to more dynamic files. Here blocks are assumed to be partially filled and the signature of the new record is inserted to the "best" block possible. The purpose is to cluster the records to prevent rapid cluttering of the upper level block descriptors. The concept of "best" is most of the times a heuristic one that depends on multicriteria one of which might be the number of new bits set in the block descriptor with the inclusion of the new record signature. This structure results in a storage overhead cost of 20-40% of the data file [Pfaltz et al. 1980].

In search for a method which will provide fast retrieval, eliminate the need for periodic reorganization and hence be appropriate for dynamic environments, Deppish has proposed the S-tree organization [Deppish 1986]. His approach criticizes the bottom-up signature approach since during insertion, new signatures are inserted into the last partially filled node instead of the one which contains the similar signatures. Similar to the top-down approach, Deppish suggests inserting these signatures into the appropriate leaves to support clustering of the signatures since the superimposition of dissimilar signatures causes the upper level signatures to get cluttered easily. A heuristic splitting procedure keeps the adjacent signatures together and the tree balanced. Performance of the method ranges from very good to very bad as the query weight decreases and hence it is suggested as a complementary technique to BS. The major problem of the S-tree structure is the high space overhead. The organization is data dependent and the response time on queries is difficult to estimate analytically [Faloutsos 1992].

VI.1.2. Inverted Signature Tree

The Inverted Signature Tree (IST) structure which uses both signature trees and inverted lists is suggested in [Cooper and Tharp 1988]. This organization has an inverted list for each candidate search word which indicates the locations where a sentence containing the word begins. Signature trees, on the other hand, act as indexes for the corresponding inverted lists. The signature tree is constructed in the usual manner by superimposing the word signatures to create the super signatures in the higher levels. There exists a signature tree for each letter and next to the tree, there are the actual words for the letter as a safeguard against the false drops. The final component of the tree contains the CD-ROM location of the word's inverted list. The use of a signature tree index enables the search words to be stored alphabetically. Such an ordering can act as an aid to determine the correct form of a word during an on-line application. Besides, since the structure is proposed for the CD-ROM environment, the storage overhead is not a concern.

The same study also discusses B+ Trees [Tharp 1988] where information requiring both sequential and direct retrieval can be stored. A comparison of the response times of Inverted Signature Trees (IST), B+ Trees and text signatures in a CD-ROM medium reveals that all three structures are equally efficient for small files. For larger sizes of the database (to search an encyclopedia, for instance), text signatures fail since they yield an unacceptably long response time. The relative performance of the IST compared to that of the B+ Tree depends on whether the B+ Tree index can be stored in the primary memory. Nevertheless, the IST structure handles unnecessary searches faster, demands less primary memory and is easier to implement [Cooper and Tharp 1989].

IV.2. Two Level Signature Files

A two level scheme for signature file organization has been proposed in [Sacks-Davis 1985; Sacks-Davis and Ramamohanarao 1987]. The first level consists of the sequential organization of the record descriptors which are formed by superimposing the term signatures in a record. Then all N records in a file are allocated to N_s blocks, each containing N_r records such that $N = N_sN_r$ holds. By superimposing all term signatures in a block, regardless of the records these terms belong to, the block signatures are generated. These block descriptors are stored using the bit-slice representation. Note that the block descriptors are typically larger than the record descriptors and are characterized by different

values of the parameters for the signature size and the number of bit set by each term. (See Figure 9 for a hypothetical example of the organization where b_s and b_r are the sizes of block and record descriptors, respectively.) During query processing, a record descriptor as well as a block descriptor is formed for the query. However, only those record descriptors whose corresponding block descriptors qualify are compared against the query record descriptor.

The method performs well when the number of qualifying records per query is low, since the block descriptors then provide an exhaustive screening. However, when the number of such records increase, since both block and record descriptors have to be accessed for many cases, the efficiency of the organization drops below that of the one level scheme using record descriptors (signatures) only [Sacks-Davis and Ramamohanarao 1987]. Hence the performance of the method is data dependent.

Another inconvenience of the above scheme is that it gives rise to unsuccessful block matches for multiterm queries since within a qualifying block, the required terms can come from different records. An encoding scheme which makes use of the frequencies of the index terms to reduce the number of unsuccessful block matches has been proposed in [Sacks-Davis and Ramamohanarao 1987]. Here bits in the block descriptors are set for pairs of frequent terms in addition to the ones set for the single terms. Such bits are called the combination bits and they do not create much of a storage overhead since the number of bits set for a pair of common words will be less than the ones set by the single terms.

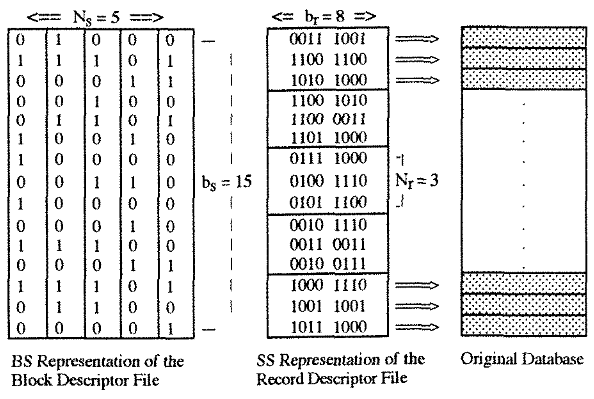


Figure 9. Two level signature file organization.

In the absence of such a scheme, another way to diminish the number of unsuccessful block matches is to have the capability to identify individual records at the first level of the index. A multiorganizational approach where block descriptors are generated in a manner to allow record identification has been proposed [Kent et al. 1990]. Instead of letting a term set k bits in a block descriptor, k different block descriptors are created each having a single bit set. These k block descriptors are stored in k block descriptor files. In contrast to the two level scheme where there is only one mapping function to assign records to the blocks (block no. = record no. div block size, where div indicates integer division), the multiorganizational scheme has possibly different mappings for each descriptor file. These mappings are called organizations and they are the keys to record identification [Kent et al. 1990].

The multilevel organization has been used to support document storage and retrieval in a nested relational database system [Zobel et al. 1991]. In general, the experimental results show that the multilevel organization is an effective access method for very large text databases whereas the two level scheme performs better for smaller ones. Slow interactive insertion remains to be a problem for both schemes but can be remedied to a certain extent using batch insertion algorithms [Kent et al. 1990; Sacks-Davis and Ramamohanarao 1987].

IV.3. Multikey Access Methods as Alternatives to Two Level Scheme

In order to improve the performance of the two level scheme proposed in [Sacks-Davis and Ramamohanarao 1987], three multikey access methods which combine the inverted indexes and the signature files and are based on term discrimination and signature clustering have been proposed [Chang, J. W. et al. 1989]. In all three methods, there exists a separate block descriptor for the terms with high discriminatory power (primary terms) and the ones with low discriminatory power (secondary terms). Each method uses a bit-slice representation for the secondary block descriptor file. Similar record signatures, instead of similar records, are clustered and this clustering is based on the similarity between the primary terms. The analysis is based on single term queries only.

The first method, Primary-signature-based-Two-level-Signature-file Method, PTSM, has the primary block descriptor file represented as a BS. Since the primary and the secondary terms have their own block descriptor (signature) files, no false drop occurs when primary and secondary terms are combined to form a block-signature. Smaller values for the false drop can be achieved by increasing the size of the primary block descriptors . The second method, Inversion-based-Two-level-Signature-file Method, ITSM, replaces the primary block descriptor by an index file. False drops are eliminated by storing the actual primary terms in the index area. However, insertion is slow and the

space overhead is high. The last method, Hash-table-based-Two-level-Signature-file Method, HTSM, is somewhere between the two extreme structures discussed above. It uses a hash table to decrease the false drops. Besides, since pointers (and not the terms themselves) are stored only, storage requirements are lower. Information needed for clustering is stored in the postings file.

The results of the study [Chang, J. W. et al. 1989] indicate that PTSM requires the least storage overhead since the structure is purely based on signatures and ITSM is the fastest. HTSM yields good performance in both retrieval speed and storage overhead. The proposed methods are also evaluated to be promising to provide additional gains in the retrieval efficiency compared to the two level scheme proposed in [Sacks-Davis and Ramamohanarao 1987].

IV.4. Problems with Multilevel Organizations

Two major problems of the multilevel schemes are addressed in [Chang W. W. and Scheck 1989]. The first one pertains to the convergence of the higher level signatures into all 1s bit vectors where all bit positions are occupied by 1s. This situation impairs the selectivity of the higher level signatures and degrades the retrieval efficiency. The analysis shows that even for optimal object signatures where half of the bits are set to 1 (see Section II.4.1), therefore, higher level signatures tend to get cluttered very quickly [Chang W. W. and Scheck 1989]. Using clustering techniques for lower level signatures and using the block descriptors can bring solutions to this signature saturation problem [Deppish 1986; Sacks-Davis 1985; Sacks-Davis and Ramamohanarao 1987].

The second problem which has not seen much treatment is related to the combinatorial error. Assuming that two records each with two fields reside on one leaf node where the contents of the records are represented as R_1 and R_2 such that

$$R_1 = (V_{11}, V_{21}) \text{ and } R_2 = (V_{12}, V_{22})$$

where V_{ji} denotes the ith value for the jth field, the parent signature that is created by superimposing these two record signatures will represent the combination of not only R1 and R2 but also that of

$$R_3 = (V_{11}, V_{22}) \text{ and } R_4 = (V_{12}, V_{21}).$$

In general, the parent signature represents not only the N records in the corresponding leaf node but also all records which can be obtained by any Cartesian Product of the field value combinations of the fields of these N records. If we let M_j denote the number of distinct V_{ji} values and f stand for the number of fields for which predicates are specified in a query, the probability of finding a matching record is given by P_{Match} such that

$$P_{Match} = \frac{N_{1...f}}{\prod_{j=1}^{f} M_j} \qquad P_{Cerror} = 1 - P_{Match}$$

and $N_{1...f}$ indicates the number of distinct records when only f fields are considered. Consequently, P_{Cerror} denotes the probability of a combinatorial error [Chang W. W. and Scheck 1989].

The combinatorial error problem pertains to text data as well since the text signatures are generated in such a way that they not only represent the original text phrase but also any phrase which can be generated by any combination of the words in the original phrase. In this case, the probability of occurrence of a combinatorial error increases as more word signatures are combined to form the signature of the text phrase and as more words are specified in the queries.

