S. P. Ghosh
C. T. Abraham

Application of Finite Geometry in File Organization for Records with Multiple-Valued Attributes

Abstract: The schemes for organizing binary-valued records using finite geometries have been extended to the situation in which the attributes of the records can take multiple values. Some new schemes for organizing records have been proposed which are based on deleted finite geometries. These new schemes permit the organization of records into buckets in such a manner that, by solving certain algebraic linear equations over a finite field, it is possible to determine the bucket in which records, pertaining to two given values of two different attributes, are stored. Since the bucket identification required for the storage of record accession numbers is based on the combination of attribute values, the file does not require any reorganization as new records are added. This is a definite advantage of the proposed schemes over many key-address transformation procedures wherein the addition of new records may lead to either a drastic revision of the file organization or significant reduction of retrieval effectiveness. The search time for the new schemes are very small in comparison to other existing methods.

1. Introduction

The problem of storing large data files of formatted records and retrieving a subset of records on the basis of some attributes that constitute the records is a difficult task, and much work has been done in this area. A summary of the work has been given by Abraham, Ghosh and Ray-Chaudhuri.[1] The problem becomes more difficult when the attributes can take multiple values and the retrieval process involves retrieving a subset of records on the basis of some values from different attributes. The authors (and Ray-Chaudhuri)[1] had given a solution to the problem based on finite geometries when binary-valued attributes are considered; in the present paper some other properties of finite geometries will be considered to solve the multiple-valued problem. The technique of forming subsets of records and identifying the subsets by algebraic equations, as discussed previously[1] will also be used in this paper, but the subsets will be formed in a different manner. The subsets will be formed by using some special properties of combinatorial configurations, which have been used in the past, to solve some problems of constructions of statistical designs (Bose and Nair[2]).

2. Balanced multiple-valued filing scheme

A large volume of data may be stored in different ways for different purposes. In many situations each item of data may be represented by an l-vector, each component of which is a number (an alphanumeric code) providing information about one of a set of l attributes $A_1, A_2, \cdots A_l$. Each item in the file will also have an identifying number i, different for different items. If ν_{ij} is the value for attributes A_j of the i^{th} item, then we shall call $a(i) = (\nu_{i1}, \nu_{i2}, \nu_{il})$ the attributes vector of the i^{th} item. This identifying number i together with the attribute vector $a(i)$ constitutes the record of the i^{th} item. The set of all records constitutes the file. We shall denote by M the number of records in this file.

A retrieval request or a query Q is a request to retrieve from the file, the subset of all records for which a certain subset of attributes possess certain specific values. A file organization scheme consists of arranging the records according to a scheme that will reduce the time needed for searching records for a given class of queries. The problem of file organization is fairly simple when queries relate to only one attribute. In this case, the most frequently employed scheme is the method of "inverted" lists, where for each attribute value a list of the identifying numbers of all records possessing that specific attribute value is formed and all queries involving several attributes are satisfied by comparison of the basic single-attribute value lists. However, when the basic lists get very large, retrieval times become correspondingly large. In addition, the starting address of contiguous locations of the storage where each list is maintained has to be obtained by a "table look-up" operation. As the number of attribute values becomes large, the table becomes too big to be located in the internal memory of a processor. Thus, the search of the table may become slow for certain files. The idea of inverted lists has been extended by many file-system implementers to include combinations of values of different attributes. But the same criticism of slow table look-up applies to these extended schemes.

The filing schemes we propose in this paper differ significantly from previous methods by providing the capability to computationally determine the storage locations of records or their identifying numbers, thus avoiding the time-consuming table look-up. Whereas the present paper discusses the retrieval problem relating to two attributes only, the more general situation of multiple attributes had been dealt with by the authors and Bose.[7]

In most computerized filing systems the records are stored in some comparatively slow storage device. The starting address of a segment of the storage device where the record is stored in its entirety is called the *accession number* of the record. A set of addresses of a comparatively faster memory or storage device is reserved for storing the accession numbers. Let this set be S. File organization schemes have two features. First, there is a rule, the *storage rule*, which defines the subset $\sigma(i)$ of the elements of S, where the accession number of the i^{th} record is stored. Again, there is a *retrieval rule* for finding out the elements of S, where accession numbers of the records pertaining to any given query relating to any subset of different attribute values are stored. It is to be noted that hardware associative memories or content addressable memories are excluded from the present discussion. From the foregoing it is evident that redundant storage of accession numbers is unavoidable in such filing schemes unless extra addresses or locations are added to S to provide "chaining" of addresses. Complete avoidance of redundancy can be achieved only by very complex chaining rules, which make retrieval very inefficient. A limited amount of chaining of a very simple nature will be used in the present procedure. Additional storage requirement due to redundancy and search time for balanced multiple-valued filing schemes will be discussed in later sections of this paper.

