Signature File Hashing Using Term Occurrence and Query Frequencies

Deniz Aktug[*] Fazli Can[†]

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

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

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE
& SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-008

Signature File Hashing Using Term Occurrence And Query Frequencies

Deniz Aktug and Fazli Can

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Signature File Hashing Using
Term Occurrence and Query Frequencies

by

Deniz Aktug Fazli Can
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-008 August, 1992

SIGNATURE FILE HASHING USING TERM OCCURRENCE AND QUERY FREQUENCIES

Deniz AKTUG **Fazli CAN**[*]

Department of Systems Analysis
Miami University
Oxford, OH 45056

July 6, 1992

Abstract

Signature files act as a filter on retrieval to discard a large number of non-qualifying data items. Linear hashing with superimposed signatures (LHSS) provides an effective retrieval filter to process queries in dynamic databases. This study is an analysis of the effects of reflecting the term occurrence and query frequencies to signatures in LHSS. This approach relaxes the unrealistic uniform frequency assumption and lets the terms with high discriminatory power set more bits in signatures. The simulation experiments based on the derived formulas explore the amount of page savings with different occurrence and query frequency combinations at different hashing levels. The results show that the performance of LHSS improves with the hashing level and the larger is the difference between the term discriminatory power values of the terms, the higher is the retrieval efficiency. The paper also discusses the benefits of this approach to alleviate the imbalance between the levels of efficiency and relevancy in unrealistic uniform frequency assumption case.

[*] (513) 529-5950, fc74sanf@miamiu.bitnet

1. INTRODUCTION

The purpose of an information retrieval system (IRS) is to locate data items which are relevant to the user queries. In a multimedia IRS the stored objects can be formatted or unformatted (text, voice, image, etc.). A non-exhaustive list of IR techniques include full text/object scanning, inverted indexes, clustering, and signature files [1, 4, 7, 15, 16, 19]. The concern of this paper is signature files.

A signature file consists of the signatures of objects which are simply special encodings, represented by bit strings, which indicate the essence of the stored data items [17]. Typically, the size of a signature file is about ten to twenty percent of the original database [4, 19]. In IR, signature files act as a filter to discard a large number of non-qualifying records. In the paper, the words "record," "logical block," and "object" will be used to signify the data items stored in the database.

During query processing, query signatures are generated like object signatures to reflect the content of each query. Next, the signature file is searched to find the qualifying data items. The retrieval using signature files is very efficient [7]. Furthermore, signature files are suitable for dynamic environments where object insertions and deletions are very common [19]. An incomplete list of the application domains for signature files includes computerized libraries, office automation, electronic encyclopedias, integrated manufacturing systems, prolog knowledge bases, and geographic information systems [9, 16].

In this paper we consider a signature file organization method, linear hashing with superimposed signatures (LHSS), which is designed for dynamic databases [18, 19]. This study is an analysis of the effects of reflecting the term occurrence and query frequencies to signatures in LHSS environment. The proposed approach relaxes the unrealistic uniform frequency assumption [18, 19] and lets the terms with high discriminatory power set more bits in signatures. The paper provides the derivation of the performance evaluation formulas to measure the efficiency of the new approach and presents the design and results of a set of simulation experiments. The experiments prove that the new approach improves the retrieval efficiency. Also discussed are the further benefits of the approach to alleviate the potential imbalance between the levels of efficiency and relevancy.

The paper is organized as follows. A review of the signature extraction and file organization methods is provided in Section 2. Section 3 explains LHSS and its performance evaluation formulas. Section 4 discusses the drawbacks of LHSS due to its equal treatment of terms regardless of their occurrence and query frequencies. Section 5 explores the potential improvement in retrieval efficiency of LHSS considering term characteristics. Section 6 and 7, respectively, provide the simulation experiments for performance evaluation and their discussion. Section 8 presents the conclusion of the study.

2. SIGNATURE FILES: AN OVERVIEW

There are two main concerns of signature applications: signature extraction and signature file organization. During the retrieval process, due to information loss in signature extraction, some object signatures seem to qualify a particular query even though the corresponding data objects do not. This situation is known as false drop or false match. False drops create unnecessary disk accesses since data objects whose signatures seem to meet the query specifications are accessed anyway. A considerable amount of research has been devoted to estimating and minimizing the false drop probability in different signature extraction techniques to improve the overall system performance [4, 7].

Two basic types of signature extraction methods are word signatures (WS) and superimposed signatures (SS) [4, 7, 16]. In WS, data elements (e.g., words of a document) are represented by bit strings which are later concatenated to form the object signatures. The query signature is generated in a similar manner and then matched with the object signatures to see if they contain the query terms.

In SS, on the other hand, the database is divided into logical blocks which contain the same number of unique nontrivial terms. Each term is hashed to a bit string of the same length which is called the term signature. Term signatures are then superimposed (ORed) to form the block signature. Similarly, a query signature is formed by superimposing all term signatures specified in the query. A block qualifies a query if all bit positions that are set in the query signature are also set in the block signature. (Figure 1 provides examples of WS and SS signature extraction methods.)

...hypertext hypermedia applications...

word signatures	0010	0110	1100
object signature		001001101100	

a. Word signature (WS) extraction.

...hypertext hypermedia applications...

terms	term signatures	
hypertext	1000 1000	
hypermedia	0010 0100	
applications	1000 1000	
	============	
	1010 1100	<==== logical block signature

b. Superimposed signature (SS) extraction.

Figure 1. Signature extraction methods.

There are two basic approaches to signature file organization: single-level and multilevel structures. In the single-level case, every signature (or a part of it) should be examined to check whether it qualifies the query specification or not. This is the method applied in sequential and bit-slice file organizations [7, 14, 16].

