THE DESIGN OF COMBINATORIAL INFORMATION RETRIEVAL SYSTEMS FOR FILES WITH MULTIPLE-VALUED ATTRIBUTES

by

Gary G. Koch

University of North Carolina

Institute of Statistics Mimeo Series No. 552

October 1967

This research was supported by the National Science Foundation Grant No. GP-5790, the Air Force Office of Scientific Research Grant No. AF-AFOSR-68-1415, and the National Institutes of Health, National Institute of General Medical Sciences Grant No. GM-12869.

DEPARTMENT OF STATISTICS

University of North Carolina

Chapel Hill, N. C.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

The author wishes to acknowledge the guidance given by his advisor, Professor R. C. Bose, towards the development of this research. He is particularly grateful for the excellent direction and encouragement received during the early stages of the work as well as for the mathematical and statistical tools taught to him.

He also wishes to thank the other members of his doctoral committee: Professor I. M. Chakravarti, Professor J. E. Grizzle, Professor N. L. Johnson, Professor R. R. Kuebler, and Professor H. L. Lucas for their helpful suggestions and comments. He is particularly indebted to Professor Kuebler for a number of improvements made in Chapter I.

For financial assistance, the author acknowledges a National Defense Education Act Title IV Fellowship and support as a research associate of the Department of Biostatistics. In the latter capactiy, the author is particularly grateful to Professor B. G. Greenberg and Professor J. E. Grizzle for providing him with a means of gaining a greater understanding of the purposes and methods of statistics during the early part of his graduate education.

Finally, he wishes to thank Miss Dorothy Talley for her typing of the manuscript.

ABSTRACT

The recent advent of large scale, high-speed computers has produced an "information revolution." One of the consequences of this has been the need for the development of filing systems which are capable of handling large volumes of data and permitting efficient information retrieval. In this research, first a review is given for a number of different types of filing schemes which have been recently discussed in the literature, with a number of appropriate generalizations being included. Then attention is turned to a general model and filing systems based on certain types of combinatorial configurations. A method of forming one type of configuration is provided through the development of a sequence of theorems indicating how to select a certain subset of m flats from a finite projective geometry which cover all (t-1)-flats, where $m \geq (t-1)$. The construction of another type of configuration is achieved through the development of suitable methods of extending some of the properties of certain small orthogonal arrays and partially balanced arrays to larger schemes. The two types of constructions may be combined to yield multi-stage filing systems which permit efficient retrieval for an appropriate set of queries.

SUMMARY

The development of large-scale, high-speed electronic computers has provided mankind with a means of comprehending large volumes of data. Because of this, a number of questions have recently arisen concerning how one may best file such information in the memory storage area of a computer in order to facilitate its use in the computer system. One criterion for evaluating the efficiency of any filing scheme is the time required for the retrieval of information pertinent to various queries of interest. The purpose of this research is to indicate the application of combinatorial mathematics to some of the problems associated with the construction of various types of efficient computer filing systems.

The basic component of a file is an element called a <u>record</u>. The information contained in a record is expressed in terms of data fields which represent various levels of attributes which are associated with the record. Moreover, each record is uniquely identified by an <u>accession number</u>. In the case of a computerized filing system, the accession number of each record is stored in a unique element of one or more disjoint subsets (called <u>buckets</u>) of the computer memory. The construction of these buckets determines the difficulty with which information, as expressed in terms of queries, can be retrieved. The queries which are of interest here are those which can be expressed in terms of a set of given levels of some particular subset of attributes.

Until recently, the best known type of filing scheme has been the first-order inverted filing system. It is formed by letting one bucket correspond to each level of each attribute. A record is then stored in each of the buckets associated with the levels of attributes which it possesses. Such systems allow efficient retrieval of queries specified in terms of only one attribute; indeed, one simply retrieves all records in the bucket corresponding to the level of that particular attribute specified in the query. However, to retrieve a query involving two attributes, one must first extract all records in each of the two corresponding buckets, and then find the records common to the two groups by matching the accession numbers. This matching process can require a large amount of computer time. Moreover, the time increases as the size of the file increases since there are then more records to be examined in the matching. With queries involving more than two attributes, this retrieval problem becomes progressively more serious.

Because of the previously cited disadvantages of the first-order inverted filing system, a need arose for the development of schemes permitting more efficient information retrieval. In recent years, research directed at the application of combinatorial mathematics to the design of filing schemes was started at the IBM Thomas J. Watson Research Center. Abraham, Ghosh, and Ray-Chaudhuni [1][1] used the theory of finite geometries to form systems which allow efficient retrieval of certain types of queries involving pairs of binary attributes. With these, attributes

[1] A number in square brackets refers to the bibliography listed at the end.

corresponded to points and buckets to lines. The retrieval of a query involving a pair of attributes may be achieved by identifying the bucket corresponding to the unique line through the appropriate points. Abraham and Ghosh [28] used deleted finite geometries to construct similar types of filing schemes permitting efficient retrieval of queries involving multiple-valued attributes. In section (1.4) of this thesis, Theorems (1.4.1), (1.4.2), and (1.4.6) are given as some straightforward generalizations of the results that have been obtained from this finite geometry approach.

Chapter II is concerned with a general mathematical model for filing systems which was motivated by Ray-Chaudhuri [43] . One type of scheme suggested by this model is based on the construction of certain types of combinatorial configurations. When two-fold queries are of interest, these configurations may be formed from certain balanced incomplete block and group divisible designs. More generally, in the case of t-fold queries, the selection of a certain subset of m-flats from a finite projective geometry which cover all (t-1) flats, where $m \geq (t-1)$, may be used. One of the primary results of this research is the development of a general system of such covers for $t = 2,3,4$. These are constructed in Theorems (2.4.2)-(2.4.8).

In Chapter III, another method of forming combinatorial configurations is presented. It is based on developing algorithms for extending the covering properties of certain orthogonal arrays and partially balanced arrays associated with a small number of attributes to arrays involving larger numbers of attributes. The suggested methods have the desirable property that the number of subsets required for the coverings increases at a noticeably slower rate than the number of attributes when the number of attributes is sufficiently large.

The filing schemes based on the combinatorial configurations of Chapter III suffer from the disadvantage that each bucket pertains to a very large number of queries. In order to make the relationship between query and bucket more specific, multi-stage filing systems similar to those formulated by Ray-Chaudhuri [43] may be used. These are discussed in Chapter IV. One of the consequences arising from the use of multi-stage systems is that the retrieval time for any query becomes linearly related to the number of buckets at each stage.

Finally, the bibliography has been extended to include a large number of references pertaining to the different topics associated with the design of the types of filing systems discussed. This seemed appropriate since such a comprehensive set of references does not exist elsewhere in the literature.

CHAPTER I

INTRODUCTION

1.1. The importance of efficient filing systems.

With the advent of large-scale computer systems, the years since World War II may in one sense be referred to as the "Information Revolution." In many ways, its effect on the culture of the world may be as dramatic as that of the "Industrial Revolution" of the 18th and 19th centuries. The lives of many individuals have already been influenced through its production of changes in the employment needs of businesses and services, its treatment of the bookkeeping details of many types of financial transactions, its capabilities as a means of solution to many computationally difficult problems in scientific research. Today, more different types of information are being obtained from more individuals by more firms, survey groups, utilities, and governmental agencies than ever before. Such data are felt to be of some importance to the interested groups, and computer technology provides methods for comprehending and using their content.

As a result of the existence and availability of vast stores of information, the question naturally arises as to how one may best file such information in order to facilitate its use in a computer system. Filing represents a method of preserving information. The success of any filing scheme can be measured in terms of the ease with which it is possible to retrieve the information pertinent to a given query or task. In the case of large-scale, high-speed electronic computers, the memory storage area can be used to hold well-organized files which can be designed so that information regarding any particular query of interest can be retrieved very rapidly. This fact provides the basis for the formulation of the problems of designing computer filing systems for efficient information retrieval. The actual construction of such files poses a variety of questions, some of which can be attacked by methods of combinatorial mathematics. The description of several different ways of achieving efficient systems represents the purpose of the present research.

To illustrate the situation, let us first consider as an example an information storage system which is called a tumor registry. The ultimate aim of such a scheme is to provide for a large number of hospitals and clinics a centralized mechanism which will allow medical researchers to readily obtain data pertaining to the medical histories of individuals having various types of cancer. Such histories would reflect background information like socio-economic status and previous medical experience(s), the basic characteristics of the tumor(s) involved, the medical treatment and outcome; also included would be information from various follow-up studies. Because many of the variables pertaining to one type of cancer do not necessarily pertain to other types (for example, smoking may be considered relevant to lung cancer but not to breast cancer) and because different hospitals may have different ways of collecting and recording similar data, the construction of a central data system appears at first to be an overwhelming task involving all variables relevant to all types of cancer and accounting for their definitions by different medical groups.

The problem may be simplified to some extent by adopting a variable-oriented point of view. By this we mean that the design of the file will be hinged to some extent on the types of queries, as expressed in terms of levels of variables, for which the system will be expected to provide information. For example, a study of the effect of a certain type of chemotherapy on middle-aged non-smokers with cancer of the larynx may be of interest. To carry out such research, we need to locate all individuals in the file who are relevant to this investigation. Suppose that all individuals have been uniquely identified by combinations of numbers reflecting the hospitals treating them and their own patient numbers within the respective hospitals. Let a system of cells be constructed in such a way that to each cell there corresponds a combination of levels of variables. In such cells, one then stores the patient identification numbers of all individuals whose histories satisfy the definition of the cell. If such a file has been constructed in a systematic way, then the cells pertaining to the different types of queries of interest can be located efficiently and quickly. Once this is done, the identification numbers stored there can be printed and the corresponding individuals then located. When this is achieved, their records can then be obtained from the files of the respective hospitals, and the relevant data extracted and analyzed.

In the example posed earlier, the variables and levels in the query are

	Variable	Level
1.	treatment	chemotherapy of given type
2.	disease type	cancer of the larynx
3.	age	middle-aged
4.	does individual smoke?	no

To conduct the study, one has the filing system locate (by a computer operation) the cell relevant to the query and print out the identification numbers of the individuals there, after which he proceeds as outlined above. The point of this example is to reveal how a variable-oriented filing system can expedite the retrieval of data pertinent to research problems which can be expressed as queries involving variable levels. As a result of such efficient retrieval, the use of the different types of variable-oriented filing schemes to be discussed in what follows has a potentially great value to the design of centralized data systems like tumor registries.

In the next section we shall assign precise definitions to some of the concepts arising in a technical discussion of filing systems. Then we shall consider some of the well-known filing systems currently in existence, as well as some others which have been proposed from research conducted at the IBM Thomas J. Watson Research Center. In addition, the concepts of retrieval time and redundancy will be introduced as two criteria for evaluating the efficiency of a system.

1.2. A technical characterization of a filing system.

Most of the terminology here arises from this author's interpretation of the papers of Buchholz [20], Abraham, Ghosh, and Ray-Chaudhuri [1], and Ray-Chaudhuri [43]. The basic component of a file is an assembly of information which is called a record and which uniquely corresponds to a particular individual or item of interest. Each record has two basic parts. The first is an identification sequence like a serial number, patient number, Social Security number, etc., which is uniquely associated with the record or the subject giving rise to it. This number is sometimes called the primary key since it represents the primary identifier. The second part of the record consists of a number of data fields which correspond in a one-to-one fashion to a number of attributes or information variables. These are sometimes called secondary keys. In this research, we will assume that, for any individual, each of the attributes can take exactly one of finitely many different values. The different values which an attribute may have will be called levels. Hence, what appears in the data fields of a record is precisely the appropriate combination of levels of attributes associated with the individual. Thus, in the example of the preceding section, if the number of patients in the registry is 900, and the numbers of levels of the four variables are 15, 40, 7, 2, respectively, a record of interest might be

	Patient number	Variable 1	2	3	4
Verbal record:	324	Chemo-therapy	Cancer of the larynx	Middle-aged	No
Decimal record:	324	10	23	5	0

Once the records of a collection of individuals have been obtained, they can be stored in some permanent memory. This may take the form of a block of filing cabinets, a card catalogue, or tape. The location of a record in the permanent memory is called its accession number. One aspect of the basic problem of file organization is the definition of the correspondence between accession numbers and primary keys. This process, which is called key transformation, has been discussed by a number of researchers in computer systems. For additional details and bibliography, the reader is referred to Buckholz [20]. Here we shall assume that the accession numbers have already been assigned (for example, one may take them, in some instances, to coincide exactly with the primary key, or one may be able to assign them serially).