A solution to the combinatorial error problem is proposed in [Chang W. W. and Scheck 1989] where in addition to the conventional leaf signatures, called S1, larger combinatorial signatures, CS1, are also generated to reduce the probability of false matches. CS1 for a leaf record is formed by setting one bit for each pair of bit positions in S1 that are both set to 1. All CS1's for a leaf node are superimposed to form the higher level combinatorial signature, called CS2. An example showing how the proposed method works is presented in Figure 10. CS2 does not confirm the combinatorial signature of the query and hence no retrieval takes place. Note that a false match would have resulted, had the original signatures (S1 and Query S1) been used for query processing. The issues of selecting an appropriate density for S1 and the algorithm to generate CS1 are further discussed in [Chang W. W. and Scheck 1989].

Record	**S1**	**CS1**
$R_1 = (V_{11}, V_{21})$	1 0 1 0	0 1 0 0 0 0
$R_2 = (V_{12}, V_{22})$	0 1 0 1	0 0 0 0 1 0
	1 1 1 1	0 1 0 0 1 0
	$\Downarrow$	$\Downarrow$
	S2	CS2
Query	**Query S1**	**Query CS1**
$Q = (V_{11}, V_{22})$	1 0 0 1	0 0 1 0 0 0

Figure 10. Query processing using combinatorial signatures.

V. HORIZONTALLY PARTITIONED SIGNATURE FILE ORGANIZATION METHODS

The basic motivation behind horizontal partitioning is to achieve better search time. Below we provide an overview and discussion of various horizontal partitioning schemes proposed up to date together with an application of the major superimposed signature methods to one such organization in a single and multiterm environment.

V.1. Gustafson's Method

In an environment where N documents each having k key words (or records having k attributes) exist, a hashing function is used to map a key word to an integer number i in the range {0, m-1}, where m is the signature size. The signature of a key word is created by setting the i^{th} bit position to 1. Word signatures are then superimposed to form the document signature. If the number of 1s in the resulting document signature, n, is less than k, (k-n) bits are set randomly. Then there are comb(m, k) = C possible document signatures, where comb(m, k) denotes combinations of m choose k items. To each such signature N/C documents will be matched. C document lists are kept and the following function is used to convert each possible bit string corresponding to a document into a number between 0 and C:

$$\text{comb}(p_1, 1) + \text{comb}(p_2, 2) + \ldots + \text{comb}(p_k, k)$$

where $p_1 < p_2 < \ldots < p_k$ and p_i's correspond to the positions of the 1s in the document signature [Gustafson 1971; Knuth 1975] and by definition, comb(0, t) = 0 for any integer t > 0.

When a query consisting of s key words is submitted, each of these key words are fed into the hashing function. If all s key words are distinct, only those documents stored in the comb(m-s, k-s) lists whose signatures contain 1s in the positions specified by the query key words have to be checked. Therefore, only ((comb(m-s, k-s)/C))*100 % of the documents have to be accessed for this query.

Using the method, extent of search decreases with the number of terms in a conjunctive query, i.e., where terms are combined using AND. However, the performance drops with increasing file size. Also queries other than the conjunctive ones are handled with difficulty [Faloutsos 1985].

V.2. Partitioning to Implement Ranking

The study of Croft and Savino on ranking using signatures [Croft and Savino 1988] has later been criticized, since the algorithm it suggests to compute the approximate term frequencies is useful for long documents only. Besides, because a separate term signature has to be created for each term in a partial match query, the response time becomes slower as the query weight increases [Wong and Lee, D. L. 1990].

Two partitioning schemes are proposed to encompass an exact representation of the term frequencies in signature files and to reduce the I/O time [Wong and Lee, D. L. 1990; Wong 1991]. The first scheme decomposes the document matrix D (in which every row represents a document, every column corresponds to a term and the number of occurrences of term j in document i is given in (D[i, j])) into a set of matrices called the tf groups. Each tf group corresponds to one value of the term weight and $tf_k[i, j] = 1$ if and only if $D[i, j] = k$ and $tf_k[i, j] = 0$ otherwise.

An entry dictionary where an entry consists of the term itself, its document frequency (df) and the term ordinal number is also kept. By convention, when the first term is inserted to the dictionary, it is given an ordinal number of 0, the second is assigned to 1 and so on. These ordinal numbers determine the position of the bits to be set in the term signatures. Such an organization for the entry dictionary does not add much to the cost of a system implementing the tf*idf ranking strategy [Can and Ozkarahan 1990; Salton and Buckley 1988] since the df values have to be updated upon insertion and retrieved during a search.

Documents are assigned to blocks based on their ordinal number and the corresponding signatures are created where the j^{th} term of a block sets the j^{th} bit in its signature. The signatures of the terms in the same tf group are superimposed only if they belong to the same block. The query signatures are generated in the same manner and the inverse document frequency (idf) values are computed for each query term based on the document frequencies and the size of the collection. Then the query terms are grouped using the rounded idf values and the signatures of the terms in a group are superimposed. These signatures are then compared to the document signatures to find the number of matching terms and to compute the document weights which in turn determine the documents to be retrieved [Wong and Lee, D. L. 1990; Wong 1991].

The second method aims to further reduce the search space by avoiding to access those signature pages which can not contribute to the weights of the documents in the ranking process. Hence by further splitting the terms into range groups based on their term frequencies and ordinal numbers, the so-called tr method adds a coarse indexing to the existing structure of the tf method. This time, only the range groups containing the query terms are accessed in contrast to the tf method which requires the scanning of all tf groups to answer a query.

The partial file scanning provided by tr reduces the I/O activity. The storage requirements of both methods are almost the same and less demanding than that of inverted indexes. Yet when viewed as an inversion method, tr is still less efficient than inverted

files. Further improvement of the response time is possible with the parallel implementation [Wong and Lee, D. L. 1990; Wong 1991].

V.3. Key-based Partitioning

When no suitable partitioning scheme is used to assign the signatures to partitions, which are disjoint sets of signatures, all partitions still have to be accessed for each query. In a parallel environment, even this can improve the speed since a processor can be assigned to each partition and all partitions can be scanned simultaneously. This is called intra query parallelism. Note that with this method, only one query can be handled at a time [Lee, D. L. 1989].

A better way to use horizontal partitioning is to assign the signatures to partitions in such a way that the signatures in one partition share the common "key." When a query is submitted to the system, only those partitions whose keys seem to qualify the query need to be accessed. Hence the search space is reduced and the retrieval speed is improved. Besides, both inter and intra query parallelism can be achieved this time since the inactive processors which are assigned to those partitions that do not have to be accessed for the query being processed can be used to service other queries.

In addition to its advantages in parallel environments, such partitioning can also bring savings in a sequential single processor environment by reducing the search space. It also requires less processing time compared to the multilevel structure. Besides, since all signatures in a partition have the same key, only the nonkey portions need to be stored. Hence non random partitioning demands less storage overhead compared to single and multilevel organizations [Lee, D. L. and Leng 1989].

A good deal of research has been devoted to find the mapping scheme using which the signatures will be assigned to the partitions. Below we will provide an overview of two such schemes one consisting of three methods which are based on the same idea but differ in the way the keys are extracted, and the other based on the principles of Linear Hashing.

V.3.1 Fixed vs. Variable Length Key Partitioning

Lee and Leng has suggested and evaluated three ways for signature mapping [Lee, D. L. and Leng 1989]. All three methods use SC technique to generate the signatures and also assume that all signatures consist of a key portion as well as a nonkey part. It is this key part that determines which partition the signature will be stored in. Key portions of all signatures in a partition are the same and constitute the partition key. Similarly, the query signatures have these two parts. The key of a query signature is extracted in the same way as the keys for the partitions and only those partitions whose keys qualify include the query key are accessed. Hence if the key portions of the query signature and the i^{th} partition, P_i, are shown as K_Q and K_{Pi}, respectively, then the partition P_i is accessed only if $(K_Q \cap K_{Pi}) = K_Q$.

The three methods provide different ways to extract keys from the signatures. Their performance is compared based on the resulting reduction in the search space and the uniformity of the workload of the processors, assuming a parallel architecture. Signature reduction ratio, which is the ratio of the number of signatures searched to the total number of signatures and the partition reduction ratio which is the ratio of the number of partitions searched to the total number of partitions, are the two measures of the first criterion. Partition activation probability, P_a, is defined as the probability that a partition will be searched for a query and the equality of the activation probabilities is accepted as an indicator of the uniformity of the workload, when a processor is assigned to a partition and the partitions have the same size.

The first method, Fixed Prefix Partitioning (FPP), takes the first k bits of the signature as the key. This method, being the simplest of all three, uses a simple key extraction algorithm and hence can be used for sequential systems. For parallel applications, however, it is not appropriate since the distribution of the workload is far from being uniform. For the second method, Extended Prefix Partitioning (EPP), the key is chosen to be the shortest prefix which contains a predefined number of zeros indicated by z, hence is of variable length. This forces each key to contain a predefined number of zeros so that no partition will be activated for all queries and a uniformity in the partition activation probabilities (P_a's) will be achieved. This is because when the key of a partition consists of all 1s (which can be the case for FPP), this partition qualifies to any query and hence is accessed at all times. However, this method creates highly non uniform partition sizes, therefore although the P_a values are equal the workload is not uniform. Floating Key Partitioning method (FKP) examines each of the consecutive nonoverlapping k-substrings of a signature and selects the leftmost substring that has the least amount of 1s. This is to avoid the non uniformity in the partition sizes seen in EPP due to the possibility of having very long key lengths belonging to partitions with very few signatures which happens when signatures which have too many ones followed by zeros are used.

Figure 11 shows how the same sequence of signatures are partitioned using the three schemes discussed above and indicates the percentage of partitions and signatures accessed for a particular query (101 000) for each organization. Signature size is taken as 6, the values for k and z are assumed to be 2. The partition keys are shown by bold digits and symbol P_i is used to refer to the ith partition. Experimental results show that the FKP method is the most attractive one for both sequential and parallel environments. It outperforms the first two methods when the signature and partition reduction ratios are

compared. The P_a values are still not equal but a more uniform workload is achieved compared to FPP. The only drawback is the relative complexity of the algorithm used to obtain the keys [Lee, D. L. and Leng 1989].

	FPP		EPP		FKP
P_1	**00** 0111	P_1	**00** 0111	P_1	**00** 0111
	00 1011		**00** 1011		**00** 1011
	00 1101		**00** 1101		**00** 1101
	00 1110		**00** 1110		**00** 1110
P_2	**01** 0011	P_2	**010** 011	P_2	01 **00** 11
	01 0101		**010** 101		10 **00** 11
	01 0110		**010** 110		11 **00** 01
	01 1001	P_3	**0110** 01		11 **00** 10
	01 1010		**0110** 10	P_3	0111 **00**
	01 1100	P_4	**01110** 0		1011 **00**
P_3	**10** 0011	P_5	**100** 011		1101 **00**
	10 0101		**100** 101		1110 **00**
	10 0110		**100** 110	P_4	**01** 0101
	10 1001	P_6	**1010** 01		**01** 0110
	10 1010		**1010** 10		**01** 1001
	10 1100	P_7	**10110** 0		**01** 1010
P_4	**11** 0001	P_8	**1100** 01	P_5	**10** 0101
	11 0010		**1100** 10		**10** 0110
	11 0101	P_9	**11010** 1		**10** 1001
	11 1000	P_{10}	**11100** 0		**10** 1010

Query Signature: 101 000

	FPP	EPP	FKP
partitions to be accessed	P3, P4	P6, P7, P10	P3, P5
no. of partitions accessed	2	3	2
no. of signatures accessed	10	4	8
partition activation ratio (percentage of partitions accessed)	50%	30%	40%
signature activation ratio (percentage of signatures accessed)	50%	20%	40%

Figure 11. FPP, EPP, FKP based signature file organizations (adopted from [Lee, D. L. and Leng 1989]).