In this paper filing schemes will be defined for records containing l attributes and these attributes can take $n_1, n_2, \cdots, n_l$ values respectively. A Balanced Multiple-valued Filing Scheme (referred to here as BMFS) with parameters $(k, n_1, n_2, \cdots n_l, b)$ is defined to be an arrangement of records with l attributes where the vector of the number of values these attributes can take is given by $(n_1, n_2, \cdots, n_l)$, in b groups (buckets), which are not necessarily mutually exclusive and which satisfies the following properties:

(2.1) The number of records in a bucket will not be greater than the number of records in the whole file.

(2.2) Records pertaining to any k ($k \geq 2$) values of k different attributes will appear in one and only one bucket.

(2.3) To every bucket there corresponds one or more sets of systems of linear algebraic equations over one or more finite fields.

A balanced multiple-valued filing scheme with parameters $(k, n_1, n_2, \cdots, n_l, b)$ is said to be of order k and is denoted by BMFS$_k$.

The problem of construction of BMFS$_k$ can be considered as a combinatorial problem, in which n elements are arranged into a number of sets such that when the n elements are partitioned into l groups of sizes $n_1, n_2, \cdots, n_l$ then any k elements belonging to k different groups will always be contained in a set, whereas no two (or more) elements of the same group will be contained in any set. This combinational problem can be solved by using finite geometries when $k = 2$.*

3. Finite geometries

Finite projective geometry PG(N, p^n) and finite Euclidean geometry EG(N, p^n) will be extensively used in this paper and they will be defined in the following paragraphs.

• *Projective geometry*

In a finite projective geometry PG(N, p^n) of N dimension based on Galois field GF(p^n), where p is a prime integer, the points can be taken as $(N + 1)$-tuples $x = (x_0, x_1, \cdots, x_N)$ where $x_0, x_1, \cdots, x_N$ are elements of GF(p^n) and the $(N + 1)$-tuple $\rho x = (\rho x_0, \rho x_1, \cdots, \rho x_N)$ is regarded as the same point as x for any nonzero element ρ of GF(p^n). By definition the $(N + 1)$-tuple $(0, 0, \cdots, 0)$ is not regarded as a point. (See Ref. 4.)

A t-dimensional flat in PG(N, p^n) is defined by the set of points which satisfy the following $N - t$ independent linear homogeneous equations.

$$\left.\begin{array}{l} a_{10}x_0 + a_{11}x_1 + \cdots \qquad + a_{1N}x_N = 0 \\ a_{20}x_0 + a_{21}x_1 + \cdots \qquad + a_{2N}x_N = 0 \\ \cdot \\ \cdot \\ \cdot \\ a_{N-t0}x_0 + a_{N-t1}x_1 + \cdots + a_{N-tN}x_N = 0 \end{array}\right\},$$

where the a's are the elements of GF(p^n). Thus the points which satisfy one linear homogeneous equation define an $N - 1$ flat in PG(N, p^n) and a point in PG(N, p^n) satisfies N independent linear homogeneous equations. Hence a point is called a 0-flat, a line a 1-flat, a plane a 2-flat and so on.

Let ϕ (N, t, s) denote the number of t-flats in PG(N, s) where

$$\phi(N, t, s) = \frac{(s^{N+1} - 1)(s^N - 1)\cdots(s^{N-t+1} - 1)}{(s^{t+1} - 1)(s^t - 1)\cdots(s - 1)}$$

and $s = p^n$.

* Even though this paper deals with files wherein all attributes can have only the same number of values, combinatorial filing schemes had been developed for the situation where attributes have unequal values. For detailed information see Ref. (7).

The ϕ's satisfy the following condition:

$$\phi(N, t, s) = \phi(N, N - t - 1, s)$$
$$\phi(N, -1, s) = 1 \text{ (by definition).}$$

• *Euclidean geometry*

A point in an N-dimensional finite Euclidean geometry EG(N, s) based on a GF(s) is defined to be an ordered N-tuple $(x_1, x_2, \cdots, x_N)$, where $x_i \in$ GF(s), $i = 1, 2, \cdots, N$. The N-tuple $(0, 0, \cdots, 0)$ is also a point of EG(N, s).