The computerized aspect of a filing system arises from the storage of the accession number of a record in several different addresses of the fast-memory of a high-speed electronic computer; e.g., magnetic disks. Usually, this fast memory may be conceived of as partitioned into a number of disjoint subsets called buckets. The construction of these buckets determines the difficulty with which information, as expressed in terms of queries, is retrieved from the filing system. Here, a query will be taken to mean a set of given levels for some particular subset of attributes. The principal criterion for evaluating the efficiency of a computerized filing system is the retrieval times required to determine the accession numbers appropriate to different types of queries. One way of decreasing retrieval time for any given query is to provide a rule which associates with the query a bucket containing the relevant accession numbers. However, for such a filing scheme to be efficient with respect to a wide class of queries, some accession numbers will be stored in more than one place. This redundancy is the price paid for efficient retrieval and need not cause worry as long as the totality of addresses is large enough to embrace the system. However, if the fast-memory is not particularly large, redundancy becomes a problem that must be adequately handled in the construction of the filing system. Apart from the restrictions implied by this, the value of any filing system will ultimately be evaluated by the retrieval time required to locate the records pertinent to the members of a class of queries.

Finally, we should indicate that the concepts of retrieval time and redundancy are not as precise measures of efficiency as they appear to be. Unfortunately, the definitions of both terms can be considered vague when viewed from a theoretical point of view. This results from the dependence of these quantities more on the properties of the particular computer involved than on the filing system. The implications of these remarks to the problem of comparing different filing systems will be seen later.

1.3. The inverted filing systems.

At present, one of the most widely used filing systems is known as the simple or first order inverted filing system. The structure of such filing schemes is characterized by a correspondence between levels of attributes and buckets; i.e., if A_{ij} represents the j-th level of the i-th attribute, where $i = 1,2, \ldots, v$ and $j = 1, 2, \ldots, n_i$, then a bucket M_{ij} is associated with each A_{ij}, giving $N_1 = \sum_{i=1}^{v} n_i$ buckets in all. The buckets $\{M_{ij}\}$ represent disjoint sets of addresses in the fast memory of a computer. The accession number of a record is stored at one of the addresses contained in M_{ij} provided that the individual involved possesses the j-th level of the i-th attribute. In doing this, we necessarily must assume that each of the sets M_{ij} contains sufficiently many addresses as to allow the storage of the accession numbers for all the records of individuals having A_{ij}. Since, in many instances, accession numbers will be of a somewhat small dimension to store (e.g., they will seldom involve more than ten decimal digits) this assumption is not too unrealistic for most large-scale computers.

The first order inverted filing system is very efficient in retrieving queries which are specified in terms of one level of one attribute. For example, to retrieve all records with A_{11}, the computer first determines M_{11} as the bucket corresponding to A_{11} and then proceeds to print the accession numbers located there. These accession numbers may then be used to extract sequentially from the slow permanent memory each of the records with A_{11}. The determination of the appropriate bucket M_{ij} for the query A_{ij} is achieved by letting a bucket identification number correspond to each bucket. For example, the bucket identification number associated with M_{ij} may be taken as $w_{ij} = \sum_{\alpha=0}^{i-1} n_\alpha + j$, where $n_0 = 0$; thus, the w's are $1,2,\ldots, N_1$. Similarly, w_{ij} may be used as a query identification number corresponding to the query A_{ij}. Hence, to determine the appropriate bucket for a given query, all that is involved is first the computation of the query identification number w_{ij} and then a comparison of w_{ij} with all the bucket identification numbers in the natural serial order until a match occurs. The addresses of the corresponding bucket (which contains the relevant accession numbers) are linked to the location of the bucket identification number by a process called chaining. Loosely speaking, this means that once a positive decision has been reached at the address of the bucket identification number, the computer is instructed to proceed to the chained address which, for example, may be the first element of the bucket. It then proceeds as indicated before.

The dominant component of the retrieval time associated with a single attribute query is the time required to match the bucket and query identification numbers. If τ_b represents the time required for each comparison, then for the query A_{ij} with query identification number w_{ij}, the matching time is approximately $w_{ij}\tau_b$. If all the single attribute queries A_{ij} are equally likely, the average retrieval time is $T_{1,a} = \frac{1}{N_1} \sum_{w=1}^{N_1} w\tau_b = (N_1 + 1)\tau_b /2$. Alternatively, in some instances, a <u>binary search technique</u> can be applied to this matching problem instead of the serial comparison discussed above. By this we mean that the query identification number is first compared with the middle bucket identification number (say $[(N_1 + 1) / 2]_-$, where $[u]_-$ denotes the greatest integer not exceeding u) to determine whether it is larger or smaller. If it is larger, then a comparison is made with the 3/4-point; otherwise, with the 1/4-point. This successive halving of relevant sub-intervals is continued until the desired match occurs. An upper bound for the time required by the binary search is approximately $T_{1,m} = [\log_2 N_1]_+ \ \tau_b$ where $[u]_+$ denotes the smallest integer greater than u. Although $T_{1,m}$ is of smaller order than $T_{1,a}$, the binary search technique may not always be feasible. Hence, both of these retrieval time functions are used to express the efficiency of the system.

Finally, one should note that the redundancy of the simple inverted filing system is $R_1 = v$. This follows from the fact that the accession number of a record is stored in exactly one of the n_i buckets M_{ij} associated with the i-th attribute, since the corresponding individual must possess exactly one of those n_i levels. Since there are v attributes in all, each accession number appears in v addresses.

From what has been said previously, the first order inverted filing system appears to be a reasonably satisfactory scheme. Unfortunately, serious complications arise for it when the information retrieval problem involves multiple attribute queries. In particular, to retrieve a query involving two attributes, the system must first extract all records in each of the two corresponding buckets and then find the records common to the two sets by matching the accession numbers. For example, to retrieve the query $\{A_{11},A_{21}\}$ first the buckets M_{11} and M_{21} must be identified and then the set of accession numbers belonging to $\tilde{M}_{11} \cap \tilde{M}_{21}$ must be determined, where $\tilde{M}_{ij}$ denotes the set of acession numbers stored at M_{ij}. It is this latter part of the retrieval procedure that can require a large amount of computer time, for it requires that each accession number in $\tilde{M}_{11}$ be compared with each of those in $\tilde{M}_{21}$ until a decision can be reached as to whether it belongs to $\tilde{M}_{11} \cap \tilde{M}_{21}$. If the accession numbers in each of the buckets are serially ordered, then the time required to match the two lists determined from a two-fold query can be reduced by applying the binary search technique mentioned earlier. In particular, if m_{11} individuals had A_{11} and m_{21} had A_{21}, then at most $[\log_2 m_{21}]_+$ comparisons need be made to determine whether a given accession number in $\tilde{M}_{11}$ has a match in $\tilde{M}_{21}$ (the non-existence of a match is determined when no match has occurred and no further cuts are possible). Thus, the upper bound on the total time required for the matching is $m_{11}[\log_2 m_{21}]_+ \tau_a$ where τ_a is the time required for each comparison. Adding to this the time required to identify M_{11} and M_{21}, we find an upper bound for the retrieval time to be $2[\log_2 N_1]_+\tau_b + m_{11}[\log_2 m_{21}]_+\tau_a$. The important point to note here is that the time required for this matching is an increasing function of the size of the file, because as more and more records are added to the file, the numbers of individuals m_{ij} with A_{ij} for $i = 1, 2, \ldots, v$ and $j = 1, 2, \ldots, n_i$ all increase. Hence, quantities like $m_{11}[\log_2 m_{21}]_+$ all increase. This fact represents the most striking disadvantage of the inverted filing system for retrieving records pertinent to two-fold queries. With queries involving more than two attributes, this problem becomes progressively more and more serious. For example, to retrieve the query $\{A_{11}, A_{21}, \ldots, A_{t1}\}$, where $t \leq v$, the set $\tilde{M}_{11} \cap \tilde{M}_{21} \cap \ldots \cap \tilde{M}_{t1}$ must be determined by successive matching. As a result, if the future demands upon a filing system will involve the frequent retrieval of multiple attribute queries, then a need arises to consider schemes which are more appropriate for handling these than the first order inverted filing system.

A direct generalization of the simple inverted filing system is the second order inverted filing system, the construction of which is oriented at the retrieval of queries involving two attributes. In this scheme, a bucket $M_{ij;i'j'}$ is made to correspond with a distinct pair of levels of different attributes A_{ij} and $A_{i'j'}$, with $i, i' = 1,2, \ldots, v$ and $i' > i$; $j = 1,2,\ldots,n_i$; $j' = 1,2,\ldots,n_{i'}$. This gives a total of $N_2 = \sum_{i=1}^{v} \sum_{i'>i}^{v} n_i n_{i'}$ buckets in all. An accession number is stored in $M_{ij;i'j'}$ provided that the individual involved possesses the j-th level of the i-th attribute and the j'-th level of the i'-th attribute. As in the case of the single-attribute buckets, the $\{M_{ij;i'j'}\}$ are assumed to be sufficiently large disjoint subsets of the fast memory as to contain the accession numbers of all records of individuals having both A_{ij} and $A_{i'j'}$.

By definition, the second order inverted filing system enables two-fold queries to be retrieved efficiently. In the example of the query $\{A_{11}, A_{21}\}$ discussed earlier, all that is involved is the identification of the bucket $M_{11;21}$. As before, query identification numbers and bucket identification numbers can be assigned. For example, the identification for $\{A_{ij}, A_{i'j'}\}$ may be taken as the ordered pair $(w_{ij}, w_{i'j'})$

where $w_{ij} = \sum_{\alpha=0}^{i-1} n_\alpha + j$, $w_{i'j'} = \sum_{\alpha=0}^{i'-1} n_\alpha + j'$, $n_o = 0$. After the bucket has been $_m$ located by matching its identification number with that of the query, chaining is then used to proceed to the relevant addresses. By arguments similar to those used before, the average retrieval time is $T_{2,a} = (N_2 + 1)\tau /2$, assuming the possible two-fold queries are equally likely. If the $(w_{ij}, w_{i'j'})$ are viewed as ordered numbers and the binary search technique is applied, then an upper bound for the search is essentially $T_{2,m} = [\log_2 N_2]_+ \tau_b$. Both $T_{2,a}$ and $T_{2,m}$ are independent of the size of the file. Hence, for files in which the number of records is large when compared to N_1, the second order inverted filing system is more efficient at retrieving two-fold queries than the simple inverted filing system.

The second order inverted filing system can also be used to handle single attribute queries. This is accomplished by associating with the single attribute a series of two-fold queries, the components of which are the given attribute level and all levels of some other attribute. For example, to retrieve $\{A_{11}\}$, the procedure can be to retrieve $\{A_{11}, A_{21}\}$, $\{A_{11}, A_{22}\}$, ... $\{A_{11}, A_{2n_2}\}$, where the fact that the buckets $M_{11;21}$, $M_{11;22}$, ..., $M_{11;2n_2}$ are consecutive can be exploited by the use of a chaining option on the single attribute queries. By an extension of this argument, the following correspondence can be constructed.

Single Attribute Query Involves Levels of Attribute	Paired Are All Levels of Attribute
1	2
2	3

...	...
v-1	v
v	1

To handle queries involving the v-th attribute, of course, the pairs $\{A_{vj}, A_{1j'}\}$ would have to be reversed, because of the condition $i < i'$ in the bucket identifications. Although this procedure appears somewhat complex, efficient chaining reduces the search to the location of the bucket corresponding to the first element of the sequence of pairs. Hence, the quantities $T_{2,a}$ and $T_{2,m}$ approximately express the magnitude of retrieval time in this system for both uni-fold and two-fold queries. Since $N_2 > N_1$, the second order inverted filing system is not as efficient for single attribute queries as is the simple inverted filing system. This essentially represents one of the prices paid for the much increased efficiency with respect to two-fold queries.

Another disadvantage of the second order inverted filing system is the much increased redundancy associated with it. Since each individual record must possess exactly one of the possible values of each pair of attributes, its accession number will appear in $\binom{v}{2} = \frac{v(v-1)}{2}$ different buckets. Hence the redundancy of this scheme is $R_2 = \frac{v(v-1)}{2}$, which exceeds R_1 for $v > 3$.

If one is willing to increase the redundancy somewhat further, uni-fold queries may be handled more efficiently by supplementing the second order inverted filing system with the N_1 buckets of the simple inverted filing system. The number of buckets for the combined system is $N_c = N_1 + N_2$. One then sets up a structure of identification numbers as before. This scheme may be readily used to retrieve uni-fold queries essentially as efficiently as the simple inverted filing system and two-fold queries essentially as efficiently as the second order inverted filing system. The price of this additional efficiency is the increase in redundancy, which now has become $R_c = v + \frac{v(v-1)}{2}$.

As one would naturally suspect, once interest arises in three-fold and higher-order queries, the second order inverted filing system is no longer efficient since retrieval involves matching of accession numbers; and hence the time required for it depends directly on the size of the file. In particular, to retrieve $\{A_{11}, A_{21}, A_{31}\}$, the buckets $M_{11;21}$ and $M_{11;31}$ must be identified and then the set of accession numbers belonging to $\tilde{M}_{11;21} \cap \tilde{M}_{11;31}$ determined. The consequences of this are essentially the same as were observed in the case of two-fold queries with the simple inverted filing system. The problem of course becomes progressively more serious as the order of the query to be retrieved increases.