When these methods are adopted in parallel environments, the query signature is sent to all search processes which extract the key portion of the query in the same way as the corresponding partition key and compare both keys to see if the partition should be accessed. Partitions with non qualifying keys terminate the search and become ready for the next query. Since the assignment of signatures to partitions is done in parallel and distributed among search processes, no special data structure is required.

In sequential environments, the need for a data structure arises where the partition keys are to be kept and compared against the key of the query signature sequentially. One such data structure which is adaptable to the growth of the signature file and applicable to all three schemes is proposed in [Lee, D. L. and Leng 1990] together with the algorithms for insertion, deletion and retrieval. The proposed scheme provides dynamic storage allocation by using a way similar to dynamic hashing [Larson 1978; Tharp 1988] and hence eliminates the problems originating from the growing number of partitions and the non uniformity of the partition sizes.

V.3.2 Linear Hashing with Superimposed Signatures: LHSS

V.3.2.1 The Method

In all of the above schemes the procedure to determine the key is somewhat static. This limits the dynamic nature of the organization [Grandi et al. 1992]. A dynamic partitioning scheme has recently been suggested where the authors have been inspired from the extensive research in dynamic storage structures for formatted data designed for primary-key-exact-match queries [Zezula et al. 1991]. However, they note that in contrast to these structures, in signature file related approaches conjunctive partial-match queries are of concern. They call the new approach Linear Hashing with Superimposed Signatures (LHSS) or Quick Filter.

The primary component of LHSS is a split function which converts the key of each signature into an integer in the address space {0, 1, . . ., n-1} where $2^{h-1} < n \leq 2^h$ is satisfied for some integer $h > 0$. The hashing function is defined as follows [Zezula 1988, Zezula et al. 1991].

$$g(s_i, h, n) = \begin{cases} \sum_{r=0}^{h-1} b_{F-r} 2^r & \text{if } \sum_{r=o}^{h-1} b_{F-r} 2^r < n \\ \sum_{r=0}^{h-2} b_{F-r} 2^r & \text{otherwise} \end{cases}$$

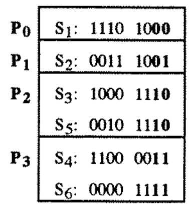
where b_i is the value of the ith binary digit of the object signature, F is the signature size, h is the hashing level, n is the number of addressable (primary) pages and s_i is the object signature i.

For the initial condition $h = 0$, $n = 1$, $g(s_i, 0, 1)$ is defined as 0. In simple terms, the hashing function, g, uses the last h or (h-1) bits of a signature to determine the number of the page where signature s_i is to be stored. If the storage limit of a primary page is exceeded, an overflow page is created, linked to the primary page and the last signature that has caused the overflow is placed in the overflow page and, a "split" is initiated, i.e., a new primary page is created. A split pointer, SP (with an initial value of 0), keeps track of the

next primary page to be split. Whenever a split takes place, all signatures on the page pointed to by SP, together with those in the associated overflow page(s) are rehashed. The nature of the hashing function guarantees that the rehashed signatures either remain in the same page or are transferred to the page that has just been created. The hashing level is increased by one just before page zero is split, and following each split process the new value of SP is computed as SP = ((SP + 1) mod 2^{h-1}). Note that at a given time in the signature file it is possible to have pages which are hashed at levels h and (h-1): The pages starting with the one pointed by SP up to the page with index 2^{h-1} (exclusive) are hashed at level (h-1). Note also that linear hashing is space efficient and does not lead to many overflows [Litwin 1980].

S_1:	1110 1000
S_2:	0011 1001
S_3:	1000 1110
S_4:	0110 0011
S_5:	0010 1110
S_6:	0000 1111

Step	Description	Split	Signature mapping (after split, if any)	SP	h	n
0	Initialization	-	**P_0**: EMPTY	0	0	1
1	Insert S_1	-	**P_0**: S_1	0	0	1
2	Insert S_2	-	**P_0**: S_1 S_2	0	0	1
3	Insert S_3	P_0	**P_0**: S_1 S_3 **P_1**: S_2	0	1	2
4	Insert S_4	-	**P_0**: S_1 S_3 **P_1**: S_2 S_4	0	1	2
5	Insert S_5	P_0	**P_0**: S_1 P_1: S_2 S_4 **P_2**: S_3 S_5	1	2	3
6	Insert S_6	P_1	**P_0**: S_1 **P_1**: S_2 **P_2**: S_3 S_5 **P_3**: S_4 S_6	0	2	4



Q_1: 1100 0111	Q_2: 1111 0000	Q_3: 1100 0110
$g(Q_1, 2, 4) = 3$	$g(Q_2, 2, 4) = 0$	$g(Q_3, 2, 4) = 2$
==> access P_3 only	==> access all pages	==> access P_2 and P_3

Figure 12. Working mechanism of LHSS.

During query processing a page qualifies if all bit positions that are set in the query signature are also set in the page signature. For simplicity, if we assume that $n = 2^h$ and if there is a query signature with k 1s in its h-bit suffix, then it is necessary to access 2^{h-k} primary pages (and the associated overflow pages). More number of 1s in the last h-bit suffix of a query makes the query processing faster. Note that even if a signature in the selected page seems to qualify the query the associated data object might not contain all query terms. Hence a false drop resolution is required using the original query before the qualifying objects are returned to the user.

Figure 12 demonstrates the working mechanism of LHSS as 6 signatures are inserted in a file where each page can hold a maximum of two signatures. The symbol S_i stands for the ith object signature, Q_j indicates the jth query signature and P_k represents the kth page. The organization of the file during each step of the insertion is shown together with a final representation of the structure. The page numbers that are highlighted in bold correspond to those pages that are hashed at level h while the page numbers in plain text are the ones that are hashed at level (h-1). Processing of three different query signatures is also explained.

V.3.2.2 Proposed Improvements

LHSS fulfills many requirements of today's applications. Its retrieval efficiency improves with the query weight and the size of the database which makes it a perfect choice for fast search of very large databases. It can also be used as an integrated access method to retrieve text, voice and image in multimedia applications where high query weight is typical. Besides, the dynamic nature of LHSS promotes easy insertion and deletions which is a major pitfall of many signature file organizations. Even exhaustive search is not very expensive since the expected overflow is low [Zezula et al. 1991].

As for the future improvements, the authors attract particular attention to the use of Extendible Hashing for the implementation of LHSS [Fagin et al. 1979; Tharp 1988]. The number of bits considered for hashing, h, grows faster in Extendible Hashing. Since the retrieval efficiency of LHSS improves with h, extra savings are projected in the earlier stages of the signature insertion.

Another opportunity for possible improvement lies in the modification of the hashing function. Signatures in one partition share the same key but the current hashing function can not prevent neighboring pages from having considerably different suffixes. As a result, qualifying pages might be apart from each other causing many random disk accesses.

Considering the signature suffixes as Gray Codes, which are proposed by Faloutsos as an alternative to multiattribute hashing, can alleviate the problem. The idea is to make successive codewords of the buckets (or partitions) differ in one bit position only so that successive buckets (partitions) hold similar record signatures. Improved clustering of records is achieved which replaces a portion of the random disk accesses by the sequential ones [Faloutsos 1986; Faloutsos 1988b].

The 4-bit binary reflected Gray Code representation is given in Figure 13 where each code represents the characteristics of a page. The pages that need to be accessed to process a query with signature 0001 have been marked for both binary and Gray Code representations. Qualifying pages are scattered when the binary code is used, whereas the clustering of the signatures reduce the number of random disk accesses when the Gray Codes are used. It has been proved that the Gray Codes never perform worse than the binary method and they are shown to provide 0-50% savings for any partial match query. The only overhead of the method resulting from conversion of code is outweighed by the savings in I/O time [Faloutsos 1986; Faloutsos 1988b].

Decimal	Binary	Gray
0	0000	0000
1	**0001**	**0001**
2	0010	**0011**
3	**0011**	0010
4	0100	0110
5	**0101**	**0111**
6	0110	**0101**
7	**0111**	0100
8	1000	1100
9	**1001**	**1101**
10	1010	**1111**
11	**1011**	1110
12	1100	1010
13	**1101**	**1011**
14	1110	**1001**
15	**1111**	1000

Query Signature : 0001

Figure 13. Query processing using binary vs. gray codes.

In a more recent work, the partition activation ratio (PAR) is defined as the ratio of the partitions activated by a query to the total number of partitions [Ciaccia and Zezula 1993] (Note the parallelism between PAR and the partition reduction ratio discussed in [Lee, D. L. 1986]) The study reported in [Ciaccia and Zezula 1993] provides an approximate and easy-to-compute formula for PAR, which is shown to be applicable for both FPP and LHSS. This approximation is useful not only because it has a very small margin of error (at most 1.2%), but also because it provides an attractive alternative over the complicated performance evaluation formulas of both methods which give exact results. Nevertheless, the applicability of the approximate formula to both methods should not be interpreted as identical behavior since these two partitioning schemes differ in their key and partition generation strategies.

V.3.3 Application of SM, MMS and MMM Schemes to LHSS

Aktug and Can have analyzed the effects of relaxing the unrealistic uniform frequency assumption and applying different treatments to terms based on their occurrence and query frequencies in LHSS environment [Aktug and Can 1993a]. They have used the SM and MMS approaches (see Section II.1) to create the signatures and then comparatively evaluated their performance. In contrast to the traditional SM method where each term sets the same number of bits, in the MMS approach, terms with high discriminatory power, which are typically characterized by low occurrence frequency coupled with high query frequency are allowed to set more bits in signatures. This in turn increases the query weight and results in an improvement in retrieval efficiency. The terms with low discriminatory power, on the other hand, set fewer bits and hence produce low weight queries for which the amount of page savings is also low. However, because queries are usually composed of terms with high discriminatory power, the gains in the former case more than offset the decrease in savings in the latter case.