The main advantage of the sequential organization is its simple structure which facilitates exhaustive searching and easy insertion. However, it is totally inappropriate for partial match retrieval based on secondary attributes. The retrieval performance drops severely as the database size increases.

Bit slice is a transposed file organization in which block signatures are stored in bit-slices rather than bit-strings. The main purpose is to decrease the query response time by accessing to the bit positions specified by the query rather than reading the whole signature. However, improvement in retrieval efficiency is limited since the number of disk accesses required to process a query increases with the query weight [14]. Additionally, since the individual bits of a particular signature can be stored far apart from each other, the insertion and deletion cost is high.

Multilevel signature file organizations consist of levels of signatures where the nature and content of an upper level signature is dependent on a predetermined number of lower level signatures [6, 16]. Hence, the higher levels act as a coarse filter that eliminates unnecessary disk accesses through the pruning mechanism.

S-tree is one such organization. However, in this case insertions and deletions are difficult because not only the original signatures in the lowest level of the tree but also those on the higher levels which are constructed by superimposing a group of lower level signatures need to be modified. Furthermore, when the number of records in the database increases, the higher level signatures get cluttered impairing the retrieval efficiency.

The S-tree structure can be improved by grouping similar signatures and by organizing them within a B-tree structure. However, for large databases, this approach becomes too slow because of the growing overhead in the tree structure [9].

A two level organization proposed by Sacks-Davis and Ramamohanarao [14] suggests creating block descriptors in addition to record descriptors and keeping the former ones in a bit-slice structure while organizing the latter descriptors in a sequential manner. In this approach a block descriptor is typically longer than a record descriptor since a block is a collection of records. Although the method is appropriate for large databases, its speed is dependent on the number of matching records for a query [14].

The need to reduce the search space is a major issue in signature files which becomes critical when the database size increases. To help solve this problem various partitioning schemes are presented in the literature [9, 13, 18, 19]. Partitioning provides significant savings in the search space, improves retrieval efficiency and is appropriate for parallel processing environments [9, 19].

3. LINEAR HASHING WITH SUPERIMPOSED SIGNATURES (LHSS)

Linear hashing is an efficient method for the organization of partitioned dynamic files [11, 17]. Its application to signature files has been studied in the literature [13]. A related method, which is originally introduced by Zezula, is linear hashing with superimposed signatures [18].

3.1 The Method

LHSS provides a method of mapping signatures to storage pages and processing the queries to find qualifying signatures. The corresponding file structure is depicted in Figure2.

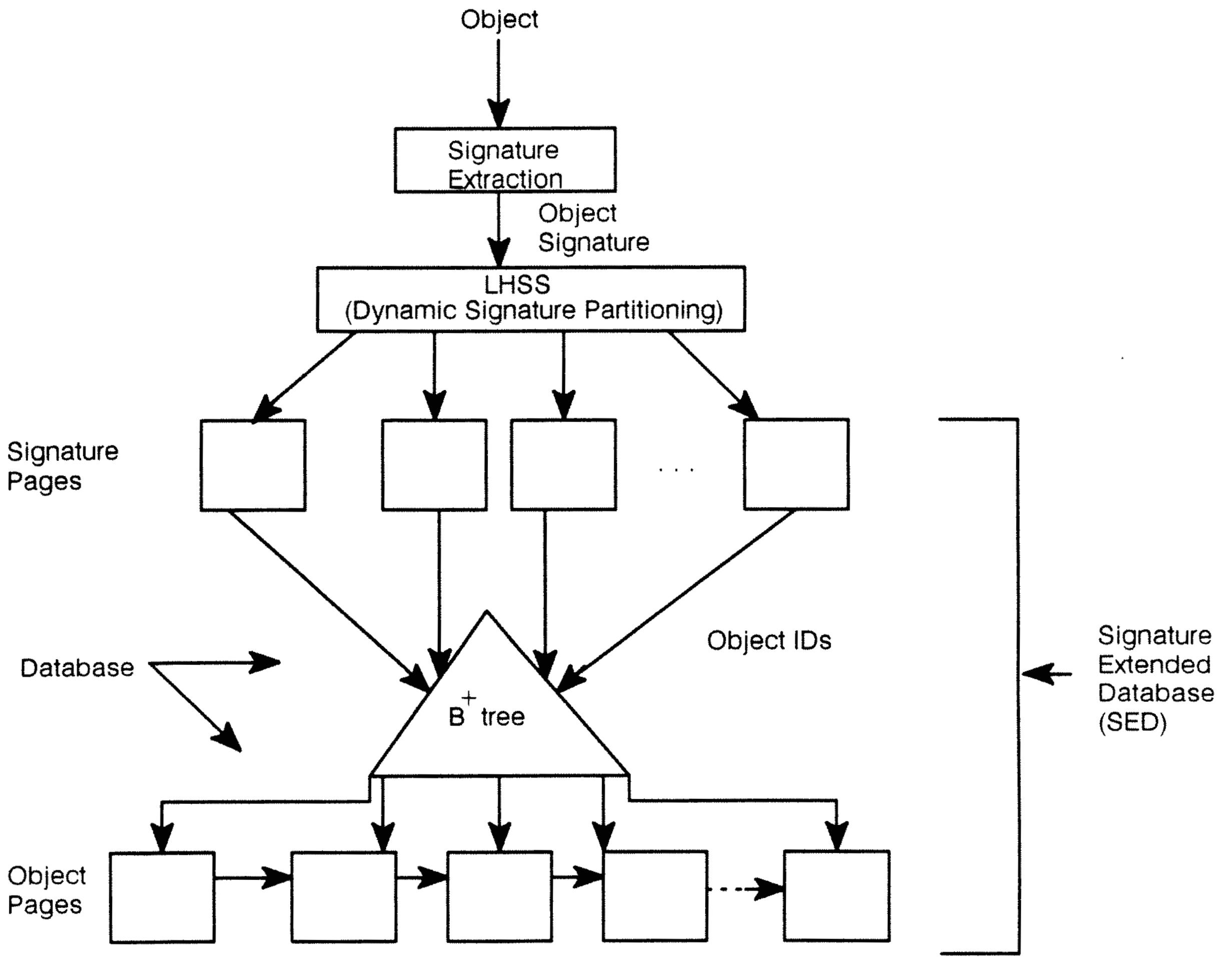


Figure 2. LHSS based database organization.