One could consider the concepts of third and higher order inverted filing systems. However, when this is done, the redundancy can become intractably large. Indeed, for the t-th order inverted filing system, $R_t = \binom{v}{t}$, which increases rapidly for increasing t up to v/2. Also the number of buckets, $N_t = \sum_{i_1 < i_2 < \ldots < i_t} n_{i_1} n_{i_2} \ldots n_{i_t}$, becomes quite large, causing the matching of query and bucket identification numbers to require much more time. These problems suggested the need for other types of filing systems which are efficient for the retrieval of multiple-attribute queries. In the remaining sections we shall consider some of the constructions possible by using various methods of combinatorial mathematics.

1.4. Filing schemes based on finite geometries.

One of the first attempts to apply the methods of combinatorial mathematics to the construction of efficient filing systems involved the use of the structure of finite geometries. In this section we shall consider the results obtained from this approach by Abraham, Ghosh, and Ray-Chaudhuri [1], and Ghosh and Abraham [28].

First, let us briefly summarize the properties of the two types of finite geometries: the finite projective geometry, denoted by PG(N,q), and the finite Euclidean geometry, denoted by EG(N,q), where q is an integer power of some prime integer p. A more complete discussion is given in, for example, Carmichael [22] or Bose [3].

1.4.1. The finite projective geometry PG(N,q).

The points of PG(N,q) are represented by (N+1)-tuples $\underset{\sim}{x}' = (x_0, x_1, \ldots, x_N)$, where $x_0, x_1, \ldots, x_N$ belong to the Galois field GF(q), a finite system of q elements on which are defined two arithmetic operations (addition and multiplication) that satisfy the same basic axioms characteristic of the rational numbers. In addition, the vectors $\underset{\sim}{x}'$ and $\rho\underset{\sim}{x}' = (\rho x_0, \rho x_1, \ldots, \rho x_N)$, where ρ is any non-zero element of GF(q), are regarded as the same point, and (0, 0, ..., 0) is not regarded as a point. Hence, there are $(q^{N+1} - 1)/(q - 1)$ points in PG(N,q).

An m-dimensional flat space, called an m-flat, is defined to be the set of points satisfying the (N - m) linearly independent homogeneous equations $\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{0}$, where $\underset{\sim}{A}$ is a full-rank $(N - m) \times (N + 1)$ matrix of elements from GF(q). Alternatively, all points whose corresponding row vectors lie in the vector space generated by the rows of $\underset{\sim}{A}$ constitute an (N-m-1)-flat, which is called the dual of the m-flat. In this sense, a point is referred to as a zero-flat; a line, a one-flat; a plane, a two-flat; etc. The number of points belonging to any (N-m-1)-flat is $(q^{N-m} - 1)/(q - 1)$ where the fact that two possible non-null combinations of the rows of $\underset{\sim}{A}$ are not allowed to be proportional is accounted for by the division. Let $\Phi(N, N-m-1, q)$ denote the number of distinct (N-m-1)-flats in PG(N,q). Then $\Phi(N, N-m-1, q)$ equals the number of ways of choosing (N-m) independent points in PG(N, q) divided by the number of ways of choosing (N - m) independent points in PG(N - m - 1, q); i.e.,

$$\Phi(N,N-m-1,q) = \frac{\left(\frac{q^{N+1}-1}{q-1}\right)\left(\frac{q^{N+1}-1}{q-1}-1\right)\left(\frac{q^{N+1}-1}{q-1}-\frac{q^{2}-1}{q-1}\right)\cdots\left(\frac{q^{N+1}-1}{q-1}-\frac{q^{N-m-1}-1}{q-1}\right)}{\left(\frac{q^{N-m}-1}{q-1}\right)\left(\frac{q^{N-m}-1}{q-1}-1\right)\left(\frac{q^{N-m}-1}{q-1}-\frac{q^{2}-1}{q-1}\right)\cdots\left(\frac{q^{N-m}-1}{q-1}-\frac{q^{N-m-1}-1}{q-1}\right)}$$

$$= \frac{(q^{N+1}-1)(q^{N}-1)\cdots(q^{m+2}-1)}{(q^{N-m}-1)(q^{N-m-1}-1)\cdots(q-1)} \tag{1.4.1}$$

With the above framework in mind, we may note that the function Φ satisfies the following relations:

$$\begin{aligned} \Phi(N, m, q) &= \Phi(N, N-m-1, q) \\ \Phi(N, -1, q) &\equiv 1, \end{aligned} \tag{1.4.2}$$

The first equality results from the duality of m-flats and (N-m-1)-flats; the second represents a definition so that the first is valid for any integer m such that $0 \leq m \leq N$.

1.4.2. The finite Euclidean geometry EG(N, q).

The points of EG(N, q) are represented by N-tuples $\underset{\sim}{x}' = (x_1, \ldots, x_N)$, where $x_1, x_2, \ldots, x_N$ belong to GF(q). Each of the possible N-tuples corresponds to a distinct point, (0, 0, ..., 0) included. Hence, there are q^N points in all.

An m-dimensional flat space is defined to be the set of points satisfying the (N - m) linearly independent non-homogeneous equations $\underset{\sim}{A}_1 \underset{\sim}{x} = \underset{\sim}{a}_0$, where $\underset{\sim}{A}_1$ is an (N - m) x N matrix and $\underset{\sim}{a}_0$ is an (N - m) x 1 vector with elements from GF(q). On the other hand, a dual (N - m)-flat may be obtained as the set of points whose corresponding row vectors lie in the vector space generated by the rows of $\underset{\sim}{A}_1$. Thus, the number of points in such an (N - m)-flat is q^{N-m}.

The Euclidean geometry EG(N, q) may be extracted from the projective geometry PG(N, q) by deleting the so-called <u>(N - 1)-flat at infinity</u> $x_0 = 0$ and all points and flats contained in it. Hence, the number of m-flats in EG(N, q) equals the number of m-flats in PG(N, q) less the number of m-flats contained in the (N - 1)-flat $x_0 = 0$; i.e., $\Phi(N, m, q) - \Phi(N - 1, m, q) = q^{N-m}\, \Phi(N - 1, m - 1, q)$.

The various m-flats can be partitioned into parallel bundles by allowing the associated vectors $\underset{\sim}{a}_0$ to assume all possible values. In this way, there are q^{N-m} m-flats in each such parallel bundle and $\Phi(N - 1, m - 1, q)$ distinct parallel bundles in all. Finally, each point in EG(N, q) lies in exactly one of the m-flats belonging to any parallel bundle.

1.4.3. <u>Balanced multiple filing schemes.</u>

Let us assume that there are $v = p^n$ attributes, each of which can take $s = p^m$ values, where p is a prime integer. Suppose further that there is interest in a filing system capable of efficient retrieval of queries involving pairs of levels of different attributes. This problem may be attacked by using finite geometries.

Let u, c, and N be any integers such that $uc = m$ and $uN = (m + n)$. In particular, we may always take $u = 1$, $c = m$, and $N = (m + n)$. Consider a parallel bundle of c-flats in the Euclidean geometry EG(N,q) where $q = p^u$.

Let each c-flat in the bundle be identified with a unique attribute, and let each point on any given one of these c-flats be identified with a unique level of the associated attribute This correspondence is a well defined one since each point in EG(N,q) belongs to exactly one of the c-flats in the parallel bundle. In addition because there are $q^{N-c} = p^n = v$ c-flats in the bundle and since each c-flat contains $q^c = p^m = s$ points, all levels of all attributes have been accounted for. The buckets of a filing system may be identified in a one-to-one way with the set of all lines in the geometry except those lying within any one of the $v = q^{N-c}$ c-flats in the given parallel bundle. Hence, the number of buckets b is given by

$$\begin{aligned} b &= q^{N-1}\,\Phi(N-1,\ 0,\ q) - q^{N-c}\,\{q^{c-1}\,\Phi\,(c-1,\ 0,\ q)\} \qquad (1.4.3)\\ &= q^{N-1}\,\{\Phi(N-1,\ 0,\ q) - \Phi(c-1,\ 0,\ q)\}\\ &= q^{N-1}\,\{\frac{q^N-1}{q-1} - \frac{q^c-1}{q-1}\}\\ &= q^{N-1}\,(q^N - q^c)/(q-1)\\ &= q^{N+c-1}(q^{N-c}-1)/(q-1)\\ &= v(v-1)s^2/q(q-1) \end{aligned}$$

Since through any two points there passes exactly one line, it follows that to each two-fold query there corresponds exactly one bucket. Moreover the fact that q points lie on any line means that any given bucket pertains to q(q-1)/2 different queries. As a result, one may note that all $v(v-1)s^2/2$ possible two-fold queries are accounted for by verifying the relation

$$bq(q-1)/2 = v(v-1)s^2/2. \qquad (1.4.4)$$

Hence, we have the following theorem.

Theorem (1.4.1). There exists a filing system oriented toward two-fold queries for the case $v = p^n$ and $n_1 = n_2 \dots = n_v = s = p^m$, where p is a prime integer. It is based on $b = v(v-1)\, s^2/q(q-1)$ buckets, each of which pertains to pairs formed from q levels of different attributes; here $q = p^u$, u being an integer which is a common divisor of m and n.

In the actual filing system described above, the accession number for a record of an individual is stored in a given bucket if he possesses any two of the levels of attributes to which the bucket pertains. The actual filing is further refined by partitioning each bucket into a number of sub-buckets so that to each of the two-fold queries associated with a bucket, there corresponds a sub-bucket. The sub-buckets may be ordered by using the implied ordering on pairs of attributes that may be derived from an ordering of the attributes. For example, if a bucket pertains to A_{21}, A_{53}, A_{72}, A_{84}, the ordering for the sub-buckets corresponds to

$$\{A_{21}, A_{53}\},\ \{A_{21}, A_{72}\},\ \{A_{21}, A_{84}\},\ \{A_{53}, A_{72}\},\ \{A_{53}, A_{84}\},\ \{A_{72}, A_{84}\}.$$

The actual sub-bucket which will contain the accession number of a record is the first one in the ordering for which the individual has the associated pair. In this way, any given record is stored at most once in any bucket.

To retrieve the query $\{A_{ij}, A_{i'j'}\}$, first the appropriate bucket is identified by determining the unique line through the points corresponding to A_{ij} and $A_{i'j'}$. After this is done, the sub-bucket is located by matching a query identification number with a sub-bucket identification number in a fashion similar to that indicated in the preceding sub-section. All records associated with this sub-bucket are then retrieved. However, not all records satisfying this query are accounted for by this sub-bucket. In fact, for each $A_{i''j''}$ associated with the bucket and such that $i'' < i$, individuals having $A_{i''j''}$, A_{ij}, and $A_{i'j'}$ are stored in the sub-bucket corresponding to $\{A_{i''j''}, A_{ij}\}$. Such records are retrieved by having their respective sets of addresses chained to the sub-bucket associated with $\{A_{ij}, A_{i'j'}\}$. Similarly, chaining must also be made to sub-buckets corresponding to $\{A_{i''j''}, A_{i'''j'''}\}$ where $i'' < i''' < i$. Some of the details involved here will be illustrated later in an example.

Filing systems like the one described above have been called <u>second order balanced multiple filing schemes</u> by Ghosh and Abraham [28]. They considered the case associated with Theorem (1.4.1) when $c = 1$, $u = m$, and (n/u) is an integer.

Another type of balanced multiple filing system may be based on the projective geometry in which the Euclidean geometry previously considered is embedded. Consider the projective analogue of the parallel bundle of c-flats together with a parallel bundle of c-flats lying in the (N-1)-flat at infinity such that all the c-flats involved intersect in the same (c-1)-flat contained in the (N-1)-flat at infinity. By duality, the number of distinct c-flats passing through a common (c-1)-flat is the same as the number of (N-c-1)-flats lying in an (N-c)-flat and hence equals Φ (N-c, N-c-1, q) = Φ (N-c, 0, q). To each of these c-flats, let there correspond a unique attribute. The number of points lying in any given one of these c-flats but not in the common (c-1)-flat is $\Phi(c, o, q) - \Phi(c-1, o, q) = q^c$. To each of these points, let there correspond a unique level of the associated attribute. If the lines in PG(N, q), other than those in the previously specified c-flats are taken to represent the buckets of a filing system, then by an argument similar to that given for theorem (1.4.1), we have

Theorem (1.4.2). There exists a second order balanced multiple filing system for the case $v = \Phi(N-c, 0, q)$ and $n_1 = n_2 = \dots = n_v = s$, where $s = q^c = p^m$ and where $q = p^u$, p being a prime integer. It is based on $b = \frac{v(v-1)s^2}{q(q+1)}$ buckets, each of which pertains to pairs formed from $(q+1)$ levels of different attributes. Moreover, to any two-fold query there corresponds exactly one bucket.