The authors also show that using MMS instead of SM accomplishes a balance between relevancy and retrieval efficiency. More specifically, when SM is used, the number of bits set by each term is identical. Hence when a single term query is specified in a query, the query weight is constant and equals m. So the expected number of bits in the last h-bit suffix of the query signature is the same regardless of the term discriminatory power values. This, in turn, means that the number of page accesses is the same for all terms. When a term with a low discriminatory power is specified in a query, a long list of documents will be returned. (Notice that terms with low discriminatory power are the ones that appear in many documents.) Yet a large portion of the returned documents will not be of interest to the user. Hence the resulting relevancy will be very low. In contrast, when a term with high discriminatory power is used in the query, only a few documents, most of which will be relevant, are returned to the user, and the relevancy level will be significantly high.

The above situation which is typical in the SM case indicates an obvious imbalance between efficiency and relevancy. For the same number of page accesses (i.e. for the same level of efficiency), it is possible to end up with low or high values of relevancy depending

on the frequency characteristics of the query term. The more significant the difference between the discriminatory power of the terms, the more severe is the imbalance described above.

When MMS is used, the terms with high discriminatory power set more bits than those with low discriminatory power. Hence, the number of page accesses required for these two cases will differ in the first place. Consequently, the terms with high discriminatory power provide relatively more page savings which will be consistent with the high level of the resulting relevancy. On the other hand, terms with low discriminatory power will somehow be penalized because now they will be setting fewer bits. The resulting page savings will be low together with the undesirably low relevancy level. The way to achieve high efficiency coupled with high relevancy is to increase the query weight. This can be accomplished by using terms with high discriminatory power in the queries or by constructing term phrases from non-discriminatory terms. In an IRS, the former can be supported by an on-line thesaurus providing group of related specific terms under more general, higher level class indicators; the latter can be implemented by automatic phrase construction [Salton 1975; Salton 1989].

The experimental analysis presented in the study explore the amount of page savings with different occurrence and query combinations at different hashing levels. The results show that the performance of LHSS improves with the hashing level and the choice of the signature size depends on the compromise between the required percent savings and the tolerable false drop rate. The results also indicate that the higher is the difference among the discriminatory power values of the terms, the higher is the extra savings provided by MMS [Aktug and Can 1993a].

A recent study by the same authors compares the performance of SM, MMS and MMM schemes (see Section II.1 for the definitions) in LHSS environment when both single and multiterm queries are considered. The main contribution of the study is to relax the single term query assumption and examine the query characteristics of the system when both single and multiterm queries can be submitted. The analysis is more complex since the query weight which is the major input of the performance evaluation formulations is no longer a constant but a random variable whose distribution is expressed using the last equation in Section II.5.

The terms in the database are assumed to be grouped into two sets, S_1 and S_2, where S_1 contains the ones with high discriminatory power. The terms from S_i set M_i $(1 \leq i \leq 2)$ number of bits and therefore m_j of the last equation in Section II.5 equals to M_1 or M_2. Let t be the maximum number of terms that can be used in a query and let P_j indicate the occurrence probability of a query with j terms where $(P_1+P_2+...+P_t = 1)$ is satisfied. The tree diagram in Figure 12 is used to depict the query structure where at each query combination, which is represented as a final outcome, the answers to these three questions are known:

1. How many terms are there in the query?
2. How many terms are from S_1 and how many are from S_2, i.e., how many of the query terms set M_1 bits and how many of them set M_2 bits?
3. Which set does the first query term belong to, i.e., how many bits does the first term set?

This information enables us to compute the value of $P(W(Q) = s+m_1 \mid m_1)$ for each query outcome for those values of s that are realizable. If the tree is traced from left to right, starting from the leftmost node, numbered as 0, we encounter t possibilities, each corresponding to a query with 'nqt' terms, where nqt stands for the number of query terms and ranges from 1 to t. Each of the t branches symbolize one of these t events (i.e., specification of a query with nqt terms) and the probability associated with each event is indicated on its corresponding branch. Note that the sum of the probabilities associated with the branches emanating from a particular node adds up to 1.

The submission of a single term query takes us to node 1 at which we have two possibilities: The term is either from S_1 or S_2.
Let b_i : number of terms from S_i $(1 \leq i \leq 2)$ in a query , then

$$\sum_{i=1}^{2} b_i = nqt$$

should be satisfied. Therefore, it is sufficient to use just b_1 (or b_2) to specify a query combination, once nqt is known. For a single term query, the possible values for b_i are 0 and 1 where

$$P\{b_1 = 1 \mid nqt = 1\} = q_1 \text{ and } P\{b_1 = 0 \mid nqt = 1\} = q_2$$

and q_i is the probability that a query term is from set i (as in Section II.1).

These two conditions take us to two final outcomes which can not be split up any further. From any node n $(2 \leq n \leq t)$, where n = nqt, (nqt+1) branches emanate, each corresponding to one possible value for b_1 in the range 0 to nqt.

$$P\{b_1 = v \mid nqt = V\} = \binom{V}{v} q_1^v q_2^{(V-v)}$$

where $0 \leq v \leq V$ and $2 \leq V \leq t$.

Therefore, starting from node n, we can end up in any one of the (n+1) outcomes. However, some of them can further be split up so that we will have the information to answer the three questions that are listed at the beginning of this section.

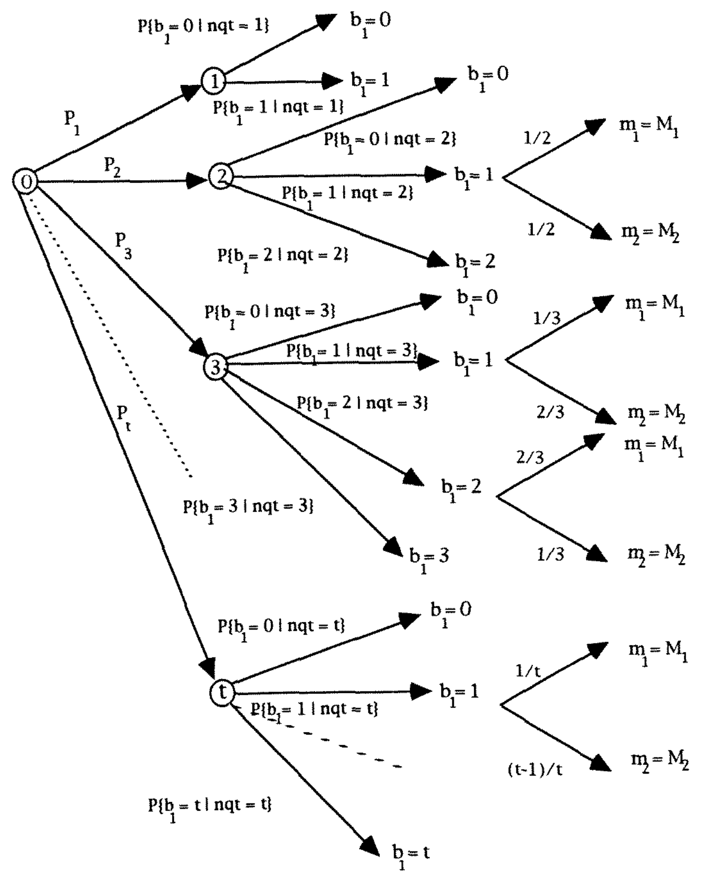


Figure 14. The tree diagram for the query outcomes.

At each of these (n+1) outcomes, the number of query terms and the number of 1s set by each term are known. For the first and (n+1)st outcomes, the number of bits set by the

first term is also known since these outcomes correspond to the cases where all query terms come from a single set. For the remaining (n-1) outcomes, we need further simplification depending on whether the first query term is from S_1 or from S_2. For each such case, let
$P\{FT \in S_i\}$: probability that first term is from S_i $(1 \leq i \leq 2)$
then

$$P\{FT \in S_1\} = \frac{b_1}{nqt} \text{ and } P\{FT \in S_2\} = \frac{b_2}{nqt}$$

and hence

$$P\{FT \in S_1\} + P\{FT \in S_2\} = 1.$$

In general, for t term queries, there are (t+1) branches and hence (t+1) outcomes. Two of these are final, the remaining (t-1) split into two. Hence we have (2 + 2(t-1)) , i.e., 2t final outcomes. For each of the final outcomes, the value for the expected number of bits in the last h suffix of the query has been computed which is converted to an overall expected value for the system using the probabilities of the branches of the tree. This value is then used to compute the percentage of pages that do not have to be accessed which is the indicator for the amount of savings obtained.

These savings for each method are computed at various experimental settings and the extra savings provided by MMM over the other methods are computed. The maximum number of terms in a query is assumed to be 10 and three specific query cases are created, LW for the situation where low weight queries are common, HW for the situation where the high weight queries are most frequent and UD for the case where all P_i are equal. The results shown that both MMS and MMM are clearly superior to SM in all cases. The extra savings provided by MMM over MMS increase as the gap among the discriminatory power values of the terms gets larger and the probability distribution of the number of terms in the query depicts a non uniform pattern. This is because MMM considers the nature of the probability distribution of the number of query terms in determining the optimal assignment strategy and emphasizes the terms with high discriminatory power in particular [Aktug and Can 1993b].

VI. PARALLEL PROCESSING OF SIGNATURE FILES

VI.1. Signature Processors

Many search strategies (full text scanning, inverted files, clustered files, signatures, PAT trees, etc.) have been proposed in the literature for text retrieval. The purpose is to find efficient ways to cope with the complexity of the operations and the increased processing time of the large databases [Faloutsos 1985; Gonnet et al. 1992; Hollaar 1992; Ozkarahan and Can 1984; Ozkarahan 1986; Salton and Buckley 1988]. As mentioned before, signatures provide simple file structure, ease of maintenance, low storage requirement and congruity with parallel processing techniques. However, even the virtues of the signature approach can be limited to a certain extent since due to the inadequacy of the von Neumann architecture, the software techniques fail to maintain the system performance as the database size grows and the access frequency increases [Lee, D. L. 1986].

Signature processors aim to bring in hardware related solutions that will improve the retrieval speed and handle complex queries [Lee, D. L. 1986; Lee, D. L. and Lochovsky 1990]. Unlike inverted file processors, which face the problem of growing index size and unstable hardware costs, and full text scanning processors which poorly utilize the disk bandwidth, signature processors are simple, regular in structure and do not place substantial hardware requirements [Lee, D. L. and Lochovsky 1990]. Two signature processor architectures known as Word-Serial, Bit-Parallel (WSBP) and Word-Parallel, Bit-Serial (WPBS) are discussed below. WSBP method stores the signatures in faster memory (CCD, magnetic bubble or RAM modules) based on the assumption that the size of the signature file is small compared to that of the database. These memories have the capability to access a large bit vector at one shot as opposed to the bit/byte serial access nature of the disks. For this feature and due to their high memory access rate, these memories have very high bandwidth.