The t-spaces $(0 \leq t \leq N - 1)$ of EG(N, s) are defined by nonhomogeneous equations. The set of points which satisfy the following $N - t$ independent linear equations from a t-space.

$$\left.\begin{array}{l} a_{10} + a_{11}x_1 + a_{12}x_2 + \cdots \qquad + a_{1N}x_N = 0 \\ a_{20} + a_{21}x_1 + a_{22}x_2 + \cdots \qquad + a_{2N}x_N = 0 \\ \cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\ \cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\ \cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\ a_{N-t0} + a_{N-t1}x_1 + a_{N-t2}x_2 + \cdots + a_{N-t,N}x_N = 0 \end{array}\right\}$$

The number of t-spaces in EG(N, s) is equal to $\phi(N, t, s) - \phi(N - 1, t, s) = s^{N-t}\phi(N - 1, t - 1, s)$. The other details of PG(N, s) and EG(N, s) will not be discussed and may be obtained from Carmichael,[4] Bose,[3] etc.

4. Constructions of BMFS$_2$ using PG(N, s) and EG(N, s)

THEOREM 1. *There exists a balanced multiple-valued filing scheme with parameters* $k = 2$, $n_1 = n_2 = \cdots = n_l = s$. $l = s^{N-1}$ *and* $b = s^{N-1}\{\phi(N - 1, 0, s) - 1\}$.

Proof: Consider a spread generated by lines (i.e., a set of disjoint lines which cover the geometry) in a EG(N, s) and delete it from the EG(N, s). In this deleted geometry the lines are identified with the buckets of BMFS$_2$. Each line of the spread corresponds to an attribute of the records and the points on a line of the spread correspond to the different values the particular attribute can take. For constructing the buckets of BMFS$_2$ the points on the lines of the deleted geometry are considered in pairs and if a record contains the pair of values corresponding to any pair of points on the line, then that particular record is stored in the bucket corresponding to that line. Duplication of records in a bucket is not permissible.

The number of points on a line in EG(N, s) is s, hence $n_1 = n_2 \cdots = n_l = s$. The number of lines in a spread of EG(N, s) $= s^{N-1}$, hence $l = s^{N-1}$. Thus the number of lines in the deleted geometry $= s^{N-1}\,\phi(N - 1, 0, s) - s^{N-1}$, hence $b = s^{N-1}\{\phi(N - 1, 0, s) - 1\}$.

In a EG(N, s) any two points determine a line; hence any pair of points can appear on one and only one line. Thus, any pair of points which appear on any given line of the spread will not appear on any line of the deleted geometry. Further a pair of points belonging to two different lines of the spread will appear on one and only one line of the deleted geometry. This establishes (2.2), with $k = 2$. Every line of EG(N, s) can be represented uniquely by a set of $N - 1$ independent linear equations over GF(s), hence (2.3) is satisfied.

As no duplication of records in a bucket is permitted, (2.1) is satisfied. This completes the proof.

THEOREM 2. *There exists a balanced multiple-valued filing scheme with parameters* $k = 2$, $n_1 = n_2, \cdots = n_l = s$, $l = \phi(N - 1, 0, s)$ *and* $b = \{\phi(N, 1, s) - \phi(N - 1, 0, s)\}$.

Proof: Consider a PG(N, s). In general it is not possible to obtain a set of lines which will form a spread of PG(N, s), but it is possible to obtain a set of lines which will form a partial spread of PG(N, s). (A partial spread of order k in PG(N, s) is defined to be a collection of k-flats which form a cover of the geometry and any two or more of these k-flats intersect in one and only one $(k - 1)$-flat). Thus for construction of a BMFS$_2$ a partial spread of order unity has to be deleted from PG(N, s). Suppose all the lines, which have one particular point in common, say the origin, are deleted from PG(N, s). This deleted geometry will have $\{\phi(N, 0, s) - 1\}$ points, $\{\phi(N, 1, s) - \phi(N - 1, 0, s)\}$ lines. The lines of the deleted geometry will correspond to the buckets of BMFS$_2$. The $\phi(N - 1, 0, s)$ lines of the partial spread will correspond to the $\phi(N - 1, 0, s)$ attributes of the records and the points on any one of these lines, excluding the origin, will correspond to the different values the attribute, corresponding to the particular line, can take. The records will be assigned to the buckets in the same manner as in the case of EG(N, s).

The remainder of the proof is similar to that of Theorem 1 and hence will be omitted.

Remark 1. BMFS$_2$ can be constructed even when n_i's are not equal. Only restriction needed is that s should be so chosen that $s \geq \max\{n_1, n_2, \cdots, n_l\}$.

Remark 2. BMFS$_2$ can also be constructed with $l \neq s^{N-1}$ or $l \neq \phi(N - 1, 0, s)$. In such situations an l' can be chosen such that $l' > l$ and $l' = s^{N-1}$ or $l' = \phi(N - 1, 0, s)$ for some N and s, and the same method may be applied.