3.1.1 Signature File Creation

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. The hashing function is defined as follows [18, 19].

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

where

b_i : value of the ith binary digit of the object signature

f : signature size in bits

h : hashing level

n : number of addressable (primary) pages

s_i : object signature i

For the initial condition h= 0, n= 1, and $g(s_i, 0, 1)$ is defined as 0. (For easy reference the definition of the important symbols of the paper is provided in Table I.)

Table I. Definition of Important Symbols

b_i	: value of the ith binary digit of the term signature
f	: size of an object signature in bits
h	: hashing level
m	: no. of bits a term sets to 1 (when each term sets the same no. of bits in the query signature)
mi	: no. of bits set by each term of the ith term subset
n	: no. of addressable pages
qi	: probability that a single query term is from Si
s_i	: ith object signature
w(Q)	: weight of query, i.e., the no. of 1s in query signature
D	: expected no. of distinct terms in a record
Di	: expected no. of distinct terms of set Si in a record
F_d	: false drop probability
N(n, h, w(Q))	: no. of pages that do not need to be accesses
P(j)	: probability that j bits are set in the h-bit suffix of the query
P(w(Q), h)	: probability of access savings
R(h)	: no. of pages hashed at level h
Si	: set i of terms with similar discriminatory power

In simple terms, the hashing function, g, uses the last h or (h-1) bits of a signature to determine the number of the addressable 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 by the formula SP = (SP + 1) mod 2^{h-1}.

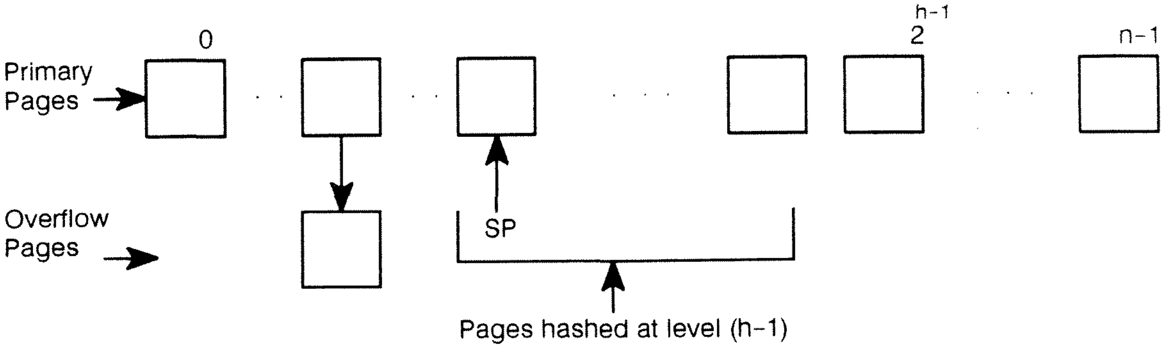


Figure 3. Linear hashing file structure ($2^{h-1} < n \leq 2^h$, $0 \leq SP < 2^{h-1}$).

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). A pictorial representation of a signature file generated by the above process is provided in Figure 3. Note also that linear hashing is space efficient and does not lead to many overflows [11].

3.1.2 Query Processing

The particular hashing function used and the resulting partitioned file organization enables reduction of search space during query processing. The query signatures are created in the same way as the object signatures, but typically they have lower weights (i.e., less number of 1s). The same hashing function is applied to the query signatures. The output is a page number that specifies the beginning of the search space. The associated page signature is checked against the query signature. If it qualifies, that page is accessed, if it does not, the process is repeated using the next page signature until end of file is encountered.

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 (see Figure 4).

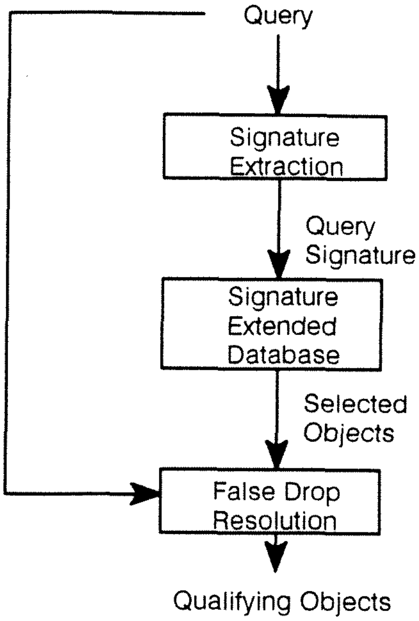


Figure 4. Query processing in signature extended databases (SEDs).

3.2 Performance Evaluation

Zezula and his coworkers have shown that in LHSS the degree of search space reduction achieved depends on the query signature weight w(Q), hashing level h, signature size f, and the number of addressable pages n [19].

Below we will give an overview of their performance evaluation formulas and will later use them as the basis for further analysis.

Our aim is to find the number of page savings as a function of the number of addressable pages (n), the hashing level (h), and the query weight (w(Q)). We assume that the signature size is fixed at f.

The first step is to find the expected number of bits in the h-bit suffix of the query, which is a function of the query weight and the hashing level. Intuitively, it is easy to see that as the weight of the query increases, the probability that we will have more number of 1s in the h-bit suffix of the query signature also increases. Besides, higher values of h will take more bits into account and hence will most probably contain more number of 1s.