A hardware solution based on WSBP architecture suggests storing signatures in high density semiconductor RAMs and retrieving them sequentially for comparison against the query signature [Ahuja and Roberts 1980]. The actual text file is stored on the disk and signatures are generated and stored in the associative memories. Since the size of a typical signature file is 10-20% of the original database, the total amount of the associative memory used is insignificant. This organization becomes more cost efficient with the decreasing cost of the semiconductor memories.

The system defined by Ahuja and Roberts is made up of two subsystems: the front-end and the back-end processors. The front-end processor handles user communication, includes a superimposed signature generator to create signatures for updates and queries, facilitates access to actual records and resolves the false matches. The back-end processor functions are confined to the signature file only. It searches the signature file to retrieve the qualifying signatures for a query and handles special query specifications (e.g., searching for N matches).

The resulting search time which provides considerable speedup is robust to the changes in the database size in contrast to the software implementations where the response time gets drastically slow as the database size increases. The results demonstrate the attractiveness of the superimposed signatures for partial-match retrieval when used with special hardware. However, the proposed method is not optimal since the whole signature file, rather than the minimum amount of bits required to process the query is read [Lee, D. L. 1986; Lee, D. L. and Lochovsky 1990]. More specifically, for a signature file that contains n signatures of size F bits each, all $(n*F)$ bits rather than $(n*W(Q))$ bits are read, where W(Q) is the query weight. Hence the method results in inefficient use of the I/O bandwidth and the associated hardware since only W(Q)/F of the bandwidth is utilized. Since W(Q) is much less than F in most occasions, the performance of the method is severely impaired.

Analogous to the reduction in the amount of data read when a bit-slice representation is used instead of a sequential one, a transposed organization can make use of the full bandwidth [Lee, D. L. and Lochovsky 1990]. The signature processor proposed in [Lee, D. L. 1986] is based on the WPBS approach where the signatures are stored in signature blocks of capacity n_b each. Signatures in a block are searched in parallel whereas the signature blocks are processed sequentially. Similar to the bit-slice approach, only W(Q) bits from every signature have to be accessed. Hence the search time of a signature block consists of W(Q) stages regardless of the value of n_b. The search time, S, will be $(W(Q)*\lceil n/n_b \rceil$ when $n > n_b$, where $\lceil x \rceil$ is the ceiling of x). Besides, a multiple-response resolver (MRR) is also a part of the architecture to facilitate the data transfer between the processors and controllers over a single bus. Hence the total time to search n signatures can be expressed as (S+MRR time) as compared to the WSBP architecture where it equals n. Both processors require the same amount of storage, but in general WPBS is approximately $n_b/W(Q)$ times faster than WSBP [Lee, D. L. 1986]. WPBS approach can also accommodate signatures of variable length and is more efficient.

The design and implementation aspects of an hybrid text retrieval machine called HYTREM, which is based on the WSBP approach, are discussed in [Lee, D. L. and Lochovsky 1990]. The structure is referred to as hybrid in the sense that it uses both text and signature processors as access methods for large text databases. The use of a signature processor provides two important advantages. First, the relatively slow access time of the conventional moving head disks that are chosen as the secondary storage medium because of their cost effectiveness is compensated for. This is because the signatures provide a filter which reduces the amount of data that needs to be accessed from the secondary storage. Secondly, the same filtering mechanism enables the encouraging performance results provided by the system which can not be accomplished using neither the inverted indexes (because of the high storage and processing requirements) nor full text scanning (simply because of the lack of any kind of filtering or indexing).

VI.2. Parallel Processing Applications

Since the time to search a signature file increases with the database size, if the conventional von Neuman architecture is kept in use, undesirably long response times are inevitable for large databases. Parallel machines are very attractive in this perspective [Stanfill 1992].

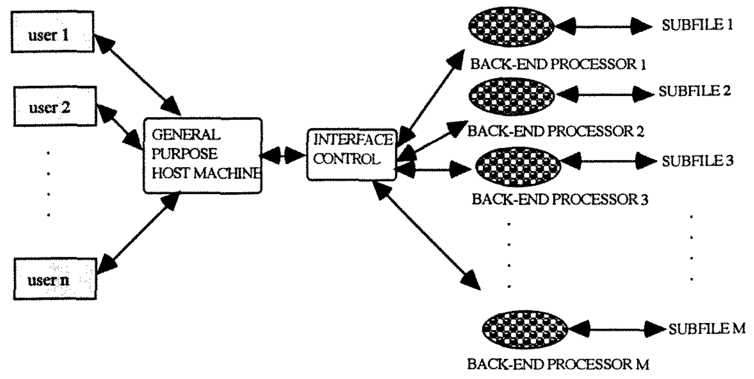


Figure 15. A typical multiprocessor arrangement using back-end search processors.

Parallel signature processors have been used in text retrieval to rank the documents once the document scores have been computed with respect to a particular query [Stanfill 1992]. A typical multiprocessor arrangement which can be used to speed up text retrieval is shown in Figure 15. This configuration enables simplicity of search applications since it provides fast response without the need for auxiliary file clustering or index maintenance operations. However, coordination of the processors must be maintained. It is also possible to extend the back-end processor philosophy and increase the number of back-end machines and let each processor control the operations on the data that is assigned to it. Conceptually, it is possible to have as many processors as the number of the documents and search the whole document collection in one extensive search operation carried on by numerous parallel machines. This organization simplifies the control operations and enables simpler individual processor design [Salton and Buckley 1988a].

One implementation of the back-end search machine is the Connection Machine (CM), which is a massively parallel computer which has up to 64k processing elements [Tannenbaum 1990] enables very fast free-text search [Hillis 1985]. Stanfil and Kahle discuss an application where the underlying data structure is called surrogate coding which is the signature approach itself [Stanfill and Kahle 1986]. They also provide the performance results of a prototype system where the use of relevance feedback is realized by the capabilities of CM. Relevance feedback is superior to both Boolean and simple (natural language) queries because it includes the construction of queries from the text of the documents that have already been marked relevant by the user. Without CM or any similar machine facilitating parallelism, the processing of feedback queries becomes costly since this technique requires handling of feedback queries with hundreds of terms. The prototype system provides fast response time, significantly better recall and precision (compared to Boolean or simple query type search strategies).

Array processors which contain hundreds or thousands of processors can lead to significant performance improvement over the sequential machines with the use of the proper algorithms. One example of the use of such parallel structures with signature files can be found in [Pogue and Willet 1987] where a general purpose array processor, called Distributed Array Processor (DAP), is used to implement an experimental document retrieval system where documents and queries are characterized by text signatures.

Another study reported in [Carrol et al. 1988] also uses DAP to implement the pattern matching part of a bibliographic retrieval system based on text signatures. Although only document titles and abstracts are included in the experimental framework, the results indicate that DAP coupled with an efficient pattern matching algorithm provides increased search efficiency compared to a conventional system.

VI.3. Frame-sliced Partitioning

In an effort to create a signature file organization that will give acceptable performance in most (possibly all) applications, Frame-sliced Partitioned Parallel Structure has been proposed [Grandi et al. 1992]. The underlying idea is based on the observation that most file structures depend on the nature of the query and the system characteristics. The aim is to generate a file structure with more stable performance. After considering the pros and cons of the existing file structures, the method attempts to combine the advantages of partitioned and bit sliced organizations. Since the performance of partitioned signature files improve with the increase in the query weight whereas that of the sliced organizations degrade, a combinatory method is hypothesized to lead to a better and stable performance [Lee, D. L. and Leng 1989; Zezula et al. 1991]. More specifically, a fragmentation scheme that combines Frame-sliced Organization (see Section III.2.2) and LHSS (see Section V.3.2) is proposed [Lin and Faloutsos 1992].

Bit-slice signature file organizations have been used as a fragmentation scheme for a parallel hardware implementation where a bus structure is used to connect the modules. In this structure, modules are activated in parallel in search of specific bit-slices upon query submission [Roberts 1979]. Using the Vertical Parallelism (VP) approach, the above idea

can be used with frame-slices where time to search the signature file is almost equal to the time to search one slice if all slices are of the same size. With Horizontal Parallelism (HP), on the other hand, a horizontal fragment of object signatures is assigned to each processor [Stanfill and Kahle 1986; Pogue and Willet 1987]. HP provides only intra query parallelism since it checks all signatures assigned to a processor. Similarly, LHSS can also be implemented in a parallel environment by assigning each partition to a processor. This time, however, inter query parallelism can be accomplished as well as the intra query parallelism since only a portion of the partitions are accessed for each query. This approach is called Partitioned Parallelism (PP).

The new scheme suggested in [Grandi et al. 1992], Frame-sliced Partitioning (FSP), is a mixed fragmentation scheme that consists of double horizontal fragmentation of vertical fragments. When used with the Shared Nothing architecture of the multiprocessor database computers, the performance of the proposed method is worse than that of VP, HP and PP for very low weight queries, especially when the frame width is large. For medium to high weight queries, however, the performance of FSP improves substantially.

The advantages of the FSP method is its flexibility in the amount of data, the number of frames and the degree of parallelism within a frame which come with a superseding performance compared to the other parallel partitioning structures, when the same level of parallelism is used. The performance of the method is not stable in all conditions as expected, but it gives improved performance compared to bit-sliced and partitioned methods. The complimentary use of the parallel architecture enables the accomplishment of adequately high performance for both large and on-line applications. As for the reliability issue, which is a major concern of the parallel architecture, the FSF method is advantageous since the failure of a processing unit does not lead to the failure of the whole signature file since the faulty unit may not even be used by a given query or even if it is used, the outcome will be nothing but an increase in false drops. In the worst case, the performance will be degraded but the system still continues to handle the queries properly [Grandi et al. 1992].

VII. SIGNATURE FILE APPLICATIONS FOR MULTIMEDIA DATABASES

In parallel with the proliferation of the multimedia databases, multimedia information systems which provide functions for the creation, extraction, correlation and distribution of information have evolved. Multimedia databases require more sophisticated access strategies compared to those that can be applicable for text and formatted data. Signature files have been successfully implemented in multimedia office file systems [Zezula et al. 1991].

The outline of an Office Filing System which consists of multimedia messages that are made up of any combination of text, voice and image is presented in [Tsichritzis et al. 1983]. The system provides filtering and browsing capabilities where filtering enables the user to define the properties of the messages he/she wants to obtain and browsing allows the user to sort out the relevant messages from the set returned as a result of the filtering operation. Browsing is emphasized as an integral part of the system since the user filters are vague most of the time. Also the user is given the capability to modify the filter while browsing a set of messages to form the link between the two functions which are mistakenly taken as independent and consecutive in many previous applications.

The system uses superimposed signatures as the access method for text and attribute values, where a fixed length signature is created for the attributes and a separate signature pertains to each block of the body of text. In addition to the signatures, miniatures; which are realistic visual abstractions of the messages displayed on the screen during browsing, image descriptions; which show the image types and positions in a message and fasttalk; which is a voice excerpt that provides brief information as to what the message is about, constitute other information abstracted from the messages.