Remark 3. BMFS$_2$ will involve large amounts of duplication of records but a considerable saving in storage space can be achieved by storing the actual data in some fixed location and storing only the accession number of records in the buckets.

Remark 4. Details of storing records have been discussed in a previous paper by the authors and Ray-Chaudhuri[1] and will not be discussed in this paper.

Remark 5. A BMFS$_2$ uses a deleted finite geometry; hence the number of buckets is less than that in a balanced filing scheme of order 2 (BFS$_2$) (Ref. 1) based on a finite geometry having same number of points.

Example 1. As an illustration, a data base which has three attributes, where each attribute can take three different values will be considered. Suppose that the i^{th} attribute can take the values $v_{i1}, v_{i2}, v_{i3}, i = 1, 2, 3$. The $BMFS_2$ for these data can be constructed using a EG(2, 3). The lines of this geometry are given by:

$$x_1 = c, x_2 = c, x_1 + x_2 = c, \quad \text{and} \quad 2x_1 + x_2 = c,$$
$$\text{where} \quad c = 0, 1, 2.$$

The points of this geometry are pairs, and for simplicity of representation they shall be written without separation commas between the coordinates, i.e., the point (x_1, x_2) shall be written as x_1x_2. Out of the 12 lines of the geometry we shall delete the lines corresponding to $x_1 = 0$, $x_1 = 1$, and $x_1 = 2$, which form a spread of the geometry. The points on the different lines of the deleted geometry are given by:

(00, 10, 20), (01, 11, 21), (02, 12, 22), (00, 12, 21),
(01, 10, 22), (02, 11, 20), (00, 11, 22), (01, 12, 20),
(02, 10, 21)

The points of EG(2, 3) will correspond to the different values the attributes can take, as follows:

$00 = v_{11}$, $01 = v_{12}$, $02 = v_{13}$, $10 = v_{21}$, $11 = v_{22}$,
$12 = v_{23}$, $20 = v_{31}$, $21 = v_{32}$, $22 = v_{33}$.

The buckets will be constructed by storing in them the accession numbers (without any duplication in the same bucket) of the records which have the following pairs of values:

		Identification No.
Bucket No. 1	$(v_{11}v_{21}, v_{11}v_{31}, v_{21}v_{31})$	(001)
Bucket No. 2	$(v_{12}v_{22}, v_{12}v_{32}, v_{22}v_{32})$	(101)
Bucket No. 3	$(v_{13}v_{23}, v_{13}v_{33}, v_{23}v_{33})$	(201)
Bucket No. 4	$(v_{11}v_{23}, v_{11}v_{32}, v_{23}v_{32})$	(011)
Bucket No. 5	$(v_{12}v_{21}, v_{12}v_{33}, v_{21}v_{33})$	(111)
Bucket No. 6	$(v_{13}v_{22}, v_{13}v_{31}, v_{22}v_{31})$	(211)
Bucket No. 7	$(v_{11}v_{22}, v_{11}v_{33}, v_{22}v_{33})$	(021)
Bucket No. 8	$(v_{12}v_{23}, v_{12}v_{31}, v_{23}v_{31})$	(121)
Bucket No. 9	$(v_{13}v_{21}, v_{13}v_{32}, v_{21}v_{32})$	(221)

The identification number attached to each bucket is the triplet of the coefficients of the equation $[\lambda_0 + \lambda_1x_1 + \lambda_2x_2 = 0 \ \lambda_i\epsilon GF(3)]$ of the line corresponding to the bucket. Within each bucket the accession numbers of the records will be subdivided into subsets, called *subbuckets*, corresponding to each pair of values. In order to avoid duplication of accession numbers in the bucket, the subbuckets will be made non-overlapping by using a chaining technique[6] for common accession numbers. The subbuckets may be identified by concatenating the codes of the pair of values they represent.

Thus the arrangement of the accession numbers will appear as follows:

Bucket Identification Number	*Subbucket Identification Number*	*Accession Number of the Records*
001	0010	
	0020	
	1020	
101	0111	
	0121	
	1121	
201	0212	
	0222	
	1222	
.	.	
.	.	
.	.	
221	0210	
	0221	
	1021	

Thus the accession number of a record which has the values v_{11} and v_{31}, will be stored in the subbucket 0020 within the bucket 001. If this record also has the value v_{21} then its accession number will be entitled to be stored in the subbuckets 0010 and 1020 within the same bucket; but in order to avoid duplication of accession numbers within the same bucket, the accession number of this record will actually be stored in only one of these three subbuckets and the other subbuckets will be chained to it. However, if a record has the values v_{11}, v_{21} and v_{32} then the accession number of this record will be stored in the subbucket 0010 of the bucket 001, and in the subbucket 1021 of the bucket 221, and thus introduce duplication. This can be avoided only by using chaining techniques between buckets but it would increase the search time and hence will not be used.