Let E[no. of bits (w(Q), h)] be the expected number of bits set in the h-bit suffix of the query signature.

$$\text{E[no. of bits (w(Q),h)]} = \sum_{j=1}^{\min[h,\, w(Q)]} j\ P(j) \qquad (2)$$

where P(j) is the probability that j bits are set in the h-bit suffix of the query and can be written as follows.

$$P(j) = \frac{\binom{f-h}{w(Q)-j}\binom{h}{j}}{\binom{f}{w(Q)}} \qquad (3)$$

The next step is to find the probability of access savings, P(w(Q), h), which is also a function of the query weight and the hashing level. This is the proportion of the number of pages that do not need to be accessed (while processing a particular query) to the total number of addressable pages. Therefore, it is equal to the following expression.

$$P(w(Q), h) = 1 - \frac{npa}{n} \qquad (4)$$

where

npa : number of pages accessed,

n : number of addressable pages in the signature file.

Recall that E[no. of bits (w(Q), h)] stands for the expected number of bits set in the h-bit suffix of the query signature. This means that only $2^{h\text{-E[no. of bits(w(Q), h)]}}$ number of pages need to be accessed.

It follows that when $2^h = n$

$$npa = \frac{n}{2^{\text{E[no. of bits (w(Q), h)]}}} \qquad (5)$$

and therefore

$$P(w(Q), h) = 1 - \frac{n / 2^{\text{E[no. of bits (w(Q), h)]}}}{n}$$

$$= 1 - \frac{1}{2^{\text{E[no. of bits (w(Q), h)]}}} \qquad (6)$$

The final step is to find an expression for, N(n, h, w(Q)), the total number page savings. First, let revisit the working mechanism of the LHSS and express the number of addressable pages hashed with level h, R(h), and those hashed with level h-1, R(h-1).

When n= 2^h, SP = 0 and all pages are hashed at level h. Under this condition, as soon as a page split takes place, the value of h is increased by 1 and both page 0 and the new page are rehashed at this level, since each split results in the rehashing of two pages, R(h) can be defined as

$$R(h) = 2\,(n - 2^{h-1}) = 2n - 2^{h}$$

where 2^{h-1} is the number of addressable pages when all pages are hashed at level h-1. The difference between n and 2^{h-1} indicates the number of page splits that have taken place since then. Each split results in the rehashing of two pages, so the multiplication of the number of splits by two gives the number of pages hashed at level h.

It follows that

$$R(h-1) = n - R(h) = 2^{h} - n$$

Finally, the total number of page savings, N(n, h, w(Q)), is defined as the number of pages that need not be accessed for a given query and can be expressed as follows.

$$N(n, h, w(Q)) = R(h)\,P(w(Q), h) + R(h-1)\,P(w(Q), h-1) \qquad (7)$$

4. EFFICIENCY / RELEVANCY CONSIDERATIONS

LHSS provides significant retrieval efficiency without creating considerable storage overhead [18, 19]. It is especially suitable for the multimedia environment where the query weight is high [19.] However, a major disadvantage of LHSS is the decrease in page savings when the query weight is low. The lower the number of bits set to 1 in the h-bit suffix of the query signature, the lower the retrieval performance. LHSS treats all terms equally regardless of their occurrence and query frequencies, creating an imbalance between efficiency and relevancy. This is because different levels of relevancy can be observed for the same level of efficiency. This issue will be discussed and the related studies in the literature will be mentioned below.

Croft and Savino's research [5] on efficiency and effectiveness suggests that inverted indexes are usually superior to signature files in text retrieval effectiveness. (In inverted index search, associated with each term there is a list of <object id, weight> pairs for each object in which that term appears, and the similarity of all database objects is then determined by traversing the query term list [1, 15].) However, because their experimental design includes comparison of inverted indexes with simple signature file organizations such as sequential and bit-slice, we find it useful to explore how we can possibly create the efficiency-relevancy balance when dynamically partitioned file organizations, like LHSS, are used.

The effectiveness concern brings out the following question: can we modify the signature extraction techniques to take into account the occurrence and query frequencies of the terms so that we can accomplish a balance between efficiency and relevancy ?

When SS is used, the weight of a single term query remains the same regardless of the occurrence and query frequency of the term specified in it. However, it is known that the terms that appear frequently in the database are not used in the queries because of their low discriminatory power. Conversely, those terms with low occurrence frequency appear frequently in the queries because of their ability to eliminate most of the irrelevant documents that do not contain them and hence are of no interest to the user.

The idea of treating terms differently depending on their discriminatory power has been used for document retrieval [2, 15]. In document indexing, for instance, the terms are assigned weights depending on their term and inverse document frequencies. Those terms with high weights are eligible to be used as index terms which in turn form the basis for query-document matching [15, p. 306].

A method based on term discrimination and signature clustering is proposed by Chang and his coworkers [4]. They assign different file structures to terms depending on their discriminatory power.

Another approach is proposed by Faloutsos and Christodoulakis [8]. It suggests applying different treatment to the terms with high discriminatory power by letting them set more bits. The incorporation of this idea to LHSS is the subject of the next section.

5. LHSS BASED ON TERM CHARACTERISTICS

Now we will explore the potential improvement in the retrieval efficiency of LHSS using a signature extraction method which considers the term occurrence and query frequencies. When there is a significant difference among the discriminatory power of the terms in the database, this difference should be accounted for in the signature extraction method to improve the efficiency. On the other hand, using the standard superimposed signature generation method might be more efficient when the term frequencies are approximately uniform.

In this section, the performance of LHSS is analyzed using two different superimposed signature extraction methods. In the first method, which will be referred to as "single m case," where m stands for the number of bits set by each term, standard superimposed signatures are used. In the second method, which will be referred to as "multiple m case," terms are grouped into disjoint subsets based on their discriminatory power. The number of bits set by the terms in one set is the same and is a function of the occurrence and query frequencies. These two cases will be called SM and MM, respectively.