MINOS is a multimedia information system based on an object-oriented model which creates and manages documents containing attributes, text, images and voice. MINOS enables active presentation and browsing capabilities, formats multimedia documents and provides efficient integrated tools for searching a specific information within a repository of multimedia documents and for extracting information from selected documents. Information can be shared, selected, transformed and merged or new information can be generated [Christodoulakis et al. 1986; Christodoulakis and Faloutsos 1986]. The role of signature files as a text access method is an important part of the system since most of the time users of a multimedia database are expected to submit queries whose content is described by text.

MULTOS, on the other hand, is a multimedia office server that facilitates filing operations and supports query processing on multimedia documents. Document sharing is allowed and the client/server architecture is used. The system's query processor uses different access structures for different document components: Separate B-tree indexes are used for formatted data, images and sequential and bit-slice signature techniques are used for text. However, query optimization becomes a problem with such a structure since a lack of integration is present among the different access methods [Thanos 1990; Zezula et al. 1991].

Zezula proposes an integrated signature generation scheme based on SC to solve this problem. An F bit signature which represents the text and image content is allocated to each document The signature of the textual part of a document is generated in the usual manner where each word sets a specified number of bits. As for the images, the underlying objects are identified and their code words are superimposed to form the image signature. The mapping of the object signatures to the code words are provided in a lookup table. If an image I consists of the objects O_1, O_2, O_3, where O_2 and O_3 are complex objects, the image contents can be represented as

$$I = O_1, O_2 (O_4, O_5 (O_7, O_8)), O_3 (O_9, O_{11})$$

Notice that O_2 is a complex object that consists of O_4 and O_5, where O_5 itself is also complex and contains two other objects (O_7, O_8). The third object in the image, O_3, is made up of two simple objects (O_9 and O_{11}).

A simple object signature is created by setting n of the F bit positions using the specifications given in the lookup table. A complex object signature, on the other hand, is generated by setting the n bits corresponding to the object itself, as well as the bits set by the simple objects that it contains. In our example, the signature of O_2 is created by superimposing the signatures of O_4 and O_5 where the signature of O_5 is generated by superimposing the signatures of O_7 and O_8. Text and image signatures are then superimposed to form the document signature. All such document signatures are kept in a single signature file which enables easy query processing. Although the object signatures are uniquely defined, due to the superimposition process, false drops can still occur. Hence a false drop resolution on the images must be provided [Zezula et al. 1991].

The study reported in [Rabbiti and Savino 1991] uses the above idea to generate image signatures for a database that consists of image data only. It presents a general purpose image query language and then describes the use of signature files to provide fast access to the images within the framework of this language. Prior to image analysis, all classes of the images specific to the system of concern (the application domain) should be determined and described. On such an application domain, an image analysis process attempts to recognize the objects present in the images together with various interpretations, associated positions in the image space and the degree of recognition. This process tries to describe the (complex) objects in terms of the simple (basic) ones that they contain. Image signatures are generated in the same way where complex objects set their own n bits in addition to the bits set by the simple objects that they contain.

A four level signature scheme which consists of image, image-interpretation, context and context-interpretation level signatures is proposed [Rabbiti and Savino 1991]. The performance of this scheme is tested using a prototype system with four types of queries (having different complexity level) on three image databases: IDB-S (containing simple images), IDB-M (containing images of medium complexity) and IDB-C (containing complex images). The results indicate that the false drop probability is highly dependent on the type of the image database and is less responsive to the changes in the signature size. False drop probabilities are computed for each of the four levels of signatures. The results show that the resulting false drop which corresponds to that of the forth level signature is very small and even zero in some cases.

These results are encouraging but somewhat limited since the authors admit that more experiments must be conducted before these findings can be extended to include other image databases. The performance of the signatures with image databases is difficult to evaluate since most of the assumptions that belong to the text file applications of signatures do not hold: Text databases are (realistically) assumed to contain large number of distinct words whereas the image databases usually contain a few dozens of distinct images. The experience with the image databases strongly suggests that the assumption concerning word frequencies will also not work. The probabilistic independence of word occurrency assumption, on the other hand, does not even make sense for the image databases because semantic rules which describe the construction of complex objects from single ones exist. Since the findings of the research on signature applications on text files can not be directly adopted to image databases, much still needs to be done. Nevertheless, the use of signature files with image databases is encouraging since the preliminary results are highly positive and promising [Rabbiti and Savino 1991].

VIII. SUMMARY AND REEVALUATION

Table III. outlines the strengths and weaknesses of the major signature generation methods. Table IV. provides an overview of the major signature file organization schemes and describes the type of environments in which they will perform best.

A recent study on signature-based multikey access methods classifies the major signature file organizations into four groups depending on whether they use multilevel indexes and whether they apply special treatment to terms with high discriminatory power [Chang, J. W. et al. 1992]. One-Path Single Level (OPSL) organization includes the BS representation whereas Two-Path Single Level (TPSL) method refers to those single level organizations where either a separate access path is defined for the terms with high discriminatory power or such terms are emphasized by letting them set more number of bits. One-Path Two-Level signature files include the two level file representation proposed in [Sacks-Davis and Ramamohanarao 1987] (discussed in Section IV.2) and the Two-Path Two-Level (TPTL) organization includes the PTSM, ITSM, and HTSM methods [Chang, J. W. et al. 1989] (discussed in Section IV.3.)

Table III. Overview of the Strengths and Weaknesses of the Signature Generation Schemes

Method	Strengths	Weaknesses
Superimposed Coding (SC)	• handles queries on parts of words and numeric fields • automatically eliminates duplicate words • enables implementation of ranking strategies • accounts for term occurrence and query frequencies • can be used for formatted and unformatted databases • can be used in the implementation of various signature file organization schemes	• can not preserve information sequence
Word Signatures (WS)	• preserves information sequence • gives high performance when used for objects defined by a variable number of descriptors • improves the efficiency of partial-match retrieval by • solving large key space problem • assigning different priority to the fields based on their relative importance	• yields relatively high false drop • gives low performance when comparing signatures for qualification • can be used with sequential organizations only
Run-length Encoding (RL)	• accomplishes the least false drop • achieves excellent compression	• requires long search time • can be used with sequential organizations only
Bit-block Compression (BC)	• yields good performance in comparing signatures for qualification	• can not handle queries about sequencing of words
Variable Bit-block Compression (VBC)	• demands the least CPU time • performs best for very long documents queries with many terms objects of variable length	• can be used with sequential organizations only • can not handle queries about sequencing of words

The comparison of the performance of these four groups of organizations show that TPSL achieves 40-80% gains on retrieval but requires 9% more storage compared to OPSL. TPTL, on the other hand, provides 20-55% gains in retrieval and requires 4% more storage overhead compared to OPTL. When the number of matching records for a query is small, TPTL supersedes TPSL, whereas TPSL outperforms in the reverse case [Chang, J. W. et al. 1992].

The performance of the indexed descriptor files [Pfaltz et al. 1980] (see Section IV.1.1), two level signature file organizations [Sacks-Davis and Ramamohanarao 1987] (see Section IV.2) and partitioned schemes [Lee, D. L. and Leng 1989] (see Section V.3.1) are compared in [Lee, D. L. et al. 1992]. All methods are assumed to use superimposed coding and a hashing function that provides a uniform distribution of the 1s in the signatures. The storage overhead and the signature reduction ratio (ratio of the number of signatures accessed to the total number of signatures) are selected as the measures of performance. The results indicate that the two level scheme provides the highest signature reduction ratio. The two level and indexed descriptor files have similar organizations but the two level scheme provides superior performance since its filtering mechanism is more effective. This is because the cluttering effect that degrades filtering is more significant for the indexed descriptor files where the size of the signatures at all levels are the same and higher level signatures are created by superimposing a group of lower level signatures.

Table IV. Overview of the Strengths and Weaknesses of the Major Signature File Organizations

Category	Scheme	Strengths	Weaknesses	Best for
Single Level	Sequential (SS)	• simple to implement • provides easy update and append operations	• response speed drops severely with increasing database size	• small database sizes
	Bit-sliced (BS)	• improves retrieval efficiency by accessing only those bit positions specified in the query	• results in costly updates • retrieval performance relies on the query weight	• queries with low weight • static environments like archival applications
	Frame-sliced (FS)	• provides faster insertion capability compared to the bit-sliced method without too much space overhead • the generalized version includes BS and SS as its specific cases	• not efficient with shared disks	• any one of the CD-ROM, WORM and magnetic disk environments
Multi level	S-tree	• decreases the search space by grouping similar signatures in the same node	• requires high space overhead	• high weight queries
	Two Level	• accomplishes improved performance over the bit-sliced organization when a small number of records qualify the query specifications	• gives rise to unnecessary block matches • retrieval performance relies on the query weight • insertion is slow	• cases where insertions can be batched • cases where typically a small number of records confirm to a query
	Multi organization	• improves performance by diminishing the number of unnecessary block matches	• insertion is slow	• cases where insertions can be batched • very large text databases

Table IV. Overview of the Strengths and Weaknesses of the Major Signature File Organizations (cont.)

Category	Scheme	Strengths	Weaknesses	Best for
Horizontal Partitioning	Fixed Prefix Partitioning (FPP)	• uses a simple algorithm to extract keys form signatures • incurs little space overhead	• results in non uniform distribution of workload among the processors in a parallel environment	• sequential systems
	Extended Prefix Partitioning (EPP)	• results in uniform partition activation probabilities by requiring every key to have a predetermined number of zeros	• gives rise to partitions with high variance of size and non uniform distribution of workload	• parallel environments with some loss of efficiency • sequential systems with data structure that enables dynamic storage allocation
	Floating Key Partitioning (FKP)	• achieves the highest reduction in search space compared to FPP and EPP	• uses a relatively complex algorithm to extract keys from the signatures	• sequential and parallel environments
	Linear Hashing with Superimposed Signatures (LHSS) alias Quick Filter	• expands and shrinks in accordance with the requirements of the dynamic environment • achieves higher performance as database size or query weight (or both) increases • enables significant reduction in the search space • facilitates easy insertion and deletion	• causes many random disk accesses when signatures in the neighboring pages have different suffixes	• dynamic environments • high weight queries • multimedia applications • sequential and parallel environments

In contrast, in the two level scheme, the size of the signatures increase at every level and higher level signatures are formed independent of the lower level signatures (no superimposition). The selected partitioned organizations (FPP and FKP) also have a limited filtering capability when the key length is kept small in order not to have too many partitions which will result in poor space utilization. This time, however, especially for low weight queries, the probability that "the bits that are set in the query signature will correspond (at least partially) to those in the key" is small. Hence, the number of eliminated partitions is not significant enough to accomplish a high signature reduction ratio.