One may verify the expression for b from

$$b = \Phi(N, 1, q) - \Phi(N-c, 0, q)\{\Phi(c, 1, q) - \Phi(c-1, 1, q)\} - \Phi(c-1, 1, q)$$

$$= \frac{(q^{N+1}-1)(q^N-1)}{(q^2-1)(q-1)} - \frac{(q^{N-c+1}-1)}{(q-1)}\frac{q^{c-1}(q^c-1)}{(q-1)} - \frac{(q^c-1)(q^{c-1}-1)}{(q^2-1)\quad(q-1)}$$

$$= \frac{(q^{2N+1}-q^{N+1}-q^N+1) - (q^c-1)\{(q+1)(q^N-q^{c-1}) + (q^{c-1}-1)\}}{(q^2-1)\quad(q-1)}$$

$$= \frac{(q^{2N+1}-q^{N+1}-q^N+1) - (q^c-1)(q^{N+1}-q^c+q^N-1)}{(q^2-1)\quad(q-1)}$$

$$= \frac{(q^{2N+1}-q^{N+1}-q^N+1) - (q^{N+c+1}-q^{2c}+q^{N+c}-q^c-q^{N+1}+q^c-q^N+1)}{(q^2-1)\quad(q-1)}$$

$$= \frac{(q^{2N+1}-q^{N+c+1}-q^{N+c}+q^{2c})}{(q^2-1)\quad(q-1)}$$

$$= \frac{q^{2c}(q^{2N-2c+1}-q^{N-c+1}-q^{N-c}+1)}{(q^2-1)\,(q-1)}$$

$$= \frac{q^{2c}(q^{N-c+1}-1)\,(q^{N-c}-1)}{(q^2-1)\,(q-1)}$$

$$= \frac{q^{N-c+1}-1}{q-1}\quad\frac{q(q^{N-c}-1)}{q-1}\quad\frac{q^{2c}}{q(q+1)}$$

$$= v(v-1)s^2/q(q+1)$$

For the case $c = 1$, the result given in Theorem (1.4.2) coincides with that obtained by Ghosh and Abraham [28].

Records are stored in the buckets and sub-buckets of the filing system associated with Theorem (1.4.2) according to rules similar to those described for Theorem (1.4.1.). The retrieval procedure for any two-fold query involves solving a set of equations to identify the bucket, matching to determine the sub-bucket, and chaining as indicated before. Let us now look at an example of the mechanical aspects of the filing schemes obtainable from Theorems (1.4.1) and (1.4.2).

Example (1.4.1). Suppose there are $v = 7$ attributes, each of which assumes $s = 2^2 = 4$ levels. Let $q = 2$. Then $4 = s = q^c$ gives $c = 2$, and $7 = v = \phi(N-c, 0, q) = \phi(N-2, 0, 2)$ gives $N = 4$. Consider the line at infinity in PG(4,2) defined by the equations

$$x_0 = 0,\ x_1 = 0,\ x_2 = 0 .$$

The seven planes through this line and their corresponding attributes are

$$\begin{aligned}
A_1&:\ x_0 = 0,\ x_1 = 0\\
A_2&:\ x_0 = 0,\ x_2 = 0\\
A_3&:\ x_0 = 0,\ x_1 + x_2 = 0\\
A_4&:\ x_1 = 0,\ x_2 = 0\\
A_5&:\ x_1 = 0,\ x_0 + x_2 = 0\\
A_6&:\ x_2 = 0,\ x_0 + x_1 = 0\\
A_7&:\ x_0 + x_2 = 0,\ x_1 + x_2 = 0
\end{aligned}$$

The points associated with the levels A_{ij} of the attributes are as follows:

A_{11}:	00100	A_{21}:	01000	A_{31}:	01100	A_{41}:	10000
A_{12}:	00101	A_{22}:	01001	A_{32}:	01101	A_{42}:	10001
A_{13}:	00110	A_{23}:	01010	A_{33}:	01110	A_{43}:	10010
A_{14}:	00111	A_{24}:	01011	A_{34}:	01111	A_{44}:	10011

A_{51}:	10100	A_{61}:	11000	A_{71}:	11100
A_{52}:	10101	A_{62}:	11001	A_{72}:	11101
A_{53}:	10110	A_{63}:	11010	A_{73}:	11110
A_{54}:	10111	A_{64}:	11011	A_{74}:	11111

The buckets of the filing scheme correspond to the lines in the geometry which do not lie entirely in any one of the seven planes associated with $A_1, \ldots, A_7$. Bucket identification numbers can be formed by sequencing the row vectors of the matrix of coefficients associated with the defining equations. However, the equations corresponding to a line not always unique. On the other hand, they can be reduced to a unique row-echelon form in which

i. the first non-zero coefficient on the left hand side of each equation is unity.

ii. if the first non-zero coefficient in the α-th equation is x_{u_α}, then $u_1 < u_2 < \ldots u_\alpha < \ldots$

iii. the coefficient of x_{u_α} is zero in every equation except the α-th

It is this form of the matrix of coefficients which will be used to assign the bucket identification numbers. For example, the line defined by $x_0 = 0$, $x_3 = 0$, $x_4 = 0$ is in row-echelon form and may be identified by the number 100000001000001. This line passes through the points A_{11}:00100, A_{21}:01000, A_{31}:01100. Sub-bucket identification numbers may be assigned by sequencing the points corresponding to the pairs of attribute levels associated with the sub-buckets. For example, the sub-bucket pertaining to $\{A_{11}, A_{21}\}$ can be denoted by 0010001000. The sub-buckets in any given bucket can be ordered on the basis of their identification numbers. For the bucket under consideration, this is

$\{A_{11}, A_{21}\}$	0010001000
$\{A_{11}, A_{31}\}$	0010001100
$\{A_{21}, A_{31}\}$	0100001100

The storage procedure for the accession number of a record having any two of A_{11}, A_{21}, A_{31} is

Part (i) of Sub-bucket 0010001000 if A_{11}, A_{21}, but not A_{31};
Part (ii) of Sub-bucket 0010001000 if A_{11}, A_{21}, and A_{31};
Sub-bucket 0010001100 if A_{11}, A_{31}, but not A_{21};
Sub-bucket 0100001100 if A_{21}, A_{31}, but not A_{11}.

The sub-buckets 0010001100 and 0100001100 are chained to Part (ii) of sub-bucket 0010001000 because the records there satisfy all three of the different possible two-fold queries associated with the bucket. What has just been indicated can be used to formulate the storage procedures for the other buckets.

Let us now consider what is involved in the retrieval of a query, say $\{A_{33}, A_{74}\}$. First, we need to determine the unique line through (01110) and (11111). This is done by solving the equations

$$a_1 + a_2 + a_3 = 0,$$
$$a_0 + a_1 + a_2 + a_3 + a_4 = 0 .$$

A solution is

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} a_0 + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} a_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} a_2 ,$$

and hence the line is given by $x_0 + x_4 = 0$, $x_1 + x_3 = 0$, $x_2 + x_3 = 0$ which is in row-echelon form and has the identification number 100010101000110. This number is then compared against an ordering of bucket identification numbers until a match occurs. In this way, the relevant bucket is located. The points lying on the above line are A_{33}: 01110, A_{42}: 10001, and A_{74}: 11111. The sub-bucket 0111011111 is located by matching and all accession numbers from it are extracted. Then the relevant part of the sub-bucket 0111010001 is reached by chaining and the retrieval procedure is completed.

1.4.4. Retrieval time for balanced multiple filing schemes.

Ghosh and Abraham [28] cite four basic components for the retrieval time in filing schemes based on finite geometries. These are

T_1 = time needed to solve the algebraic equations to determine the bucket identification number.

T_2 = time needed for matching the bucket identification number.

T_3 = time needed for matching the sub-bucket identification number.

T_4 = time needed for tracing sub-bucket chaining when necessary.

Let τ_G be the time needed to compare two identification numbers. The quantities T_1, T_4, and τ_G are assumed to be parameters of the particular system involved. If the bucket and sub-bucket **identification** numbers have been ordered as previously indicated, then the average retrieval time is given by

$$T_{G,a} = T_1 + T_4 + \left(\frac{b+1}{2}\right) \tau_G + \begin{cases} \left[\left(\frac{q}{2}\right) + 1 \right]\tau_G/2 & \text{for } EG(N,q) \text{ system} \\ \left[\left(\frac{q+1}{2}\right) + 1 \right]\tau_G/2 & \text{for } PG(N,q) \text{ system} \end{cases}$$

where b has the value appropriate to the system. The binary search technique is not overly useful here because the upper bound associated with it for $T_2 + T_3$ is τ_G [$\log_2 \{v(v-1)s^2/2\}$ $]_+ = \tau_G$ [$\log_2(N_2)$ $]_+$ which is the same quantity given for the second order inverted filing system. However, if the matching time for determining the bucket is some quantity $\epsilon_G < \tau_G$, then the binary search technique is practical. In this case, the following upper bound is of interest.

$$T_{G,m} \leq T_1 + T_4 + \epsilon_G[\log_2 b]_+ + \begin{cases} \tau_G[\log_2(\frac{q}{2})]_+ & \text{for } EG(N,q) \text{ system} \\ \tau_G[\log_2(\frac{q+1}{2})]_+ & \text{for } PG(N,q) \text{ system} \end{cases}$$

where b has the value appropriate to the system. Note that both $T_{G,a}$ and $T_{G,m}$ do not depend on the number of records in the file..

1.4.5. Redundancy in balanced multiple filing schemes.

Suppose that there are M records, where M is an integer multiple of s^v, and that each of the possible s^v records occurs equally often; i.e., we assume a uniform distribution of records. Let the redundancy of the balanced multiple filing scheme be defined as the average number of times each record appears in the file. In this sense, an exact expression will now be derived for the redundancy. The basic approach used represents a slight extension of that of Ghosh and Abraham [28] who obtained approximate results.

Let the attribute levels corresponding to the different points on a line be denoted $(a_1, a_2, \ldots, a_r)$ where the subscript ordering is derived from the original ordering of the attributes. Let (a_i, a_j) denote the number of records stored in the sub-bucket corresponding to the a_i, a_j combination of attributes. Let $[a_i, a_j]$ denote the number of records having a_i, a_j. Then we have the following results

$$(a_1,a_2) = [a_1,a_2] = M/s^2$$

$$(a_1,a_3) = [a_1,a_3] - [a_1,a_2,a_3] = (M/s^2) - (M/s^3)$$

$$(a_1,a_4) = [a_1,a_4] - [a_1,a_2,a_4] - [a_1,a_3,a_4] + [a_1,a_2,a_3,a_4]$$

$$= (M/s^2)-2(M/s^3) + (M/s^4)$$

$$\cdots$$

$$(a_1,a_r) = [a_1,a_r]-[a_1,a_2,a_r] -\cdots- [a_1,a_{r-1},a_r] + [a_1,a_2,a_3,a_r]$$

$$+ \ldots + (-1)^r[a_1,a_2,\ldots a_r]$$

$$= M(s-1)^{r-2}/s^r$$

$$(a_2,a_3) = [a_2,a_3] - [a_1,a_2,a_3] = M(s-1)/s^3$$

$$(a_2,a_4) = [a_2,a_4] - [a_1,a_2,a_4] - [a_2,a_3,a_4] + [a_1,a_2,a_3,a_4]$$

$$= M(s-1)^2/s^4$$

$$\cdots$$

$$(a_2,a_r) = [a_2,a_r] - [a_1,a_2,a_r] - \ldots - [a_2,a_{r-1},a_r] + [a_1,a_2,a_3,a_r]$$

$$+ \ldots + (-1)^r [a_1,a_2,\ldots a_r]$$

$$= M(s-1)^{r-2}/s^r$$

$$(a_3,a_4) = [a_3,a_4] - [a_1,a_3,a_4] - [a_2,a_3,a_4] + [a_1,a_2,a_3,a_4]$$

$$= M(s-1)^2/s^4$$

$$\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots$$

$$(a_{r-1},a_r) = [a_{r-1},a_r] - [a_1,a_{r-1},a_r] - \ldots - [a_{r-2},a_{r-1},a_r] +$$

$$[a_1,a_2,a_{r-1},a_r] + \ldots + (-1)^r[a_1,a_2,\ldots,a_r]$$

$$= M(s-1)^{r-2}/s^r$$

Thus, the total number of records associated with any bucket is given by

$$C = \{M/s^2\}\{(r-1)(1-\tfrac{1}{s})^{r-2} + (r-2)(1-\tfrac{1}{s})^{r-3} + \ldots + 3(1-\tfrac{1}{s})^2 + 2(1-\tfrac{1}{s}) + 1\}$$

$$= \{M/s^2\} \sum_{j=0}^{r-2} (j+1)(1-\tfrac{1}{s})^j$$

$$= M \frac{d}{ds} \{ \sum_{j=0}^{r-2} \left(1-\frac{1}{s}\right)^{j+1} \}$$

$$= M \frac{d}{ds} [\frac{s-1}{s} \{1 - \left(\frac{s-1}{s}\right)^{r-1}\}/\{1 - \frac{s-1}{s}\}]$$

$$= M \frac{d}{ds} [(s-1) \{1 - \left(\frac{s-1}{s}\right)^{r-1}\}]$$

$$= M \{ 1 - (r+s-1)(s-1)^{r-1}/s^r\}$$

The redundancy $R_G = Cb/M$, where b is the number of buckets, is given by

$$R_G = \{1 - \left(\frac{r+s-1}{s}\right)\left(1 - \frac{1}{s}\right)^{r-1}\} b.$$

For the scheme of Theorem (1.4.1), we have

$$R_{EG} = \{1 - \left(\frac{q+s-1}{s}\right)\left(1 - \frac{1}{s}\right)^{q-1}\}\{\frac{v(v-1)s^2}{q\ (q-1)}\},$$

while for the scheme of Theorem (1.4.2), we have

$$R_{PG} = \{1-\left(\frac{q+s}{s}\right)\left(1 - \frac{1}{s}\right)^{q}\}\{\frac{v(v-1)s^2}{q(q+1)}\},$$

In particular, for Example (1.4.1), we have

$$R_{PG} = \{1 - \left(\frac{2+4}{4}\right)\left(1 - \frac{1}{4}\right)^2\} \left(\frac{7(6)\ (16)}{6}\right) = 17.5.$$

Since $17.5 < \binom{7}{2} = 21$, the above system is less redundant than the second order inverted filing system appropriate to the example.