5.1 Detailed Analysis

This section provides the derivation of the LHSS performance evaluation formulas for SM and MM. The next section provides the details and results of our experiments which are based on the derived formulas.

Faloutsos and Christodoulakis [8] have suggested partitioning all possible terms in the database into ns disjoint subsets S1,S2,...,Sns according to their discriminatory power where

$$S1 \cap S2 \cap S3 \cap ... \cap Sns = \varphi \quad \text{and}$$
$$S1 \cup S2 \cup S3 \cup ... \cup Sns = S$$

where S is the set of all terms in the database.

Furthermore,

qi : probability that a single term query is from Si and

Di : expected number of distinct terms of set Si in a record

where

$$\sum_{i=1}^{ns} qi = 1 \quad \text{and} \quad \sum_{i=1}^{ns} Di = D$$

and

D : expected number of distinct terms in a record.

5.1.1 SM Case

In this case, each term sets the same number of 1s in a signature. The optimal value of the number of bits set by each term, m, can be computed as [8]

$$m = \frac{f \ln 2}{D} , \quad \text{where f : signature size} \qquad (8)$$

when there is no partitioning or when

$$\frac{q1}{D1} = \frac{q2}{D2} = ... = \frac{qns}{Dns}$$

is satisfied.

This is the value of m which minimizes the false drop probability. Equation (8) will be used to find the number of bits set by each term in the SM case. Next, the total number of page savings will be computed.

We observe that the weight of a single term query will be equal to m since each term sets the same number of bits in the query signature. That is to say

$$w(Q) = m = \frac{f \ln 2}{D}$$

Substituting the value of w(Q) in the performance evaluation formulas for LHSS, the expected number of bits in the h-bit suffix of the query signature can be computed as follows.

$$E[\text{no. of bits}(m, h)] = \sum_{j=1}^{\min[h,m]} j\, P(j)$$

where

$$P(j) = \frac{\binom{f-h}{m-j}\binom{h}{j}}{\binom{f}{m}}$$

Next, the value of E[no. of bits (m, h)] is plugged into equation (6) to compute the probability of access savings, which in turn is substituted into equation (7) to find the total page savings.

5.1.2 MM Case

For this case, the optimal number of bits set by each term in the set Si can be computed as follows [8].

$$mi = \frac{f \ln 2}{D} + \frac{1}{\ln 2}\left[\ln\frac{qi}{Di} - \frac{\sum_{i=1}^{ns} Di \ln\frac{qi}{Di}}{D}\right] \qquad (9)$$

where

$$\sum_{i=1}^{ns} qi = 1 \quad \text{and} \quad \sum_{i=1}^{ns} Di = D$$

These are the optimal mi values that minimize the false drop probability. It is also shown that when 80-20% rule holds, approximately 50% savings can be achieved by taking the term occurrence and query frequencies into account [8].

Once again, the query weight will be computed. This time, since the number of terms set by each term differ, the weight of a single term query can no longer be represented by a constant. However, we can use the number of bits set by each term together with its query frequency to derive an expression for the expected query weight.

$$E[w(Q)] = \sum_{i=1}^{ns} qi\; mi \qquad (10)$$

where

E[w(Q)] : expected value of the weight of a single term query.

Next, the expected number of 1s in the h-bit suffix of the query will be computed. This can be defined as:

$$E[\text{no. of bits}] = \sum_{i=1}^{ns} qi\; E[\text{no. of bits}(mi,h)] \qquad (11)$$

where

$$\text{E[no. of bits (mi,h)]} = \frac{\binom{f-h}{mi-j}\binom{h}{j}}{\binom{f}{mi}}$$

The value of E[no.of bits] is used in equation (6) to compute the probability of access savings, which in turn is substituted into equation (7) to find the total page savings.

We also define the percent savings, PS, as the ratio of the total number of savings to the total number of accessible pages multiplied by 100. That is

$$PS = \frac{N(n,h,w(Q))}{n} \times 100$$

In the following experimental analysis, PS is used as the measure of the performance of the retrieval efficiency.

6. EXPERIMENTAL EVALUATION

This section provides four sets of experiments which compare the performance of SM and MM under various conditions.

Experiment 1.

Purpose

The purpose of this experiment is to compare the performance of SM and MM in terms of retrieval efficiency measured by percent savings (PS) at different qi/Di combinations.

Parameters

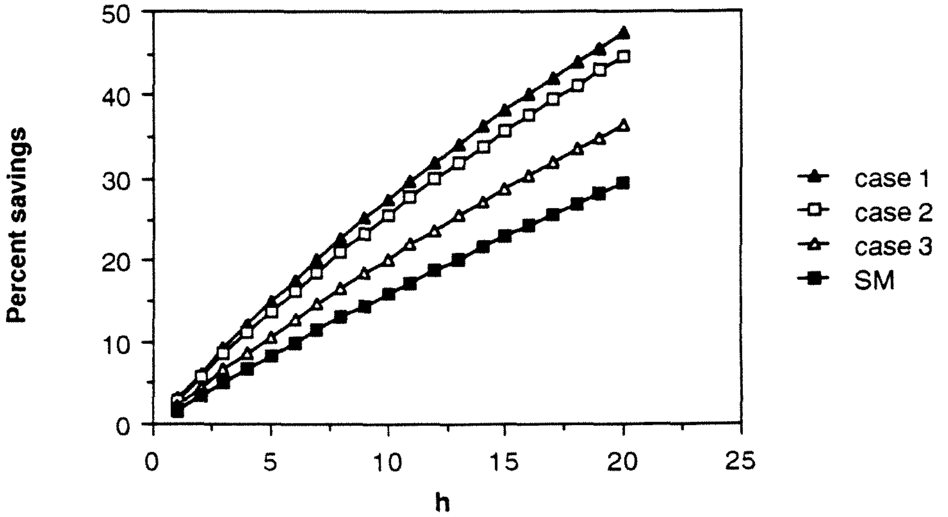
The values of the input parameters are as follows.