As for the storage overhead, for unsuccessful searches, the two level scheme is shown to require the same amount of space as the single level organization to accomplish the same amount of false drop. The results might sound surprising since higher level signatures apparently create extra storage overhead. However, since upper level signatures are created independent of the lower level ones, a false drop introduced in a lower level can be discarded by the next higher level and so on. Besides, the signatures at all levels of a two level structure are shorter that those in the single level structure. On the other hand, for successful searches, the two level scheme requires more storage overhead than the single level one, as expected.

Based on the above findings, D. L. Lee, Kim and Patel propose a generalized method based on the idea of the two level scheme which consists of multilevels of signatures. This multilevel organization is then shown to supersede all other methods (two level, indexed descriptor and partitioned) while requiring the same amount of storage overhead. It is also indicated that the optimal number of levels can be computed using the formula $\log_2 n$ where n is the number of object signatures [Lee, D. L. et al. 1992].

IX. CONCLUSION AND FUTURE RESEARCH POINTERS

Signature files have been successfully used as an information storage and retrieval technique for both formatted and unformatted databases. Signature generation techniques have evolved to take the discriminatory power values of the terms into account so that the terms with low database occurrence frequency and high query frequency are emphasized to decrease the false drop probability and hence improve the retrieval efficiency. The use of such signature generation schemes also helps improve the efficiency relevancy balance. Signature file organizations, on the other hand, have enhanced from single level schemes to multilevel and partitioned organizations.

Future research on signature generation can deal with the application of the FWB approach (see Section II.1) to multiterm query cases as well as to various signature file organization schemes. Encouraging results are likely because FWB is expected to work well with multilevel and partitioned signature file structures where the performance improves with the query weight. This is because the FWB weight assignment approach gives high weight to the frequent query terms [Leng and Lee, D. L. 1992]. Future research dealing with such applications will most probably yield enhanced search performance together with a decrease in false drop probability. Additionally, a coordination between the concepts of signature generation and signature file organization can be maintained as suggested in [Aktug and Can 1993b].

As for signature file organization, the principles of Gray Codes and extendible hashing can be incorporated to improve the retrieval efficiency in dynamic schemes like LHSS [Zezula et al. 1991]. Combinatorial organizations, similar to frame-sliced partitioning, should be explored: Signature partitions can be organized in a multilevel structure and the performance of this organization can be analyzed to observe whether the low storage overhead and direct access capability of the partitions can be enhanced with the efficiency

of the multilevel structure [Lee, D. L. et al. 1992]. Signature applications to parallel environments should also be emphasized since the presence of multiprocessors is an effective solution to the problem of handling very large databases without creating unacceptably long response times.

Future effort should also be devoted to the performance evaluation of the proposed signature file organizations in multiterm query environments as opposed to adoption of the unrealistic single term query assumption. The results pertaining to the derivation of the probability distribution of the query weight provided in [Murphree and Aktug 1992] can serve as guidelines for such research. Simpler formulations for the solution to the same problem can also be generated to enable less cumbersome performance evaluation analysis. The results provided in [Ciaccia and Zezula 1993] can be used as inspiration for this type of research.

Signature applications on image and multimedia databases should also be pursued to explore the extent of the encouraging results that have been obtained so far. However, a new set of principles need to be established for image analysis and representation, since the assumptions that are used for text signatures are shown to be inapplicable in this case. In addition to text, multimedia and relational database applications, the signature [illegible] can also be used in the indexing of object-oriented databases [Lee, W-C and [illegible]].

The latest developments in signature file organizations have made s[illegible] even more effective and efficient for dynamic environments and for very large databases. Recent signature file organizations have especially been proved to give outstanding performance in parallel environments and the signature technique has been pinpointed as a promising integrated access method for the manipulation of the multimedia databases. All this positive evidence suggests that the field of information storage and retrieval has a lot more to gain from future research and development in signature files.

REFERENCES

AHUJA, S. R., AND ROBERTS, C. S. 1980. An associative/parallel processor for partial match retrieval using superimposed codes. In *Proceedings of the 7th Annual Symposium on Computer Architecture.* (France, May), pp. 218-227.

AKTUG, D., AND CAN, F. 1993a. Signature file hashing using term occurrence and query frequencies. In *Proceedings of the 12th Annual IEEE International Phoenix Conference on Computers and Communications.* (Phoenix, Arizona, March), pp. 148-153.

AKTUG, D., AND CAN, F. 1993b. Analysis of multiterm queries in a dynamic signature file organization. To appear in *16th Annual International ACM-SIGIR Conference* (Pittsburgh, Penn., June).

BLAIR, D. C. 1990. *Language and Representation in Information Retrieval.* Elsevier, Netherlands.

BORDOGNA, G., CARRARA, P., GAGLIARDI, I., MERELLI, D., MUSSIO, P., NALDI, F. AND PADULDA, M. 1990. Pictorial indexing for an integrated pictorial and textual IR environment. *Journal of Information Science*. 16, 165-173.

BURKOWSKI, F. J. 1990. Surrogate subsets: A Free space management strategy for the index of a text retrieval system. In *Proceedings of the 13th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. (Brussels, Belgium, September), ACM, New York, pp. 211-226.

CAN, F. 1993a. On the efficiency of the best-match cluster searches. To appear in *Information Processing and Management*.

CAN, F. 1993b. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems*. 11, 2 (April), 143-164.

CAN, F., AND OZKARAHAN, E. A. 1987. Computation of term/document discrimination values by use of the cover coefficient concept. *Journal of the American Society for Information Science*. 38, 3 (May), 171-183.

CAN, F., AND OZKARAHAN, E. A. 1990. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems*. 15, 4 (December), 483-517.

CARROLL, D. M., POGUE, C. A., AND WILLET, P. 1988. Bibliographic pattern matching using the ICL Distributed Array Proccessor. *Journal of the American Society for Information Science*. 40, 11 (November), 390-399.

CHANG, J. W., LEE, J. H. , AND LEE, Y. J. 1989. Multikey access methods based on term discrimination and signature clustering. In *Proceedings of the 12th Annual International ACM-SIGIR Conference* (Boston, Mass., September), ACM, New York, pp. 176-185.

CHANG, J. W., YOO, J. S., AND LEE, Y. J. 1992. Performance comparison of signature-based multikey access methods. *Microprocessing and Microprogramming*. 35, 1, 5, 345-352.

CHANG, W. W., AND SCHECK, H. J. 1989. A signature access method for the starburst database system. In *Proceedings of 15th International Conference on VLDB* (Amsterdam, Netherlands, August), pp. 145-153.

CHRISTODOULAKIS, S., AND FALOUTSOS, C. 1984. Design considerations for a message file server. *IEEE Transactions on Software Engineering*. 10, 2 (March), 201-210.

CHRISTODOULAKIS, S., AND FALOUTSOS, C. 1986. Design and performance considerations for an optical disk-based, multimedia object server. *Computer* 12, (December), pp. 45-56.

CHRISTODOULAKIS, S., THEODORIDOU, M., HO, F., PAPA, M. AND PATHRIA, A. 1986. Multimedia document presentation, information extraction, and document formation in MINOS: A model and a system. *ACM Transactions on Office Information Systems*. 4, 4 (October), 345-383.

CIACCIA, P., ZEZULA, P. 1993. Estimating accesses in partitioned signature file organizations. *ACM Transactions on Information Systems*. 11, 2 (April), 133-142.

COLOMB, R. M., AND JAYASOORIAH. 1985. A clause indexing system for PROLOG based on superimposed coding. *The Australian Computer Journal*. 18, 1 (August), 18-25.

COOPER, L., K., D., AND THARP A. 1988. Inverted Signature trees and text searching on CD-ROMs. *Information Processing & Management*.. 25, 2, (March), 161-169.

CROFT, W., B., AND SAVINO, P. 1988. Implementing ranking strategies using text signatures. *ACM Transactions on Office Information Systems*. 6, 1 (January), 42-62.

DATE, C. J. 1990. *An Introduction to Database Systems, 5th edition*. Addison Wesley, Reading, MA.

DEPPISH, U. 1986. S-tree: A Dynamic balanced signature index for office retrieval. In *Proceedings of the 9th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (Pisa, September), ACM, N.Y., pp. 77-87.

FAGIN, R., NIEVERGELT, N., PIPENGER, N., AND STRONG, H. R. 1979. Extendible hashing: A fast access method for dynamic files. *ACM Transactions on Database Systems*. 4, 3 (September), 315-344.

FALOUTSOS, C. 1983. Access methods for text. *ACM Computing Surveys*. 17, 1 (March), 49-74.

FALOUTSOS, C. 1986. Multiattribute hashing using gray codes. In *Proceedings of ACM SIGMOD '86 Conference* (Washington, D.C., May), pp.227-238.

FALOUTSOS, C. 1987. Signature files: An integrated access method for text and attributes, suitable for optical disk storage. In *University of Maryland Computer Science Technical Report Series*, June.

FALOUTSOS, C. 1988a. Signature Files: An integrated access method for text and attributes, suitable for optical disk storage. *BIT*. 28, 4, 736-754.

FALOUTSOS, C. 1988b. Gray codes for partial match and range queries. *IEEE Transactions on Software Engineering*. 14, 10 (October), 1381-1393.

FALOUTSOS, C. 1992. Signature files. In *Information Retrieval Data Structures and Algorithms*, edited by W. B. Frakes, R. Baeza-Yates, Prentice Hall, Englewood Cliffs, N.J., pp 44-65.

FALOUTSOS, C, AND CHRISTODOULAKIS, S. 1984. Signature files: An access method for documents and its analytical performance evaluation. *ACM Transactions on Office Information Systems*. 2, 4 (October), 267-288.

FALOUTSOS, C., AND CHRISTODOULAKIS, S. 1985. Design of a signature file method that accounts for non uniform occurrence and query frequencies. In *Proceedings of the 11th International Conference on VLDB* (Stockholm, Sweden, August). VLDB Endowment, pp. 165-170.

FALOUTSOS, C., AND CHRISTODOULAKIS, S. 1987a. Optimal signature extraction and information loss. *ACM Transactions on Database Systems*. 12, 3 (September), 395-428.

FALOUTSOS, C., AND CHRISTODOULAKIS, S. 1987b. Description and performance analysis of signature file methods for office filing. *ACM Transactions on Office Information Systems*. 5, 3 (July), 237-257.

FALOUTSOS, C, AND CHAN, R. 1988. Fast text access methods for optical and large magnetic disks: designs and performance comparison. In *Proceedings of the 14th VLDB Conference*. (Los Angeles, California), pp. 280-293.