Suppose a query was posed as "All records which have v_{13} and v_{23} are to be retrieved." Then v_{13} and v_{23} will first be converted into the points of the geometry by a table lookup. These points are 02 and 12. Next the line in EG(2, 3) which contains the points (0, 2) and (1, 2) has to be determined by solving the equation $\lambda_0 + \lambda_1x_1 + \lambda_2x_2 = 0$, in GF(3). On substituting these points in the equation, we get $\lambda_1 = 0$, $\lambda_0 = \lambda_2 \Rightarrow \lambda_0 + \lambda_0x_2 = 0$ or $1 + x_2 = 0$ or $x_2 = -1 = 2$. Thus the bucket corresponding to the line $x_2 = 2$ contains the required records. The identification number of the bucket is 201 and, within this bucket, the subbuckets have to be searched. The records pertaining to the values v_{13} and v_{23} are to be retrieved, hence the identification number of the required subbucket will be 1222 and this search can be done by matching the subbucket identification numbers. The subbucket 1222 will contain the accession numbers of the records which have both v_{13} and v_{23}.

Suppose the query were as follows: "All the records which have v_{11} and v_{12} are to be retrieved." Then the bucket corresponding to the line which contains the points (0, 0) and (0, 1) would be the required bucket. It is easy to see that the line $x_1 = 0$, contains these two points, but there is no bucket corresponding to this line. Thus this BMFS$_2$ will not be able to answer queries when two values of the same attribute are involved.

Remark 6. In order to simplify the scheme, it would be better if an ordering is introduced between the values of the attributes, say $(v_{11}, v_{12}, v_{13}) > (v_{21}, v_{22}, v_{23}) > (v_{31}, v_{32}, v_{33})$, and whenever a query is made on a combination of values then the value with higher rank will occupy the first position, i.e., $v_{13}v_{22}$ will be used instead of $v_{22}v_{13}$, and so on.

• *Retrieval time in BMFS$_2$*

Suppose that

T_1	=	time needed to solve the algebraic equation to determine the bucket
T_2	=	time needed for matching the bucket identification number
T_3	=	time needed for matching the subbucket identification number
T_4	=	time needed for tracing subbucket chaining, if required
ϵ	=	time needed in locating any bucket or subbucket address; ϵ will depend on the specific storage device used. Thus for a random access storage ϵ will be the seek time plus the read time.
τ	=	time needed for matching one machine word with another.

Since the bucket and subbucket identification numbers can be ordered, it is easy to see that the total retrieval T is given by

$$T \leq T_1 + T_4 + \begin{cases} [(N-1)\log_2 s + \log_2 \times \{\phi(N-1, 0, s) - 1\}]\epsilon + \tau \log_2 \binom{s}{2} & \text{for EG}(N, s) \\ [\log_2 \{\phi(N, 1, s) - \phi(N-1, 0, s)\}]\epsilon + \tau \log_2 \binom{s+1}{2} & \text{for PG}(N, s) \end{cases} \quad (4.1)$$

• *Comparison with "inverted" lists scheme*

Under the assumption that the file is uniformly distributed with $M = cs^l$ records, where c is an integer constant, we shall derive the expression for retrieval time using an inverted lists scheme consisting of lists of accession numbers of records on the basis of single attribute values. The number of such lists $= sl$. The number of accession numbers per list is cs^{l-1}. In order to retrieve for a query based on two different attribute values, a table look-up followed by comparison of items on two lists must be performed. Assuming that the table is in the internal storage or core memory, the table look-up time is insignificant if some coding scheme is employed and content addressability is accomplished. For most practical situations, the table may be external. If τ denotes the time needed for comparing two numbers or attribute values by the processor, then the table look up will require at least $2\tau \log_2 (ls)$ units of time, assuming there is ample storage for internal sorting. The time needed to locate the two lists in the external storage will be $2\epsilon \log_2 (ls)$, where ϵ is the time needed for locating a specific address of the external storage. The comparison of the two lists, each containing cs^{l-1} ordered items, will be very time consuming when cs^{l-1} is too large to permit internal sorting. The minimum time required for the comparison of the two lists is $\tau cs^{l-1} \log_2 cs^{l-1}$. However, this would imply the availability of $2cs^{l-1}$ internal storage locations. If sufficient storage is not available more time will be required, since the matching will have to be done on segments of the lists and in stages.