D1 = 11 D2 = 14 f = 80

The Di values are kept constant while changing the values for qis to create different Di/qi combinations. To make our analysis tractable, the number of distinct terms in a record is kept below a reasonable limit and a consistent value for the signature length is used. This, however does not create a loss of generality but helps us get rid of the unnecessary complexity. (We also make use of the 80-20 % rule which is a representative of many real life situations.) In this experiment, we have three cases: case1, 2, and 3. The q1 and q2 values for these cases, respectively, are (0.90, 0.10), (0.80, 0.20), and (0.60, 0.40).

Results

The results of the experiment are summarized in Figure 5. The performance of SM is constant in all cases regardless of the changes in the qi values. This is because the value for the optimal m depends on D only (see equation 8). Recall that D = (D1+ D2 +...+ Dns) and is kept constant. This provides us a reference line against which we can compare the performance of MM. The MM cases prove to be more efficient than the SM case. The outcome is consistent with our expectations: treating terms differently depending on their discriminatory values enables us to access fewer pages when the query contains a term with high discriminatory value. The savings are relatively low with terms with low discriminatory power but this does not affect the overall performance since such terms are rarely specified in the queries.



D1=11, D2=14, f=80
(q1, q2) values for case 1, case 2 and case 3: (0.90,0.10), (0.80,0.20), (0.60,0.40)
Figure 5. Percent savings versus hashing level.

Figure 5 also indicates that the percent savings increase with the hashing level for both SM and MM cases. However, since MM provides more savings at each level of h, it is clearly advantageous.

The high performance of MM can be accounted for the significant difference between the discriminatory power of the terms. Analytically speaking, qi/Di ratios are significantly different for the two disjoint subsets S1 and S2. Hence we are better off when we adjust our signature extraction method to treat terms differently.

Notice that the performance of MM improves as the difference between the query frequencies increases, provided that the occurrence frequencies are kept constant.

Experiment 2.

Purpose

We have already observed that when (D1/q1 = D2/q2 = ...= Dns/qns) is satisfied, all terms set the same number of bits, hence MM converges to SM (see Section 5.1).

Our concern in this experiment is to check if it is possible to make generalizations on the amount of savings that can be achieved at particular Di/qi levels.

Parameters

Table II shows the Di and qi values for each of the four selected cases.

Let

Dij = value of Di in the jth case

qij = value of qi in the jth case

where $1 \leq i \leq 2$, $1 \leq j \leq 4$.

Table II. Parameters of Experiment 2

case no.	D1	q1	D1/q1	D2	q2	D2/q2
1	10	0.2	50	40	0.8	50
2	30	0.6	50	20	0.4	50
3	10	0.8	12.5	14	0.2	70
4	5	0.4	12.5	42	0.6	70

Results

The comparisons of case 1 and case 2 in terms of percent savings is provided in Table III.

Table III. Percent Savings when D11/q11 = D21/q21 = D12/q12 = D22/q22 = 50

	case 1		case 2	
h	SM	MM	SM	MM
5	4.24	4.24	4.24	4.24
10	8.30	8.30	8.30	8.30
15	12.19	12.19	12.19	12.19
20	15.91	15.91	15.91	15.91

Note that

D11/q11 = D21/q21 = D12/q12 = D22/ q22 = 50

and therefore not only MM case converges to SM within each case, but also the performance of case 1 turns out to be the same as that of case 2.

Table IV. provides the comparison between case 3 and case 4.

Table IV. Percent Savings when D13/q13 = D14/q14 = 12.5, D23/q23 = D24/q24 = 70

	case 3		case 4	
h	SM	MM	SM	MM
5	8.30	13.70	4.24	7.50
10	15.91	25.52	8.30	14.44
15	22.89	35.72	12.19	20.86
20	29.29	44.52	15.91	26.80

Note that this time

D13/q13 = D14/q14 = 12.5 ≠ D23 /q23 = D24/q24 = 70

Within each case, the discriminatory power of the terms is different, hence MM gives different results compared to SM. The corresponding Di/qi values are equal for cases 3 and 4 but this does not guarantee that the percent savings will be identical. In fact the savings in case 3 are higher than those in case 4.

A closer look at the occurrence and query frequencies in both cases reveals the following: in case 3, higher query frequency is coupled with lower occurrence frequency which is desired, whereas in case 4, the high query frequency is coupled with a relatively high occurrence frequency which degrades the overall performance. Although the latter case is not typical in real life, the analysis shows that not only the value of the Di/qi ratio but also the relative size of its components determine the level of savings.

Experiment 3.

Purpose

The purpose of this experiment is to observe the effect of the signature size on the amount of savings provided by MM.

Parameters

The values of the input parameters are as follows:

q1 = 0.8	D1 = 11
q2 = 0.2	D2 = 14

Three cases will be considered; associated f values are 80, 160, 240, respectively.