1.4.6 Balanced filing schemes.

Before the development of the balanced multiple filing schemes which were discussed in the preceding sub-sections, Abraham, Ghosh, and Ray-Chaudhuri [1] considered a situation in which only one level of any given attribute was of interest with respect to retrieval. In some sense, attributes may be viewed as having two levels here; namely "presence" of the relevant level and "absence" of it. However, retrieval only pertains to the concept of "presence". When this is done, queries may be specified by simply listing the combination of attributes involved. Here, we shall let $A_1, A_2, \ldots, A_v$

denote the particular attributes (levels). Filing schemes which permit efficient retrieval of queries involving pairs of these attributes may be constructed by using finite geometries. Let the points of a finite geometry (either some $PG(N, q)$ or some $EG(N,q)$) correspond to the attributes in a one-one fashion. The buckets of the filing system are uniquely identified with the lines of the geometry. Since only one line passes through any pair of distinct points, exactly one bucket corresponds to any query involving two attributes. The following theorems apply.

Theorem (1.4.3). Given that retrieval pertains to only one level of $v = \Phi(N, 0, q)$ attributes, there exists a filing system based on $PG(N,q)$ which is oriented toward two-fold queries. It consists of $b = \Phi(N, 1, q)$ buckets, each of which is relevant to pairs formed from $(q+1)$ different attributes. Moreover, to any two-fold query there corresponds exactly one bucket.

Theorem (1.4.4). Given that retrieval pertains to only one level of $v = q^N$ attributes, there exists a filing system based on $EG(N,q)$ which is oriented toward two-fold queries. It consists of $b = q^{N-1}\Phi(N-1, 0, q)$ buckets, each of which is relevant to pairs formed from q different attributes. Moreover, to any two-fold query there corresponds exactly one bucket.

The above theorems are given in Abraham, Ghosh, and Ray-Chaudhuri [1] They called the filing system based on them balanced filing schemes.

The mechanics of the balanced filing schemes are basically the same as those of the balanced multiple filing schemes. The buckets are divided into sub-buckets which are ordered. Similar storage and retrieval procedures are employed, with chaining being used when necessary. As a result, the expression for average retrieval time is given by $T_{G,a}$ while an appropriate upper bound is given by $T_{G,m}$. Finally, by means of an argument similar to that given in sub-section 1.4.5, the redundancy is

$$R_{G,2} = \{1 - (r + 1)\ (1/2)^r\}b$$

where a uniform distribution of records is assumed in the sense that each of the 2^v possible records occurs equally often.

1.4.7. Some other filing schemes based on finite geometries.

The balanced filing scheme described in the preceding sub-section is directed at retrieval of two-fold queries. If there is interest in queries involving three attributes (again, with each having only one pertinent level), matching of accession numbers as described in the case of the second order inverted filing system will have to be performed. Alternatively, one may attempt to develop third order balanced filing systems. Unfortunately, such schemes are quite difficult to form. However, Abraham, Ghosh, and Ray-Chaudhuri [1] suggest the following simple construction based on the geometry EG(N,2).

Let each point of EG(N,2) correspond to an attribute. The buckets will be identified with the planes of the geometry. Since each line in this geometry contains only two points, no three points are collinear, and hence any three points determine a unique plane. Thus, we have the following theorem.

Theorem 1.4.5. Given that retrieval pertains to only one level of $v = 2^N$ attributes, there exists a filing system based on EG(N,2) which is oriented toward three-fold queries. It consists of

$b' = 2^{N-2}\,\Phi\,(N-1, 1, 2)$ buckets, each of which is relevant to triples formed from four different attributes. Moreover, to any three-fold query there corresponds exactly one bucket.

The redundancy of the above scheme is

$$R_{G,3} = b\,\{1/8 + 3(1/8 - 1/16)\} = (5/16)b.$$

Comparing $R_{G,3}$ with the redundancy of the third order inverted filing system R_3, we have

$$(R_3/R_{G,3}) = \frac{2^N(2^N-1)\,(2^N-2)/6}{5(2^N-1)\,(2^{N-1}-1)2^{N-2}/48}$$

$$= (8/6)\,(48/5) = 64/5$$

In other words, the third order inverted filing system contains nearly 13 times as much redundancy as the third order balanced filing scheme.

Another type of third order balanced filing scheme may be based on the structure of a homogeneous, non-degenerate quadric in PG(3, q) where $q > 2$. The properties of these surfaces are discussed in Bose [5], Primrose [35], and Ray-Chaudhuri [41]. To construct the filing system, let there correspond to each attribute a point belonging to the quadric surface $a_1x_1^2 + a_{12}x_1x_2 + a_2x_2^2 = x_0x_3$ where $\varphi\,(x_1, x_2) = a_1x_1^2 + a_{12}x_1x_2 + a_2x_2^2$ is an irreducible quadratic form with coefficients belonging to GF(q). Such a quadric contains $v = (q^2 + 1)$ points no three of which are collinear. The buckets will be identified with the planes which pass through at least three of the points on the quadric. Since any plane either intersects the quadric in a conic section with (q+1) points or in a single point, the number of buckets is given by $\binom{q^2+1}{3}/\binom{q+1}{3} = q(q^2 + 1) = \Phi(3, 2, q) - q^2 - 1$.

Hence, we have the theorem.

Theorem (1.4.6). Given that retrieval pertains to only one level of $v = q^2 + 1$ attributes, where $q = p^u > 2$, there exists a filing system based on a quadric in PG(3, q) which is oriented toward three-fold queries. It involves $b = q(q^2 + 1)$ buckets, each of which is relevant to triples formed from (q+1) different attributes. Moreover, to any three-fold query there corresponds exactly one bucket.

Example (1.4.2). Suppose v = 10. Let the attributes correspond to the points of the quadric $x_0^2 + x_1^2 = x_2 x_3$ in PG(3, 3) as follows

A_1:	0010	A_6:	1022
A_2:	0001	A_7:	1121
A_3:	0111	A_8:	1112
A_4:	0122	A_9:	1212
A_5:	1011	A_{10}:	1221

Hence, the buckets and the attributes associated with them are

$x_0 = 0$	A_1, A_2, A_3, A_4	$x_0 + x_1 + 2x_2 = 0$	: A_2, A_3, A_5, A_7
$x_1 = 0$	A_1, A_2, A_5, A_6	$x_0 + 2x_1 + x_2 = 0$	: A_2, A_3, A_6, A_9
$x_0 + 2x_1 = 0$	A_1, A_2, A_7, A_8	$x_0 + 2x_1 + 2x_2 = 0$	: A_2, A_4, A_5, A_{10}
$x_0 + x_1 = 0$	A_1, A_2, A_9, A_{10}	$x_0 + x_1 + x_3 = 0$	: A_1, A_4, A_6, A_7
$x_0 + 2x_2 = 0$	A_2, A_5, A_8, A_9	$x_0 + x_1 + 2x_3 = 0$	: A_1, A_3, A_5, A_8
$x_0 + x_2 = 0$	A_2, A_6, A_7, A_{10}	$x_0 + 2x_1 + x_3 = 0$	: A_1, A_3, A_6, A_{10}
$x_0 + 2x_3 = 0$	A_1, A_5, A_7, A_{10}	$x_0 + 2x_1 + 2x_3 = 0$	: A_1, A_4, A_5, A_9
$x_0 + x_3 = 0$	A_1, A_6, A_8, A_9	$x_0 + x_2 + 2x_3 = 0$	: A_3, A_4, A_8, A_9
$x_1 + 2x_2 = 0$	A_2, A_3, A_8, A_{10}	$x_0 + 2x_2 + x_3 = 0$	: A_3, A_4, A_7, A_{10}
$x_1 + x_2 = 0$	A_2, A_4, A_7, A_9	$x_1 + x_2 + 2x_3 = 0$	: A_5, A_6, A_8, A_{10}
$x_1 + 2x_3 = 0$	A_1, A_3, A_7, A_9	$x_1 + 2x_2 + x_3 = 0$	: A_5, A_6, A_7, A_9
$x_1 + x_3 = 0$	A_1, A_4, A_8, A_{10}	$x_0 + x_1 + x_2 + x_3 = 0$	: A_3, A_5, A_9, A_{10}

$x_2 + 2x_3 = 0$: A_3, A_4, A_5, A_6 $x_0 + x_1 + 2x_2 + 2x_3 = 0$: A_4, A_6, A_9, A_{10}

$x_2 + x_3 = 0$: A_7, A_8, A_9, A_{10} $x_0 + 2x_1 + x_2 + x_3 = 0$: A_4, A_5, A_7, A_8

$x_0 + x_1 + x_2 = 0$: A_2, A_4, A_6, A_8 $x_0 + 2x_1 + 2x_2 + 2x_3 = 0$: A_3, A_6, A_7, A_8

The coefficients of x_0, x_1, x_2, x_3 may be used to form four-digit bucket identification numbers. Similarly, each bucket may be divided into four sub-buckets which can be labelled by sequencing the coordinates of the points involved. The actual storage and retrieval procedures are essentially the same as those indicated previously.

By an argument similar to that given for the second order balanced filing system, one may verify that the redundancy for the system associated with Theorem (1.4.6) is

$$R_{G,3} = \frac{d^2}{dx^2} \left\{ \frac{x^2 - x^{q+1}}{1 - x} \middle| \; x = 1/2 \right\} (1/2)^4 b$$

$$= b\{1 - [1 + (q+1) + \binom{q+1}{2}](1/2)^{q+1}\}$$

This quantity is substantially less than R_3 as can be seen by considering their ratio.

The principal disadvantage of the second order and third order balanced filing systems is that they do not handle lower order queries efficiently. As was exhibited in the case of first order queries in the second order inverted filing system, some modifications can be introduced so that a lower order query is transformed into a number of the appropriate higher order queries which are then subsequently retrieved. However, this is more complicated to do for the filing schemes considered in this section. A better approach would be to increase the redundancy somewhat by supplementing the higher order filing scheme with the relevant lower order schemes. For example, to handle three-fold and lower order queries, one could combine a third order balanced filing scheme with a second order balanced filing scheme and a first order inverted filing scheme. The query type would then direct the system (to the appropriate component it should refer to) in order to perform retrieval.

Alternatively, the above situation may be approached by using another type of filing system which is oriented toward retrieval of more general type queries. Abraham, Ghosh, and Ray-Chaudhuri [1] introduced the concept of generalized balanced filing scheme in the following theorem.

Theorem (1.4.7). Given that retrieval pertains to only one level of v attributes, there exists a filing system which is oriented toward queries involving any t or fewer attributes. The buckets are identified with the 0-flats, 1-flats ..., and (t-1)-flats of a finite projective geometry. As a result,

$$v = \Phi(N, 0, q),\quad b = \sum_{\alpha=0}^{\min(t-1,N)} \Phi(N, \alpha, q) \quad \text{if } PG(N,q) \text{ is used,}$$

$$v = q^N,\quad b = \sum_{\alpha=0}^{\min(t-1,N)} q^{N-\alpha}\, \Phi(N-1,\alpha-1, q) \text{ if } EG(N, q) \text{ is used.}$$

The accession number of a record is stored in a bucket corresponding to a m-flat if the individual has at least (m+1) attributes such that the associated points all lie in the m-flat and form a basis of it. A series of sub-buckets and chaining is used to determine the exact location. To retrieve a u-fold query, the m-flat of minimum dimension which contains all the u points associated with the query is determined. This identifies the bucket. The relevant sub-buckets are then located and retrieval is completed. A complete description of the above type of filing system is given in the previously cited reference.

Although the generalized balanced filing scheme is capable of efficiently handling a general class of queries in the sense that retrieval time does not depend on the size of file, the system does have some disadvantages. In particular, the redundancy is quite high because each record will be stored in a large number of buckets. Secondly, the system is quite complex and may be difficult to implement. Another type of general filing system which avoids some of these problems will be discussed in the next chapter.