Thus, the total retrieval time, T', satisfies the following inequality

$$\mathrm{T}' \geq 2(\tau + \epsilon) \log_2 (ls) + \tau cs^{l-1} \log_2 cs^{l-1}. \quad (4.1a)$$

On simplification (4.1) will give

$$\left.\begin{aligned} \mathrm{T} &\simeq T_1 + T_4 + (\epsilon + 1)(N-1) \log_2 s \\ &\quad + 2\tau \log s - \tau \text{ for EG} \\ &\simeq T_1 + T_4 + \epsilon 2(N-1) \log \\ &\quad + 2\tau \log s - \tau \text{ for PG} \end{aligned}\right\} . \quad (4.1b)$$

When an EG is used, $l = s^{N-1}$. In this case

$$\begin{aligned} \mathrm{T}' &= 2(\tau + \epsilon)N \log_2 s + \tau cs^{(s^{N-1}-1)} \log \{cs^{(s^{N-1}-1)}\} \\ &\simeq 2(\tau + \epsilon)N \log_2 s + \tau cs^{(s^{N-1}-1)} \log c \\ &\quad + \tau cs^{(s^{N-1}-1)} (s^{N-1} - 1) \log s . \end{aligned} \quad (4.1c)$$

A comparison of (4.1c) with (4.1b) clearly demonstrates the advantage of BMFS$_2$ over the inverted lists scheme.

• *Repetition of records in BMFS$_2$*

When EG(N, s) or PG(N, s) is used to construct BMFS$_2$ then the number of values that any attribute can take is s. If we assume that all these values are equally likely for every attribute, then we get a uniform distribution of records with respect to the values. An exact expression for repetition of records in a file with uniform distribution of records is given below.

The total number of records for an uniformly distributed file will be $M = cs^l$, where c is an integer constant. Suppose each line has m-values; then these m-values belong to m-different attributes. Suppose the different values corresponding to different points on a line are given by

$$A_{i_1} = v_{i_1},\ A_{i_2} = v_{i_2},\ \cdots,\ A_{i_m} = {}_i v_m\ ,$$

where $A_{i_j} j = 1, 2, \cdots, m$ are m-attributes. A record has l values corresponding to the l different attributes. A record will not be stored in the bucket if its $A_{i_j} \neq v_{i_j}$ for all $j = 1, 2, \cdots, m$ or $A_{i_j} \neq v_{i_j}$ for $m - 1$ of j's; $j = 1, 2, \cdots, m$.

Each attribute can take only s values, hence the number records which will not be stored in this bucket is equal to

$$c(s-1)^m s^{l-m} + cm(s-1)^{m-1} s^{l-m}$$
$$= cs^l \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right).$$

Hence the total number of records in the bucket

$$= cs^l - cs^l \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)$$
$$= cs^l \left\{1 - \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)\right\}$$

Thus the total number of records in the filing scheme

$$= M\left\{1 - \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)\right\}$$

$\times$ number of buckets.

$$= \left[\begin{array}{l} M\left\{1 - \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)\right\} \\ \quad \times \{\phi(N, 1, s) - \phi(N-1, 0, s)\} \text{ for } PG(N, s) \\ M\left\{1 - \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)\right\} s^{N-1} \\ \quad \times \{\phi(N-1, 0, s) - 1\} \text{ for } EG(N, s)\ . \end{array}\right.$$

Thus the redundancy factor is

$$\left\{1 - \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)\right\}$$
$$\times \{\phi(N, 1, s) - \phi(N-1, 0, s)\} \text{ for } PG(N, s) \tag{4.3}$$
$$\left\{1 - \left(\frac{s-1}{s}\right)^{m-1} \left(\frac{s+m-1}{s}\right)\right\} s^{N-1}$$
$$\times \{\phi(N-1, 0, s) - 1\} \text{ for } EG(N, s)\ .$$

In the example discussed previously the redundancy factor is $2 + 1/3$.

• *Comparison with "inverted" lists*

For a uniformly distributed file, the redundancy factor for the method of inverted lists is easily seen to be l. In the specific example, this will reduce to a redundancy factor of 3 for inverted lists against a redundancy factor of $2\frac{1}{3}$ for BMFS$_2$. However, depending on the parameters N, m and s, it is possible that BMFS$_2$ may have more redundancy than the inverted lists scheme.