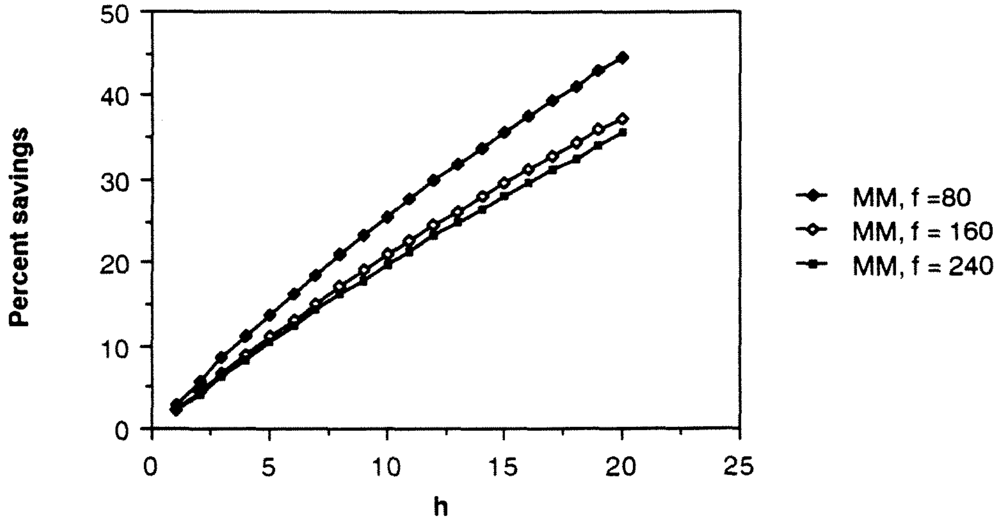


Figure 6. Percent Savings in MM case for different signature sizes.

Results

Figure 6 shows that the percent savings decrease as the signature size increases as indicated by equation (9). Notice that, since qi and Di values are kept constant throughout the experiment, the second operand on the right hand side of this equation becomes a constant. Therefore:

$$mi = \frac{f \ln 2}{D} + K$$

where K is a constant.

The above equation shows that as the value of f is doubled, the mi values also increase, but with a lower rate. Consequently, the proportion of the bits set in a single term query becomes slightly lower. This, in turn, results in a relatively low reduction in the search space which accounts for the drop in percent savings. However, the associated false drop probability is lower due to the decrease in the degree of overlapping and cluttering in the signatures. Therefore, the choice of the signature size depends on the compromise between the required percent savings and the allowable false drop rate.

Experiment 4.

Purpose

The purpose of this experiment is to observe the change in percent savings when the terms are grouped into three disjoint subsets rather than two. The cases with two and three subsets are referred to as MM2 and MM3, respectively.

Parameters

Four cases will be considered (1A, 1B, 2A, 2B) The values for the input parameters for all cases are provided in Table V. The signature size, f, is constant and equals to 80 throughout the experiment.

Table V. Parameters for Experiment 4.

case 1.A		case 1. B		
D1 = 25	D2 = 30	D11 = 5	D12 = 20	D2 = 30
q1 = 0.90	q2 = 0.10	q11 = 0.70	q12 = 0.20	q2 = 0.10
case 2.A		case 2.B		
D1 = 25	D2 = 30	D11 = 10	D12 = 15	D2 = 30
q1 = 0.95	q2 = 0.05	q11 = 0.80	q12 = 0.15	q2 = 0.05

In cases 1.A and 2.A, terms are grouped into two disjoint subsets based cn their discriminatory power. In fact, subset S2 includes all non-discriminatory terms. (The value of q2 equals 0.10 in case 1.A and 0.05 in case 2.A.. The corresponding D value is 30 for both cases.) The "rest" of the terms which have relatively high discriminatory power are all grouped together in S1. Within S1, no special treatment is applied to the member terms; they all set equal number of bits.

In contrast, in cases 1.B and 2.B, the terms in S1 are analyzed even further and are divided into two subgroups based on their occurrence and query frequencies. This detailed analysis enables us to discriminate the terms with high discriminatory power from those with medium discriminatory power.

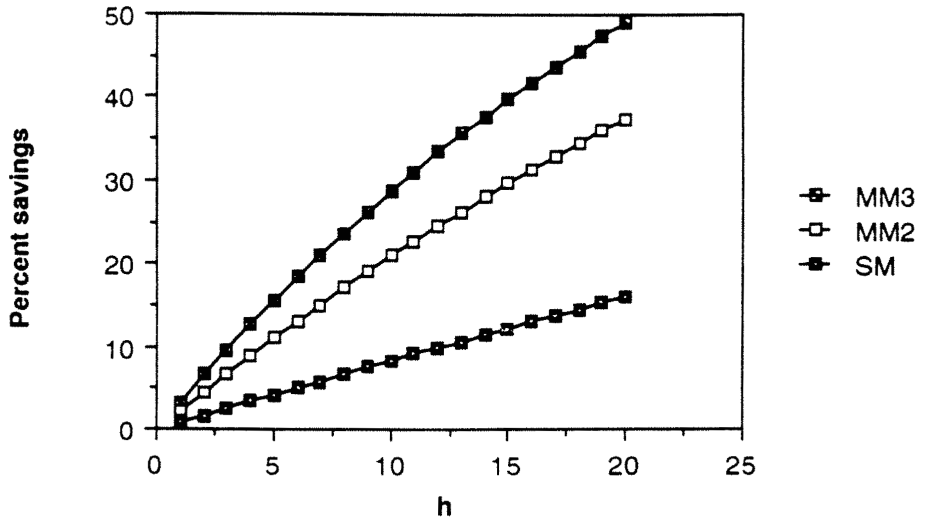


Figure 7. The comparison of SM, case 1.A (MM2), and case 1.B (MM3).