1.4.8. Some further remarks.

In the preceding sub-sections, some filing schemes have been constructed for particular values of v and, as in sub-section 1.4.3, the number of levels of the attributes s. For certain other situations as, for example, a case where v does not assume one of these particular values or where the n_i are different from each other, an appropriate system can be constructed by using a geometry in which the desired properties of the scheme can be embedded. In particular, for the case of the balanced filing scheme, we can use the appropriate geometry with the smallest number of points provided the number of points is at least v. For additional details concerning this, one is referred to Abraham, Ghosh, and Ray-Chaudhuri [1] and Ghosh and Abraham [28].

CHAPTER II

A GENERAL MATHEMATICAL MODEL AND SOME RELATED FILING SYSTEMS

2.1. A mathematical model for filing systems.

In this section, a mathematical model for filing systems will be formulated. The approach used is similar to that of Ray-Chaudhuri [43] for the case in which retrieval pertains to only one level of each of v attributes.

As in the previous chapter, let A_{ij} denote the j-th level of the i-th attribute where $i = 1, 2, \ldots, v$; $j = 1, 2, \ldots, n_i$. A file F is denoted by the triple $F = (\mathfrak{J}, \Omega, f)$ where

a. $\mathfrak{J}$ represents the population of individuals.

b. Ω represents the set $\{A_{11}, \ldots, A_{1n_1}, \ldots, A_{v1}, \ldots, A_{vn_v}\}$ of attribute levels.

c. f is a function from $\mathfrak{J}$ to subsets of Ω such that f(I) denotes the set of attribute levels possessed by individual I. Since each individual has exactly one level of each attribute, it is clear that $|f(I) \cap A_{i0}| = 1$ for each i where $A_{i0} = \bigcup_{j=1}^{n_i} A_{ij}$ and $| C |$ is the number of elements in the set C.

The storage procedure S for the filing scheme is characterized by the triple $S = (\mathfrak{J}, M, \sigma)$ where

a. $\mathfrak{J}$ represents the population of individuals.

b. M represents a set of positive integers corresponding to the set of possible addresses.

c. σ is a 1-1 function from $\mathfrak{J}$ to disjoint subsets of M; the subset $\sigma(I)$ contains the addresses where the accession number of I's record is stored.

The third and most important aspect of the filing scheme is the retrieval procedure R. This may be identified with the triple $R = (\mathcal{G}, M, r)$ where

a. **Let $\mathcal{G}$ represent a class of subsets from Ω such that each A in $\mathcal{G}$** contains at most one element from each group A_{io} (since any individual can possess only one level of any attribute); **then we may take $\mathcal{G}$ to represent the class of queries.**

b. M represents the set of addresses available for storage.

c. r is a function from $\mathcal{G}$ to subsets of M with the subset r (A) being such that if f(I) contains A, then $|\sigma(I) \cap r(A)| = 1$; in other words, only one of the addresses, where the accession number of I's record is stored, is related to the retrieval of the query A.

The filing system is said to be of order t if for each A belonging to $\mathcal{G}$, the relation $|A| \leq t$ holds.

To illustrate the applicability of this model, let us consider a system called the <u>extended inverted filing system</u> by Ray-Chaudhuri. In this scheme, to each subset A in $\mathcal{G}$, there corresponds a subset M_A of M such that $M_A \cap M_{A'}$ is empty where A' is any other subset in $\mathcal{G}$. The accession number of I's record is stored in M_A if f(I) contains A. Hence, the set $\sigma(I)$ contains an element corresponding to each of the subsets A in $\mathcal{G}$ which is contained in f(I). The size of this set indicates the redundancy associated with the storage of I's accession number. The retrieval rule for the query A is simply $r(A) = M_A$. The retrieval time essentially reduces to the time required to locate the bucket M_A by matching identification numbers in a fashion similar to that indicated in Chapter I. As a result, it is directly related to the number of subsets in $\mathcal{G}$. Finally, when $\mathcal{G}$ is such that $|A| \leq t$, the extended inverted filing system becomes identical with the t-th order inverted filing system considered earlier.

2.2. Combinatorial configurations and combinatorial filing systems.

A combinatorial configuration $(\Omega, k, \mathcal{G}, b)$ consists of a master set Ω (the set of attribute levels), a class of subsets $\mathcal{G}$ (the queries), and blocks $B_1, B_2, \ldots, B_b$ (which are certain subsets of Ω) such that

i. $|B_h| \leq k$

ii. for every A in $\mathcal{G}$, there exists an h such that $A \subseteq B_h$.

If $|A| \leq t$ for each A in $\mathcal{G}$, then the configuration is said to be of order t and is denoted as an (Ω, k, t, b) scheme. The actual construction of (Ω, k, t, b) configurations with minimum b is a very difficult problem in combinatorial mathematics. For the case of $t = 2$ and $n_1 = n_2 = \ldots = n_v = s$, such arrays are equivalent to certain group divisible (GD) designs used in statistical research. Some of the possible solutions obtainable here will be indicated later. However, in most situations, such optimal schemes are largely unknown and perhaps can be found only through systematic trial and error. As a result, in the later sections of this research, we shall be mostly concerned with the development of schemes which are easy to construct and seem practical in the sense that b is not excessively large.

A combinatorial filing system may be based on a combinatorial configuration as follows. Let the blocks $B_1, B_2, \ldots, B_b$ be arranged in serial order. For each A in $\mathcal{G}$, define $\gamma(A) = h$ if A is contained in B_h but is not contained in $B_{h'}$ for $h' < h$. Hence B_h is the first block which contains A. Let $\mathfrak{a}_h$ denote the collection of all subsets A of Ω such that $\gamma(A) = h$. To each combination of A and h, let there correspond sufficiently large disjoint subsets $M_{h,A}$ of M. The accession number of the I-th individual's record is stored in an element of $M_{h,A}$ if and only if the largest set which f(I) has in common with B_h is the subset A in $\mathfrak{a}_h$; i.e., if $f(I) \cap B_h = A$. Let

$$M_h = \bigcup_{A \in \mathfrak{a}_h} M_{h,A} \tag{2.1.1}$$

The sets M_h may be called the buckets of the filing system while the subsets $M_{h,A}$ may be called the sub-buckets.

The retrieval procedure for any query simply involves the determination of the appropriate bucket by identifying the first block which contains the subset specified in the query. Afterwards, all sub-buckets corresponding to subsets which contain the query set are located and the accession numbers therein obtained. Thus, the retrieval function may be formally written

$$r(A) = \bigcup_{A \subseteq C \in \mathfrak{a}} M_{h,C} \tag{2.1.2}$$

where $A \in \mathfrak{a}$ and $\gamma(A) = h$. Hence, from the preceding remarks, one can see that once a combinatorial configuration (which is efficient in the sense of b not being too large) has been constructed, a reasonable filing scheme may be readily based on it. In particular, Bose, Abraham, and Ghosh [9] have used a procedure similar to this. Finally, the concepts of combinatorial configuration and combinatorial filing scheme as developed here are equivalent to the ones considered by Ray-Chaudhuri [43] for the situation in which only one level of any attribute was of interest with respect to retrieval.

<u>Example (2.1.1).</u> Suppose there are $v = 5$ attributes, each of which assumes $s = 3$ levels; i.e; $n_1 = n_2 = n_3 = n_4 = n_5 = 3$. Hence, the set Ω is given by $\Omega = \{A_{11}, A_{12}, A_{13}; A_{21}, A_{22}, A_{23}; A_{31}, A_{32}, A_{33}; A_{41}, A_{42}, A_{43}; A_{51}, A_{52}, A_{53}\}$. A second order combinatorial configuration $(\Omega, 4, 2, 15)$ is provided by the blocks

$$\begin{array}{lll}
B_1 = \{A_{21},A_{31},A_{41},A_{51}\} & B_2 = \{A_{22},A_{32},A_{42},A_{52}\} & B_3 = \{A_{23},A_{33},A_{43},A_{53}\} \\
B_4 = \{A_{11},A_{31},A_{52},A_{43}\} & B_5 = \{A_{12},A_{41},A_{32},A_{53}\} & B_6 = \{A_{13},A_{51},A_{42},A_{33}\} \\
B_7 = \{A_{11},A_{21},A_{42},A_{53}\} & B_8 = \{A_{12},A_{51},A_{22},A_{43}\} & B_9 = \{A_{13},A_{41},A_{52},A_{23}\} \\
B_{10} = \{A_{11},A_{51},A_{32},A_{23}\} & B_{11} = \{A_{12},A_{21},A_{52},A_{33}\} & B_{12} = \{A_{13},A_{31},A_{23},A_{53}\} \\
B_{13} = \{A_{11},A_{41},A_{22},A_{33}\} & B_{14} = \{A_{12},A_{31},A_{42},A_{23}\} & B_{15} = \{A_{13},A_{21},A_{32},A_{43}\}
\end{array}$$

The buckets of the combinatorial filing system are sets of addresses which correspond to the blocks B_h while the sub-buckets therein correspond to the possible subsets. If the subsets are represented by writing all the attribute levels associated with the block in a four-tuple and then placing a bar over the ones to be excluded from the subset, then we have

$$\begin{array}{llll}
M_1 = \{ & (A_{21},\bar{A}_{31},\bar{A}_{41},\bar{A}_{51}), & (\bar{A}_{21},A_{31},\bar{A}_{41},\bar{A}_{51}), & (\bar{A}_{21},\bar{A}_{31},A_{41},\bar{A}_{51}), \\
& (\bar{A}_{21},\bar{A}_{31},\bar{A}_{41},A_{51}), & (A_{21},A_{31},\bar{A}_{41},\bar{A}_{51}), & (A_{21},\bar{A}_{31},A_{41},\bar{A}_{51}), \\
& (A_{21},\bar{A}_{31},\bar{A}_{41},A_{51}), & (\bar{A}_{21},A_{31},A_{41},\bar{A}_{51}), & (\bar{A}_{21},A_{31},\bar{A}_{41},A_{51}), \\
& (\bar{A}_{21},\bar{A}_{31},A_{41},A_{51}), & (A_{21},A_{31},A_{41},\bar{A}_{51}), & (A_{21},A_{31},\bar{A}_{41},A_{51}), \\
& (A_{21},\bar{A}_{31},A_{41},A_{51}), & (\bar{A}_{21},A_{31},A_{41},A_{51}), & (A_{21},A_{31},A_{41},A_{51})\}. \\
M_2 = \{ & (A_{22},\bar{A}_{32},\bar{A}_{42},\bar{A}_{52}), & (\bar{A}_{22},A_{32},\bar{A}_{42},\bar{A}_{52}), & (\bar{A}_{22},\bar{A}_{32},A_{42},\bar{A}_{52}), \\
& (\bar{A}_{22},\bar{A}_{32},\bar{A}_{42},A_{52}), & (A_{22},A_{32},\bar{A}_{42},\bar{A}_{52}), & (A_{22},\bar{A}_{32},A_{42},\bar{A}_{52}), \\
& (A_{22},\bar{A}_{32},\bar{A}_{42},A_{52}), & (\bar{A}_{22},A_{32},A_{42},\bar{A}_{52}), & (\bar{A}_{22},A_{32},\bar{A}_{42},A_{52}), \\
& (\bar{A}_{22},\bar{A}_{32},A_{42},A_{52}), & (A_{22},A_{32},A_{42},\bar{A}_{52}), & (A_{22},A_{32},\bar{A}_{42},A_{52}), \\
& (A_{22},\bar{A}_{32},A_{42},A_{52}), & (\bar{A}_{22},A_{32},A_{42},A_{52}), & (A_{22},A_{32},A_{42},A_{52})\}.
\end{array}$$

$$\begin{aligned}
M_3 = \{ & (A_{23},\bar{A}_{33},\bar{A}_{43},\bar{A}_{53}), && (\bar{A}_{23},A_{33},\bar{A}_{43},\bar{A}_{53}), && (\bar{A}_{23},\bar{A}_{33},A_{43},\bar{A}_{53}), \\
& (\bar{A}_{23},\bar{A}_{33},\bar{A}_{43},A_{53}), && (A_{23},A_{33},\bar{A}_{43},\bar{A}_{53}), && (A_{23},\bar{A}_{33},A_{43},\bar{A}_{53}), \\
& (A_{23},\bar{A}_{33},\bar{A}_{43},A_{53}), && (\bar{A}_{23},A_{33},A_{43},\bar{A}_{53}), && (\bar{A}_{23},A_{33},\bar{A}_{43},A_{53}), \\
& (\bar{A}_{23},\bar{A}_{33},A_{43},A_{53}), && (A_{23},A_{33},A_{43},\bar{A}_{53}), && (A_{23},A_{33},\bar{A}_{43},A_{53}), \\
& (A_{23},\bar{A}_{33},A_{43},A_{53}), && (\bar{A}_{23},A_{33},A_{43},A_{53}), && (A_{23},A_{33},A_{43},A_{53})\}.
\end{aligned}$$