• *Use of two-stage organization method to reduce search time in BMFS$_2$.*

When all the possible values of all the attributes are taken as the points of one finite geometry, the number of buckets can be very large and the search time will also be very large according to (4.1). This can be reduced to a great extent by using first one finite geometry to locate the attributes and then another finite geometry, with as many points as the total number of possible values the specified attributes can take. The buckets and subbuckets corresponding to the finite geometry used to locate the attributes will be called the *first-stage buckets* and *subbuckets*, respectively. The buckets and subbuckets corresponding to the finite geometries used to locate the values will be called *second-stage buckets* and *subbuckets*, respectively. It is obvious that the totality of first-stage buckets and the second-stage bucket (of any one geometry only) will be much less than the total number of buckets that would arise if all the possible values of all attributes were used as points of one geometry. Thus the average search time for the two-stage bucketing organization would be much less than the search time for BMFS$_2$. The first-stage buckets will be constructed using a BFS$_2$ (Ref. 1), where $l = s^N$ or $l = s^N + s^{N-1} + \cdots + s + 1$ according as EG(N, s) or PG(N, s) is used. The subbuckets of this BFS$_2$ will not contain the accession number of any record but will contain the prefix of a location that will contain the second-stage buckets and subbuckets corresponding to this first-stage subbucket. It will also contain the parameters for calculating the suffix of location of the second-stage buckets, i.e., N_{ij} and s_{ij} such that $n_{ij} = n_i + n_j = s_{ij}^{N_{ij}}$ or $n_{ij} = \phi(N_{ij}, 0, s_{ij})$, depending on whether EG(N_{ij}, s_{ij}) or PG(N_{ij}, s_{ij}) is to be used. Thus if the query involves the values v_{ik} and v_{jm} then using these values in the appropriate geometry with parameters N_{ij} and s_{ij}, the suffix of the location of the second-stage bucket can be calculated. Then, concatenating the prefix and suffix part of the second-stage bucket, the exact location of the required second-stage bucket is determined. Within this second-stage bucket the appropriate second-stage subbucket will be determined by matching the identification numbers against the query. The second-stage subbucket will contain the accession numbers of the records which have the appropriate values of the attributes.

It is obvious that the second-stage subbuckets will have provision for queries on combination of values like v_{ik} and

v_{jm}, where $i = j$. If the file organization scheme does not need such provision, then these second-stage subbuckets may be deleted, thus providing a true BMFS$_2$. Sometimes queries of the this type may be of practical importance but if all second-stage subbuckets corresponding to any pair $v_{ik}v_{jm}$ with $i = j$ are retained, then it will introduce too much duplication. This can, however, be avoided by following some predetermined rule. One such rule may be suggested as follows: Suppose a query is made on two values of the i^{th} attribute, then to determine the first-stage bucket and subbucket a dummy attribute, say $(i + 1)^{\text{th}}$ may be introduced. (For the l^{th} attribute the 1^{st} attribute can be introduced as a dummy.) Within the second stage bucket corresponding to the i^{th} and j^{th} attribute retain all the subbuckets corresponding to $v_{ik}v_{jm}$ with $i \neq j$ if $j \neq i + 1$. If $j = i + 1$, then retain all the subbuckets corresponding to $v_{ik}v_{jm}$ where $i \neq j$ and also the second-stage subbuckets corresponding to $v_{ik}v_{im}$ but not the second-stage subbuckets corresponding to $v_{jk}v_{jm}$ except when $j = l$.

- *Remark on the general solution*

The problem of constructing BMFS$_k$ for $k > 2$ using combinatorial algebra or finite geometry has not been yet solved. It appears that more powerful mathematical tools have to be developed before this problem can be solved for any value of k.

5. Numerical example of storage and search on IBM 2311 disk storage using System 360

Assume we have 17 attributes, and each attribute can take 17 different values, and that there are 58,824 records, each of which consists of 17 values belonging to the 17 different attributes. These records are stored on an IBM 2311 disk store with 203 tracks on each disk and 3625 bytes on each track. Each record will have an accession number attached to it, which is 16 bits or 2 bytes long. Assume each value of the attribute takes approximately 4 bytes and a record with its accession number will take about 70 bytes of storage. Thus there will be about 50 records per track and the total number of tracks needed will be about 1154.*

Assume a BMFS$_2$ is constructed with parameters $l = 17$, $s = 17$, $b = 289$ using an EG(2, 17) for storing and retrieving the accession numbers of these 58,824 records. The number of subbuckets within a bucket will be 136.

Assuming that the records are uniformly distributed from formula (4.3), the redundancy factor is obtained as 76. Hence on an average there will be about 4.5 million accession numbers. For simplicity we shall assume that the number of accession numbers per bucket will be the same ($= 4.5 \times 10^6/289 \approx 15{,}570$). It was pointed out in Remark 4, Section 4, that there will be repetition of accession numbers only between buckets but not within buckets. Under the uniform distribution assumption it may be further assumed that the number of records pertaining to any query will be the same, namely, $58{,}824 \div 289 \approx 204$. As pointed out in Example 1, the subbuckets will be further divided into subgroups and the subgroups will be chained. The number of chains needed in any bucket will depend on the data. A subbucket identification number or a chain identification or an accession number can take at most 1 byte, hence the number of subbuckets per track will be $3625/(115 + 2) \approx 31$. There are 136 subbuckets within a bucket; thus a bucket can be stored on 5 tracks, leaving more than 222 bytes for a bucket updating. The total number of buckets is 289 and their storage will need $289 \times 5 = 1445$ tracks.