FALOUTSOS, C., AND JAGADISH, H. V. 1992. Hybrid index organizations for text databases. *EDBT '92* (Vienna, Austria, March 23-27), pp. 310-327.

FELLER, W. 1968. *An Introduction to Probability Theory and its Applications, 3rd ed.* John Wiley & Sons, New York.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractability*. W.H. Freeman and Co., San Fransisco, California.

GONNET, G. H., BAEZA-YATES, R. A. , AND SNIDER, T. 1992. New indices for text: PAT trees and PAT arrays *Information Retrieval Data Structures and Algorithms*, edited by W. B. Frakes, R. Baeza-Yates, Prentice Hall, Englewood Cliffs, N.J., pp 66-82.

GRANDI, F., TIBERIO, P., AND ZEZULA, P. 1992. Frame-sliced partitioned parallel signature files. In the Proceedings of *15th Annual International ACM-SIGIR Conference* (Kopenhagen, Denmark, June), ACM, New York, pp. 286-297.

GUSTAFSON, R. A. 1971. Elements of the randomized combinatorial file structures. *In the Proceedings of the ACM SIGIR Symposium on Information Storage and Retrieval*, ACM, New-York, pp. 163-174.

HILLIS, D. 1985. *The Connection Machine*. Cambridge, Mass. MIT Press.

HOLLAAR, L. A. 1992. Special purpose hardware for information retrieval *Information Retrieval Data Structures and Algorithms*, edited by W. B. Frakes, R. Baeza-Yates, Prentice Hall, Englewood Cliffs, N.J., pp. 443-458.

KENT, A., SACKS-DAVIS R., AND RAMAMOHANARAO, K. 1990. A Signature file scheme based on multiple organizations for indexing very large text databases. *Journal of the American Society for Information Science*. 41, 7 (October), 508-534.

KNUTH, D., E. 1975. *The Art of Computer Programming*. vol. 3: *Sorting and Searching*. Addison-Wesley, Reading, Mass.

KOTAMARI, U., AND THARP, A., L. 1990. Accelerating text search through signature trees. *Journal of the American Society for Information Science*. 41, 2 (March), 79-86.

LARSON, P. 1978. Dynamic hashing. BIT, 18, 2, 184-201.

LEE, D. L. 1986. A word-parallel, bit-serial processor for superimposed coding. In *Proceedings of the 2nd International Conference on Data Engineering* (Los Angeles, California, February), IEEE, New York, pp. 352-359.

LEE, D. L., AND LENG, C. W. 1989. Partitioned signature files: Design issues and performance evaluation. *ACM Transactions on Information Systems*. 7, 2 (April), 158-180.

LEE, D. L., AND LENG, C. W. 1990. A partitioned signature file structure for multiattribute and text retrieval. In *Proceedings of the 6th International Conference on Data Engineering* (Los Angeles, California), pp. 389-397.

LEE, D. L., AND LOCHOVSKY F. H. 1990. HYTREM- A hybrid text-retrieval machine for large databases. *IEEE Transactions on Computers*. 39,1 (January), 111-123.

LEE, D. L., KIM, Y. M., AND PATEL, G. 1992. Efficient signature file methods for text retrieval. (Preprint. 1st author's address: Department of Computer and Information Science, Ohio State University, Columbus, OH, 43210-1277, USA)

LEE, W-C., AND LEE, D. L. 1992. Signature file methods for indexing object-oriented databases. In *Proceedings of International Computer Science Conference* (Hong Kong, December), IEEE, pp. 616-622.

LENG, C.-W R., LEE, D. L. 1992. Optimal weight assignment for signature generation. *ACM Transactions on Database Systems*. 17, 2 (June), 346-373.

LIN, Z., AND FALOUTSOS, C. 1992. Frame-sliced signature files. *IEEE Transactions on Knowledge and Data Engineering*. 4, 3 (June), 281-289.

LITWIN, W. 1980. Linear hashing: A new tool for files and tables addressing. In *Proceedings of the 6th International Conference on VLDB* (Montreal, October), pp. 212-223.

MORAN, S. 1983. On the complexity of designing optimal partial-match retrieval systems. *ACM Transactions on Database Systems*. 8, 4 (December), 543-551.

MURPHREE, E., AND AKTUG, D. 1992. Derivation of probability distribution of the weight of the query signature. (Preprint. 1st author's address: Department of Mathematics and Statistics, Miami University, Oxford, OH 45056, USA.)

OROSZ, G., AND TAKACZ, L. 1956. Some probability problems concerning the marking of codes into the superimposition field. *Journal of Documentation*. 12, 4 (December), 231-234.

OZKARAHAN, E. A. 1986. *Database Machines and Database Management*. Prentice-Hall, Englewood Cliffs, N. J.

OZKARAHAN, E. A., AND CAN, F. 1984. An integrated fact/document information system for office automation. *Information Technology Research and Development*. 3, 3, 142-156.

OZKARAHAN, E. A., AND CAN, F. 1986. An automatic and tunable document indexing system. In *Proceedings of the 9th Annual International ACM-SIGIR Conference* (Montreal, Canada, September), ACM, New York, pp. 234-243.

OZKARAHAN, E. A., AND CAN, F. 1991. Multimedia document representation and retrieval. In *Proceedings of the ACM 1991 Computer Scienece Conference* (San Antonio, Texas, March), ACM, New York, pp.420-429.

PFALTZ, J. L., BERMAN, W. J. , AND CAGLEY, E. M. 1980. Partial-match retrieval using indexed descriptor files. *Communications of the ACM*. 23, 9 (September), 522-528.

POGUE, C., WILLET, P. 1987. Use of text signature for document retrieval in a high parallel environment. *Parallel Computing*. 4, 259-268.

RABBITI, F., AND SAVINO, P. 1991. Image query processing on multilevel signatures. In *Proceedings of the 14th Annual International ACM-SIGIR Conference* (Chicago, Illinois, September), ACM, New York, pp. 305-314.

RAMAMOHANARAO, K., AND LLOYD, J. W. 1983. Partial-match retrieval using hashing and descriptors. *ACM Transactions on Database Systems*. 8, 4 (December), 552-576.

RAMAMOHANARAO, K., AND SHEPHERD, J. 1986. A superimposed codeword indexing scheme for very large prolog databases. In *Proceedings of the International Conference on Logic Programming*, pp. 569-576.

ROBERTS, C.S. 1979. Partial match retrieval via the method of superimposed codes. In *Proceedings of the IEEE*. 67, 12 (December), pp. 1624-1642.

SACKS-DAVIS, R. 1985. Performance of a multikey access method based on descriptors and superimposed coding techniques. *Information Systems*. 10, 4, 391-403.

SACKS-DAVIS, R., AND, RAMAMOHANARAO, K. 1987. Multikey access methods based on superimposed coding techniques. *ACM Transactions on Database Systems*. 12, 4 (December), 655-696.

SALTON, G. 1975. A theory of indexing. *Regional Conference Series in Applied Mathematics*, 18. Society for Industrial and Applied Mathematics, Philadelphia, PA.

SALTON, G. 1989. *Automatic Text Processing: The Transformation Analysis, and Retrieval of Information by Computer.* Addison Wesley, Reading, Mass.

SALTON, G., AND BUCKLEY, C. 1988a. Parallel text search methods. *Communications of the ACM.* 31, 2 (February), 202-215.

SALTON, G., AND BUCKLEY, C. 1988b. Term-weight approaches in automatic text retrieval. *Information Processing and Management.* 24, 5 (May), 513-523.

SALTON, G., AND MCGILL, M.J. 1983. *Introduction to Modern Information Retrieval.* McGraw-Hill Computer Series.

STANFILL C. 1992. Parallel information retrieval algorithms. In *Information Retrieval Data Structures and Algorithms,* edited by W. B. Frakes, R. Baeza-Yates, Prentice Hall, Englewood Cliffs, N.J., pp. 459-497.

STANFILL C., AND KAHLE B. 1986. Parallel free-text search on the connection machine system. *Communications of the ACM.* 29, 12 (December), 1229-1239.

STIASSNY, S. 1960. Mathematical analysis of various superimposed coding methods. *American Documentation.* 11, 2 (February), 155-169.

TANNENBAUM, A., S. 1990. *Structured Computer Organization.* Prentice Hall, Englewood Cliffs, N.J.

THANOS, C., ED. 1990. Multimedia office filing and retrieval: The MULTOS approach. In *North Holland Series in Human Factors in Information Technology*, North-Holland, Amsterdam.

THARP, A. L. 1988. *File Organization and Processing.* John Wiley and Sons, New York, N.Y.

TIBERIO, P., AND ZEZULA, P. 1991. Selecting signature files for specific applications. In *Proceedings of Advanced Computer Technology, Reliable Systems and Applications, 5th Annual European Conference.* (Bologna, Italy, May) IEEE, 1991, pp. 718-725.

TSICHRITZIS, D., AND CHRISTODOULAKIS, S. 1983. Message files. *ACM Transactions on Office Information Systems.* 1, 1 (January), 88-98.

TSICHRITZIS, D., CHRISTODOULAKIS, S., ECONOMOPOULOS, P., FALOUTSOS C., LEE, A., LEE, D., VANDENBROEK, J. and WOO, C. 1983. A multimedia office filing system. In *Proceedings of 9th International Conference on VLDB* (Florence, Italy, October), VLDB Endowment, pp. 2-7.

VAN RIJSBERGEN, C. J. 1979. *Information Retrieval, 2nd edition.* Butterworths, London.

ULLMAN, J. D. 1988. *Principles of Database and Knowledge-Base Systems.* Computer Scienece Press, Rockville, Maryland.

WONG W. Y. P. 1991. Design and performance evaluation of access methods and heuristic techniques for implementing document ranking strategies. Ph. D. Dissertation, Dept. of Computer and Information Sciences, The Ohio State University, Columbus, OH.

WONG W. Y. P., AND LEE D. L. 1990. Signature file methods for implementing a ranking strategy. *Information Processing and Management.* 26, 5, pp. 641-653.

ZEZULA, P. Linear hashing for signature files. 1988. In *Proceedings of the IFIP TC6 and TC8 Open Symposium on Network Information Processing Systems.* (Sofia, Bulgaria, May), pp. 243-250.

ZEZULA, P., RABITTI, AND F., TIBERIO, P. 1991. Dynamic partitioning of signature files. *ACM Transactions on Information Systems.* 9, 4 (October), 336-367.

ZIPF, G.K. 1949. *Human Behavior and Principle of Least Effort: An Introduction to Human Ecology.* Addison Wesley, Mass.

ZOBEL, J., THORN, J. A. AND SACKS-DAVIS, R. 1991. Efficiency of nested relational document database systems. In *Proceedings of the 17th International Conference on Very Large Databases* (Barcelona, Catalonia, Spain, September), Morgan Kaufman, San Mateo, California, pp. 91-102.