$$\begin{aligned}
M_4 = \{ & (A_{11},\bar{A}_{31},\bar{A}_{52},\bar{A}_{43}), && (A_{11},A_{31},\bar{A}_{52},\bar{A}_{43}), && (A_{11},\bar{A}_{31},A_{52},\bar{A}_{43}), \\
& (A_{11},\bar{A}_{31},\bar{A}_{52},A_{43}), && (\bar{A}_{11},A_{31},A_{52},\bar{A}_{43}), && (\bar{A}_{11},A_{31},\bar{A}_{52},A_{43}), \\
& (\bar{A}_{11},\bar{A}_{31},A_{52},A_{43}), && (A_{11},A_{31},A_{52},\bar{A}_{43}), && (A_{11},A_{31},\bar{A}_{52},A_{43}), \\
& (A_{11},\bar{A}_{31},A_{52},A_{43}), && (\bar{A}_{11},A_{31},A_{52},A_{43}), && (A_{11},A_{31},A_{52},A_{43})\}.
\end{aligned}$$

$$\begin{aligned}
M_5 = \{ & (A_{12},\bar{A}_{41},\bar{A}_{32},\bar{A}_{53}), && (A_{12},A_{41},\bar{A}_{32},\bar{A}_{53}), && (A_{12},\bar{A}_{41},A_{32},\bar{A}_{53}), \\
& (A_{12},\bar{A}_{41},\bar{A}_{32},A_{53}), && (\bar{A}_{12},A_{41},A_{32},\bar{A}_{53}), && (\bar{A}_{12},A_{41},\bar{A}_{32},A_{53}), \\
& (\bar{A}_{12},\bar{A}_{41},A_{32},A_{53}), && (A_{12},A_{41},A_{32},\bar{A}_{53}), && (A_{12},A_{41},\bar{A}_{32},A_{53}) \\
& (A_{12},\bar{A}_{41},A_{32},A_{53}), && (\bar{A}_{12},A_{41},A_{32},A_{53}), && (A_{12},A_{41},A_{32},A_{53})\}.
\end{aligned}$$

$$\begin{aligned}
M_6 = \{ & (A_{13},\bar{A}_{51},\bar{A}_{42},\bar{A}_{33}), && (A_{13},A_{51},\bar{A}_{42},\bar{A}_{33}), && (A_{13},\bar{A}_{51},A_{42},\bar{A}_{33}), \\
& (A_{13},\bar{A}_{51},\bar{A}_{42},A_{33}), && (\bar{A}_{13},A_{51},A_{42},\bar{A}_{33}), && (\bar{A}_{13},A_{51},\bar{A}_{42},A_{33}), \\
& (\bar{A}_{13},\bar{A}_{51},A_{42},A_{33}), && (A_{13},A_{51},A_{42},\bar{A}_{33}), && (A_{13},A_{51},\bar{A}_{42},A_{33}), \\
& (A_{13},\bar{A}_{51},A_{42},A_{33}), && (\bar{A}_{13},A_{51},A_{42},A_{33}), && (A_{13},A_{51},A_{42},A_{33})\}.
\end{aligned}$$

$$\begin{aligned}
M_7 = \{ & (A_{11},A_{21},\bar{A}_{42},\bar{A}_{53}), && (A_{11},\bar{A}_{21},A_{42},\bar{A}_{53}), && (A_{11},\bar{A}_{21},\bar{A}_{42},A_{53}), \\
& (\bar{A}_{11},A_{21},A_{42},\bar{A}_{53}), && (\bar{A}_{11},A_{21},\bar{A}_{42},A_{53}), && (\bar{A}_{11},\bar{A}_{21},A_{42},A_{53}), \\
& (A_{11},A_{21},A_{42},\bar{A}_{53}), && (A_{11},A_{21},\bar{A}_{42},A_{53}), && (A_{11},\bar{A}_{21},A_{42},A_{53}), \\
& (\bar{A}_{11},A_{21},A_{42},A_{53}), && (A_{11},A_{21},A_{42},A_{53})\}.
\end{aligned}$$

$$\begin{aligned}
M_8 = \{ & (A_{12},A_{51},\bar{A}_{22},\bar{A}_{43}), && (A_{12},\bar{A}_{51},A_{22},\bar{A}_{43}), && (A_{12},\bar{A}_{51},\bar{A}_{22},A_{43}), \\
& (\bar{A}_{12},A_{51},A_{22},\bar{A}_{43}), && (\bar{A}_{12},A_{51},\bar{A}_{22},A_{43}), && (\bar{A}_{12},\bar{A}_{51},A_{22},A_{43}), \\
& (A_{12},A_{51},A_{22},\bar{A}_{43}), && (A_{12},A_{51},\bar{A}_{22},A_{43}), && (A_{12},\bar{A}_{51},A_{22},A_{43}), \\
& (\bar{A}_{12},A_{51},A_{22},A_{43}), && (A_{12},A_{51},A_{22},A_{43})\}.
\end{aligned}$$

$$\begin{aligned} M_9 =\{ &(A_{13},A_{41},\bar{A}_{52},\bar{A}_{23}), \quad (A_{13},\bar{A}_{41},A_{52},\bar{A}_{23}), \quad (A_{13},\bar{A}_{41},\bar{A}_{52},A_{23}), \\ &(\bar{A}_{13},A_{41},A_{52},\bar{A}_{23}), \quad (\bar{A}_{13},A_{41},\bar{A}_{52},A_{23}), \quad (\bar{A}_{13},\bar{A}_{41},A_{52},A_{23}), \\ &(A_{13},A_{41},A_{52},\bar{A}_{23}), \quad (A_{13},A_{41},\bar{A}_{52},A_{23}), \quad (A_{13},\bar{A}_{41},A_{52},A_{23}), \\ &(\bar{A}_{13},A_{41},A_{52},A_{23}), \quad (A_{13},A_{41},A_{52},A_{23})\}. \end{aligned}$$

$$\begin{aligned} M_{10} =\{ &(A_{11},A_{51},\bar{A}_{32},\bar{A}_{23}), \quad (A_{11},\bar{A}_{51},A_{32},\bar{A}_{23}), \quad (A_{11},\bar{A}_{51},\bar{A}_{32},A_{23}), \\ &(\bar{A}_{11},A_{51},A_{32},\bar{A}_{23}), \quad (\bar{A}_{11},A_{51},\bar{A}_{32},A_{23}), \quad (\bar{A}_{11},\bar{A}_{51},A_{32},A_{23}), \\ &(A_{11},A_{51},A_{32},\bar{A}_{23}), \quad (A_{11},A_{51},\bar{A}_{32},A_{23}), \quad (A_{11},\bar{A}_{51},A_{32},A_{23}), \\ &(\bar{A}_{11},A_{51},A_{32},A_{23}), \quad (A_{11},A_{51},A_{32},A_{23})\}. \end{aligned}$$

$$\begin{aligned} M_{11} =\{ &(A_{12},A_{21},\bar{A}_{52},\bar{A}_{33}), \quad (A_{12},\bar{A}_{21},A_{52},\bar{A}_{33}), \quad (A_{12},\bar{A}_{21},\bar{A}_{52},A_{33}), \\ &(\bar{A}_{12},A_{21},A_{52},\bar{A}_{33}), \quad (\bar{A}_{12},A_{21},\bar{A}_{52},A_{33}), \quad (\bar{A}_{12},\bar{A}_{21},A_{52},A_{33}), \\ &(A_{12},A_{21},A_{52},\bar{A}_{33}), \quad (A_{12},A_{21},\bar{A}_{52},A_{33}), \quad (A_{12},\bar{A}_{21},A_{52},A_{33}), \\ &(\bar{A}_{12},A_{21},A_{52},A_{33}), \quad (A_{12},A_{21},A_{52},A_{33})\}. \end{aligned}$$

$$\begin{aligned} M_{12} =\{ &(A_{13},A_{31},\bar{A}_{22},\bar{A}_{53}), \quad (A_{13},\bar{A}_{31},A_{22},\bar{A}_{53}), \quad (A_{13},\bar{A}_{31},\bar{A}_{22},A_{53}), \\ &(\bar{A}_{13},A_{31},A_{22},\bar{A}_{53}), \quad (\bar{A}_{13},A_{31},\bar{A}_{22},A_{53}), \quad (\bar{A}_{13},\bar{A}_{31},A_{22},A_{53}), \\ &(A_{13},A_{31},A_{22},\bar{A}_{53}), \quad (A_{13},A_{31},\bar{A}_{22},A_{53}), \quad (A_{13},\bar{A}_{31},A_{22},A_{53}), \\ &(\bar{A}_{13},A_{31},A_{22},A_{53}), \quad (A_{13},A_{31},A_{22},A_{53})\}. \end{aligned}$$

$$\begin{aligned} M_{13} =\{ &(A_{11},A_{41},\bar{A}_{22},\bar{A}_{33}), \quad (A_{11},\bar{A}_{41},A_{22},\bar{A}_{33}), \quad (A_{11},\bar{A}_{41},\bar{A}_{22},A_{33}), \\ &(\bar{A}_{11},A_{41},A_{22},\bar{A}_{33}), \quad (\bar{A}_{11},A_{41},\bar{A}_{22},A_{33}), \quad (\bar{A}_{11},\bar{A}_{41},A_{22},A_{33}), \\ &(A_{11},A_{41},A_{22},\bar{A}_{33}), \quad (A_{11},A_{41},\bar{A}_{22},A_{33}), \quad (A_{11},\bar{A}_{41},A_{22},A_{33}), \\ &(\bar{A}_{11},A_{41},A_{22},A_{33}), \quad (A_{11},A_{41},A_{22},A_{33})\}. \end{aligned}$$

$$\begin{aligned} M_{14} =\{ &(A_{12},A_{31},\bar{A}_{42},\bar{A}_{23}), \quad (A_{12},\bar{A}_{31},A_{42},\bar{A}_{23}), \quad (A_{12},\bar{A}_{31},\bar{A}_{42},A_{23}), \\ &(\bar{A}_{12},A_{31},A_{42},\bar{A}_{23}), \quad (\bar{A}_{12},A_{31},\bar{A}_{42},A_{23}), \quad (\bar{A}_{12},\bar{A}_{31},A_{42},A_{23}), \\ &(A_{12},A_{31},A_{42},\bar{A}_{23}), \quad (A_{12},A_{31},\bar{A}_{42},A_{23}), \quad (A_{12},\bar{A}_{31},A_{42},A_{23}), \\ &(\bar{A}_{12},A_{31},A_{42},A_{23}), \quad (A_{12},A_{31},A_{42},A_{23})\}. \end{aligned}$$

$$\begin{aligned} M_{15} =\{ &(A_{13},A_{21},\bar{A}_{32},\bar{A}_{43}), \quad (A_{13},\bar{A}_{21},A_{32},\bar{A}_{43}), \quad (A_{13},\bar{A}_{21},\bar{A}_{32},A_{43}), \\ &(\bar{A}_{13},A_{21},A_{32},\bar{A}_{43}), \quad (\bar{A}_{13},A_{21},\bar{A}_{32},A_{43}), \quad (\bar{A}_{13},\bar{A}_{21},A_{32},A_{43}), \end{aligned}$$

$(A_{13},A_{21},A_{32},\bar{A}_{43})$, $(A_{13},A_{21},\bar{A}_{32},A_{43})$ $(A_{13},\bar{A}_{21},A_{32},A_{43})$
$(\bar{A}_{13},A_{21},A_{32},A_{43})$, $(A_{13},A_{21},A_{32},A_{43})\}$.

In the preceding, $(A_{21},\bar{A}_{31},\bar{A}_{41},\bar{A}_{51})$ corresponds to the individuals who have A_{21}, A_{32} or A_{33}, A_{42} or A_{43}, A_{52} or A_{53}, and any level of A_1 while $(A_{21},A_{31},A_{41},A_{51})$ corresponds to those having $A_{21},A_{31},A_{41},A_{52}$ or A_{53}, and any level of A_1.

To retrieve the query $A = \{A_{11}, A_{21}\}$, first it is necessary to determine B_7 as the first block which contains the set; i.e., $\gamma(A) = 7$. Hence, M_7 is identified as the bucket from which retrieval is to be performed. The relevant sub-buckets are $(A_{11},A_{21},\bar{A}_{42},\bar{A}_{53})$, $(A_{11},A_{21},A_{42},\bar{A}_{53})$, $(A_{11},A_{21},\bar{A}_{42},A_{53})$, and $(A_{11},A_{21},A_{42},A_{53})$ since all the records associated with these subsets have A_{11}, A_{21}. Hence, this two-fold query may be efficiently retrieved.

Next, let us consider the query $A = \{A_{11}\}$. The first block to contain A_{11} is B_4; hence $\gamma(A) = 4$ and the relevant bucket is M_4. The sub-buckets retrieved are $(A_{11},\bar{A}_{31},\bar{A}_{52},\bar{A}_{43})$, $(A_{11},A_{31},\bar{A}_{52},\bar{A}_{43})$, $(A_{11},\bar{A}_{31},A_{52},\bar{A}_{43})$, $(A_{11},\bar{A}_{31},\bar{A}_{52},A_{43})$, $(A_{11},A_{31},A_{52},\bar{A}_{43})$, $(A_{11},A_{31},\bar{A}_{52},A_{43})$, $(A_{11},\bar{A}_{31},A_{52},A_{43})$, $(A_{11},A_{31},A_{52},A_{43})$.