In a 2311 disk store the seek time ranges between 80 and to 145 milliseconds. We shall take the average seek time to be about 10^{-1} second. The rotation time will be 25 millisecond and the track jump time will be 30 millisecond. The reading of a subbucket will be about 0.5 rotation time $= 12.5$ milliseconds.

In EG(2, 17) the lines of the type $x_1 = c$, where $c \in \text{GF}(17)$ will be taken as the attributes, and the points on any one of these lines will be taken as the permissible values for the attribute corresponding to the line. These lines will be deleted from the geometry. The remaining lines of the deleted geometry can be represented by $ax_1 + x_2 = b$, where $a, b, \in \text{GF}(17)$. Suppose a query includes finding the records which have the values of the attributes corresponding to the points (u_1, v_1) and (u_2, v_2), then $a = -(v_1 - v_2)/(u_1 - u_2)$ and $b = au_1 + v_1$. These calculations have to be performed in the field of integers mod (17). The time needed for such calculations on Model 30 of the IBM/360 System is about 1.8 milliseconds. The time needed to position the reading of disk storage will be 10^{-1} seconds. While the reading head of the disk storage is being set, another table containing the subbucket headings of the particular bucket and their positions on the track will be read into the memory of the computer from a tape unit. This will take about 25 milliseconds, but the processing will be done in parallel with the bucket seeking, and thus it will not add to the retrieval time. Once in core, this table look-up will take only a few microseconds and hence will not enter into our calculations. Positioning the reading head to the beginning of the subbucket and reading the accession numbers will involve a track jump and on an average half-rotation time. This will take $30 + 12.50 = 42.5$ milliseconds.

On an average the records pertaining to any pair of values will be chained to $136/76 \approx 2$ subbuckets. Thus the maximum time needed for reading the accession numbers for a query will involve 1 more setting of the reading head, and 1 more half-rotation (on the average), and will be 112.5 milliseconds. Starting from solution of the equation to reading the accession numbers will take 256.8 milliseconds. The

* If the attribute values are coded, then each attribute value will need only 9 bits, so that a record will be 20 bytes in length. In this case 58,824 records can be stored in 330 tracks. Since the table of codes is small (289 codes), it can be maintained in core storage.

primary file search will involve retrieving the required 204 records, whose accession numbers are given, from the 58,824 records. On an average, retrieving each record will involve one seek time and one reading time, which is equal to 112.5 milliseconds. Hence the time to retrieve the 204 records will be 22,950 milliseconds. Thus the total time needed from start of the query to retrieving the records will take 23.207 seconds. Sometimes it is possible to do the search for the accession numbers and the primary file search in parallel; in that case the total search time reduces to 23.09 seconds. If there are more records per subbucket then, by using either cylinder mode or a surface mode search, saving in search time can be achieved.

Acknowledgment
The authors wish to thank Dr. M. E. Senko for his valuable discussions during the preparation of this paper.

References

1. C. T. Abraham, S. P. Ghosh and D. K. Ray-Chaudhuri, "File Organization Schemes based on Finite Geometries." *IBM Report RC-1459* (1965).
2. R. C. Bose and K. R. Nair, "Partially Balanced Incomplete Block Designs." *Sankhya* **4**, 337–372 (1939).
3. R. C. Bose, "On the Construction of Balanced Incomplete Block Designs." *Annals of Engenics* **9**, 353–399 (1939).
4. R. D. Carmichael, *Introduction to the Theory of Groups of Finite Order*, Ginn and Co., Boston, Mass., 1937.
5. L. R. Johnson, "An Indirect Chaining Method for Addressing on Secondary Keys." *Comm. ACM* **4**, No. 5, 218-222 (1961).
6. W. Buchholz, "File Organization and Addressing," *IBM Systems Journal* **2**, 86–111 (1963).
7. C. T. Abraham, R. C. Bose and S. P. Ghosh, "File Organization of records with unequal valued attributes for multi-attribute queries" *Formatted File Organization Techniques; Final Report, Contract AF 30 (602)-4088, Thomas J. Watson Research Center IBM Corporation*, pp. 107–124 (1967).

Received June 7, 1967