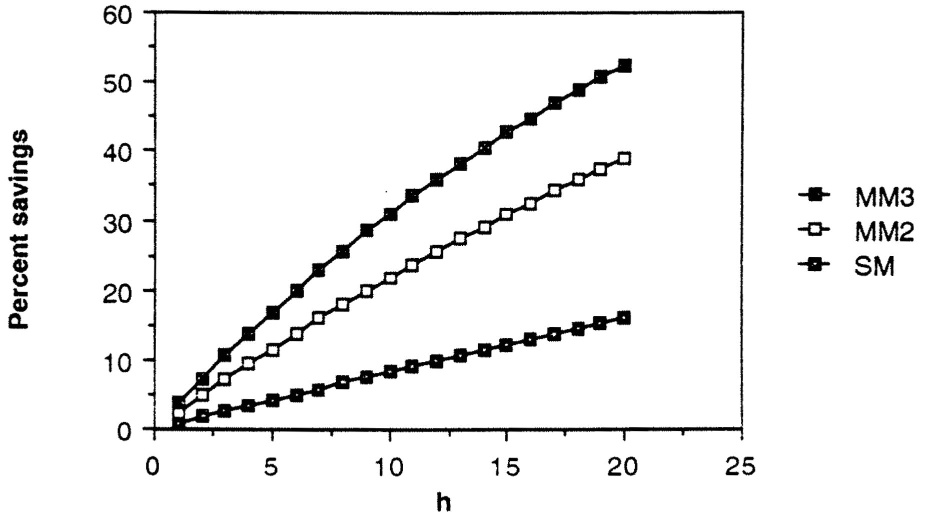


Figure 8. The comparison of SM, case 2.A (MM2), and case 2.B (MM3).

Results

Figures 7 and 8 clearly indicate that MM3 outperforms MM2 in both cases. Note that the savings achieved through MM3 and MM2 are far above the savings provided by SM. Note also that the slope of the line that represents the behavior of MM3 is higher than that of MM2 which is apparently higher than that of SM. This means that the amount of increase in percent savings for an incremental increase in the hashing level is considerably high in MM3, relatively modest in MM2 and significantly lower in SM.

Between the two cases where MM2 is used, the savings in case 2.A outperform those in case 1.A. Similarly, within the two MM3 cases, the savings in case 2.B are higher than those in case 1.B. This is because the distinction among the term discriminatory power values is sharper in cases 2.A and 2.B, which enables these two cases to provide the highest savings for MM2 and MM3, respectively.

The comparison of MM2 and MM3 cases indicate that it is worthwhile to apply more specific treatment to terms as long as there really is a significant difference among their discriminatory power values.

7. DISCUSSION

Our experimental analysis shows that the retrieval efficiency can be improved when the occurrence and query frequencies of the terms are taken into account in determining the number of bits set by each term. The savings are particularly apparent when the difference between the discriminatory power values of the terms is significant.

When SM is used, the number of bits set by each term is identical. When a single term query is specified in a query, the query weight is constant and equals m. Hence 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.

Now, let us observe how MM can alleviate the level of this imbalance: when MM 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 [2; 15, p. 299].

8. CONCLUSION

Signature files act as a filter on retrieval to discard a large number of non-qualifying data objects. Linear hashing with superimposed signatures, LHSS, provides an efficient retrieval filter to process queries in dynamic databases [18, 19]. In LHSS each term sets a fixed number of bit positions in the signatures, regardless of their query and occurrence frequencies.

This study provides an analysis of the effects of relaxing the unrealistic uniform frequency assumption and applying different treatments to terms based on their occurrence and query frequencies. In this 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.

Our experiments explore the amount of page savings with different query and occurrence frequency combinations at different hashing levels. The results show that the performance of LHSS improves with the hashing level and the larger is the difference between the term discriminatory power values of the terms, the higher is the retrieval efficiency. The choice of the signature size depends on the compromise between the required percent savings and the tolerable false drop rate. In the paper we also discuss the benefits of this approach to alleviate the imbalance between the levels of efficiency and relevancy in unrealistic uniform frequency assumption case.

A very recent study provides a new formula to compute the optimal values for the number of bits set by each term that minimize the false drop probability [10]. In this approach the weight of each signature is controlled to a constant rather than keeping the "expected signature weight" constant. By shifting the focus from "expected values" to "exact values" provide simple and efficient formulas. Furthermore, the false drop probability is lower. This new optimal approach can also be used with dynamic signature file organizations to improve search efficiency.

REFERENCES

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

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

3. Chang, J. W., Lee, J. H., Lee, Y. J. Multikey access methods based on term discrimination and signature clustering. In *Proceedings of the 12th Annual ACM-SIGIR Conference* (Cambridge, Mass., June 1989) ACM, N.Y., 1989, pp. 176-185.

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

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

6. Deppisch, U. S-tree: A dynamic balanced signature index for office retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval* (Pisa, Sept. 8-10, 1986) ACM, N.Y., 1986, pp. 77-87.

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

8. Faloutsos, C., Christodoulakis, S. Design of a signature file method that accounts for non-uniform occurrence and frequency frequencies. In *Proceedings of the 11th International Conference on VLDB* (Stockholm, Aug. 1985). VLDB Endowment, 1985, pp. 165-170.

9. Lee, C.-W., Leng. C.-W. Partitioned signature files: design issues and performance evaluation. *ACM Transactions on Information Systems.* 7, 2 (Apr. 1989), 158-180.

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

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

12. Lloyd, J. W., Ramamohanarao, K. Partial-match retrieval for dynamic files. *Bit.* 22, (1982), 150-168.

13. Ramamohanarao, K., LLoyd, J. W., Thom, J. A. Partial-match retrieval using hashing and descriptors. *ACM Transactions on Database Systems.* 8, 4 (Dec. 1983), 552-576.

14. Sacks-Davis, R., Ramamohanarao, K. "Multikey Access Methods Based on Superimposed Coding Techniques." *ACM Transactions on Database Systems.* 12, 4 (December 1987), 655-696.

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

16. Tiberio, P., Zezula, P. Selecting signature files for specific applications. In *Proceedings of Advanced Computer Technology, Reliable Systems and Applications, 5th Annual European Conference.* (Bologna, Italy, May 1991) IEEE, 1983, pp. 718-725.

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

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

19. Zezula, P., Rabitti, F., Tiberio, P. "Dynamic Partitioning of Signature Files. *ACM Transactions on Information Systems.* 9, 4 (Oct. 1991), 336-367.