By procedures similar to those indicated above, any one-fold or two-fold query can be readily retrieved. Also, the system can handle a limited number of three-fold and four-fold queries. The principal operations required in this type of filing system are that of determining whether one set contains another. These are performed in areas where the contents of the buckets and sub-buckets are stored. After the appropriate identifications have been performed, the addresses which correspond to the pertinent accession numbers are located by chaining.

2.3. Second order combinatorial configurations based on incomplete designs.

The problem of constructing second order combinatorial configurations is essentially the same as that of constructing certain incomplete block designs used in statistical research. Of special interest are balanced incomplete block designs and group divisible designs. The combinatorial properties of these designs have received much attention in the literature. In particular, the reader is referred to Bose [3], Bose [4], Bose, Shrikhande, and Bhattacharya [17], Rao [39], Sprott [46].

2.3.1. Balanced incomplete block designs.

A balanced incomplete block (BIB) design is an arrangement of v objects into b subsets called blocks such that

i. each block contains k objects

ii. each object occurs in r distinct blocks

iii. each pair of objects occurs together in λ distinct blocks.

If only one level of each attribute is of interest from the point of view of retrieval as in sub-section 1.4.6, then a BIB design with parameters (v, b, r, k, $\lambda = 1$) represents a combinatorial configuration $(\Omega, k, 2, b)$ where $\Omega = \{A_1, A_2, \ldots, A_v\}$ denotes the set of v attributes. Such configurations are optimal in the sense that each pair of attributes is covered exactly once, and hence for the given k, b is a minimum. Thus, the formation of optimal combinatorial filing systems appropriate for two-fold or one-fold queries may be based on the construction of BIB designs with $\lambda = 1$.

As was the case with the balanced filing systems considered earlier, such BIB designs may be obtained from finite geometries. In particular, points are identified with objects (attributes) and lines with blocks (buckets). The resulting schemes are similar to the balanced filing systems except for the fact that sub-buckets are formed in accordance with section 2.2.

More generally, Bose [3], [4] has given some fundamental theorems which may be used to form BIB designs. These methods were then applied to the construction of some designs in the following series.

T_1: $v = 3(2t + 1)$, $b = (3t + 1)(2t + 1)$, $r = 3t + 1$, $k = 3$, $\lambda = 1$

T_2: $v = 6t + 1$, $b = t(6t + 1)$, $r = 3t$, $k = 3$, $\lambda = 1$

F_1: $v = 12t + 1$, $b = t(12t + 1)$, $r = 4t$, $k = 4$, $\lambda = 1$

F_2: $v = 4(3t + 1)$, $b = (4t + 1)(3t + 1)$, $r = 4t + 1$, $k = 4$, $\lambda = 1$

G_1: $v = 20t + 1$, $b = t(20t + 1)$, $r = 5t$, $k = 5$, $\lambda = 1$

G_2: $v = 20t + 5$, $b = (5t + 1)(4t + 1)$, $r = 5t + 1$, $k = 5$, $\lambda = 1$

The actual existence of the designs belonging to F_1, F_2, G_1, G_2 depend upon further conditions given in the cited references. Some examples where the contitions are satisfied are

v	b	r	k	λ
13	13	4	4	1
25	50	8	4	1
16	20	5	4	1
28	63	9	4	1
41	82	10	5	1
61	183	15	5	1
25	30	6	5	1
45	99	11	5	1

Rao [39] and Sprott [46] indicate the construction of the following additional designs with k = 4, 5

v	b	r	k	λ
37	111	12	4	1
40	130	13	4	1
65	208	16	5	1

as well as the designs

v	b	r	k	λ
66	143	13	6	1
91	195	15	7	1
81	216	16	6	1
91	273	18	6	1
96	304	19	6	1
113	226	16	8	1
120	255	17	8	1
153	323	19	9	1
145	290	18	9	1
145	232	16	10	1
181	362	20	10	1

For situations with larger v, additional BIB designs need to be developed. Also, designs with large k would be desirable, particularly in instances where multi-stage schemes (as will be discussed later) are envisioned. The problems posed here are not easily solved and will require additional research.

2.3.2. Group divisible designs.

A group divisible (GD) design is an arrangement of vs objects, belonging to v groups of s objects each, into b blocks such that

i. each block contains k objects

ii. each object occurs in r distinct blocks

iii. each pair of objects, belonging to the same group, occur together in λ_1 blocks

iv. each pair of objects, belonging to different groups, occur together in λ_2 blocks.

Hence, a GD design with $\lambda_1 = 0$, $\lambda_2 = 1$ represents a combinatorial configuration (Ω, k, 2, b) appropriate to the multi-level attribute case with Ω being the set of v attributes, each with s levels. Such configurations are optimal in the sense that each pair of levels of different attributes is covered exactly once, and hence for the given k, b is a minimum. Thus, optimal combinatorial filing schemes appropriate for two-fold and one-fold queries may be constructed if the corresponding GD designs with $\lambda_1 = 0$, $\lambda_2 = 1$, exist.

Bose, Shrikhande, and Bhattacharya [17] give the following simple method of constructing group divisible designs.

Theorem (2.3.1.). By omitting a particular treatment φ and all blocks containing it from a BIB design with $\lambda = 1$, one obtains a group divisible design with $\lambda_1 = 0$, $\lambda_2 = 1$.

They list the following designs as obtainable by this method

v	s	b	r	k	λ_1	λ_2
5	3	15	4	4	0	1
6	4	24	5	5	0	1
8	6	48	7	7	0	1
9	7	63	8	8	0	1
10	8	80	9	9	0	1
6	2	20	5	3	0	1
7	2	28	6	3	0	1
8	3	42	7	4	0	1
9	2	48	8	3	0	1
9	3	54	8	4	0	1
10	2	60	9	3	0	1
10	4	72	9	5	0	1
11	4	88	10	5	0	1

They also give methods for constructing the following additional designs

v	s	b	r	k	λ_1	λ_2
7	2	14	4	4	0	1
13	2	52	8	4	0	1
4	4	32	6	3	0	1
4	6	72	9	3	0	1
5	3	30	6	3	0	1
6	4	80	10	3	0	1
4	4	16	4	4	0	1
5	5	25	5	5	0	1
7	7	49	7	7	0	1
8	8	64	8	8	0	1

v	s	b	r	k	λ_1	λ_2
9	9	81	9	9	0	1
7	4	54	8	4	0	1
7	3	63	9	3	0	1
9	5	90	10	5	0	1

Some of the above designs as well as some others which are not listed may be obtained from finite geometries by procedures similar to those illustrated in Theorems (1.4.1) and (1.4.2). Indeed, the buckets of the balanced multiple filing schemes constructed there coincide with the blocks of a corresponding GD design. Also, GD designs may be extracted from the BIB designs with $r \geqq 11$ listed in the previous subsection by applying Theorem (2.3.1). As a final note, further research is required toward the construction on GD designs for larger values of v, s, and k.

2.4. Combinatorial configurations for the case when retrieval pertains to only one level of each attribute.

Here, we shall indicate some methods of construction given by Ray-Chaudhuri [43] for general (Ω, k, t, b) configurations where $\Omega = \{A_1, A_2, \ldots, A_v\}$.

2.4.1. Configurations based on coverings of a finite projective space.

An m-flat π in PG(N,q) is said to cover a (t-1)-flat Σ if $\Sigma \subseteq \pi$ where $N \geq m \geq t-1$. A class of m-flats $(\pi_1, \pi_2, \ldots, \pi_b)$ is defined to be a (b, t, m)-cover if every (t-1)-flat in PG(N,q) **is contained in at** least one of the m-flats π_h belonging to the class. The function b(N, t, m, q) will be used to represent the smallest value of b for which there exists a (b, t, m)-cover, in which case the cover is called a minimum (b, t, m)-cover. Given the above framework, we now prove the following theorem of Ray-Chaudhuri [43].

Theorem (2.4.1). There exists an (Ω, k, t, b) combinatorial configuration for the case $v = (q^{N+1} - 1)/(q-1)$, $k = (q^{m+1} - 1)/(q - 1)$ $b = b(N, t, m, q)$ where $N \geqq m \geqq t-1$ and $q = p^u$ with p being a prime.

Proof: Let $(\pi_1, \pi_2, \ldots, \pi_b)$ be a class of m-flats in PG(N,q) which covers all (t-1) flats and $b = b(N, t, m, q)$. Since $m \geq t-1$, such a covering exists. Let the points of PG(N,q) be identified with the elements of Ω and let the m-flats $\pi_1, \pi_2, \ldots, \pi_b$ be identified with the blocks $B_1, B_2, \ldots, B_b$ of the configuration. Because any set of t-points or less is contained in some (t-1)-flat and hence is contained in one of the m-flats of the cover, the resulting construction is an (Ω, k, t, b) configuration with parameters as listed above.

If the number of attributes is less than $(q^{N+1} - 1)/(q-1)$, Theorem (2.4.1) may still be applied because if all elements of Ω except those in a subset Ω^* are deleted from Ω and from each of the blocks $B_1, B_2, \ldots, B_b$, the resulting system becomes an (Ω^*, k, t, b) configuration. This follows because an upper bound on the block size is still k and all t-plets of elements from Ω^* are still covered. Actually, b may become smaller, if the deletion process causes some blocks to have fewer than t elements, in which case they may also be discarded.

Ray-Chaudhuri does not discuss to any great extent how minimum (b, t, m)-covers are constructed. He does, however, indicate the following corollary.

Corollary (2.4.1.1). An (Ω, k, t, b) combinatorial configuration exists for $v = (q^{N+1} - 1)/(q-1)$, $k = (q^t - 1)/(q-1)$, $b = \Phi(N, t-1, q)$.

Proof: Take $m = t-1$ in Theorem (2.4.1).

Let us now supplement the above result with some additional methods for forming covers. Except for the simplest cases, no claim of optimality is made for the constructions given. However, they are felt to be satisfactory until better ones become known.

Theorem (2.4.2). There exists a $(b, 1, 1)$-cover for the geometry $PG(N,q)$ with $b = \Phi(N-1, 0, q)$ where q is a prime power.

Proof: Consider the set of $b = \Phi(N-1, 0, q)$ lines through some fixed point P_0 in $PG(N,q)$. This represents a $(b, 1, 1)$-cover since each point of $PG(N,q)$ lies on exactly one of the lines through P_0.

Corollary (2.4.2.1). There exists an optimal $(b, 1, 1)$-cover for $PG(2,q)$ where q is a prime power; i.e., $b(2, 1, 1, q) = q+1$.

Proof: Since each line in $PG(2,q)$ contains $(q+1)$ points, we have that $b(2, 1, 1, q) \geq (q^2 + q + 1)/(q+1) > q$; i.e., $b(2, 1, 1, q) \geq (q+1)$. But the construction from Theorem (2.4.2) has $b = \Phi(1, 0, q) = q+1$ and hence is optimal.

Theorem (2.4.3). There exists a $(b, 1, m)$-cover for the geometry $PG(N,q)$ with $b = \Phi(N-m, 0, q)$ where q is a prime power and $N \geq m$.

Proof: Consider the set of $b = \Phi(N-m, 0, q)$ m-flats through some fixed $(m-1)$-flat π_0 in $PG(N,q)$. This represents a $(b, 1, m)$-cover since each point of $PG(N, q)$ which is not on π_0 determines a unique m-flat through π_0.

<u>Corollary (2.4.3.1)</u>. There exists an optimal (b, 1, m)-cover for PG(m+1, q) where q is a prime power; i.e., b (m+1, 1, m, q) = (q+1).

Since each m-flat in PG(m+1, q) contains $(q^{m+1} - 1)/(q-1)$ points, we have that $b(m+1, 1, m, q) \geq (q^{m+2} - 1)/(q^{m+1} - 1) > q$; i.e., $b(m+1, 1, m, q) \geq q+1$. But the construction of Theorem (2.4.3) has $b = \Phi(1, 0, q) = q+1$.

<u>Theorem (2.4.4)</u>. There exists a (b, 2, 2)-cover for the geometry PG(3, q) with $b = q^2 + q + 1$ where q is a prime power.

<u>Proof</u>: Consider all lines in a given plane π_0 through a given point P_0. There are (q+1) lines in π_0 through P_0. Form the (b, 2, 2)-cover by taking all planes through these lines other than π_0 together with π_0. Hence, $b = q(q+1) + 1 = q^2 + q + 1$. Since any line L, not in π_0, intersects π_0 in a point P_1 and since the line connecting P_0 and P_1 and the line L determine a unique plane, each line is covered.

<u>Theorem (2.4.5)</u>. There exists a (b, 2, 2)-cover for the geometry PG(N,q) with $b = \sum_{\alpha=0}^{N-2} q^{\alpha}\Phi(\alpha, 0, q)$ where q is a prime power and $N \geq 2$.

<u>Proof</u>: Consider all lines, in a given (N-1)-flat π_0, through a given point P_0. There are $\Phi(N-2, 0, q)$ such lines. Form the (b, 2, 2)-cover of PG(N,q) by taking all planes through these lines other than those lying in π_0 together with a (b, 2, 2)-cover of PG(N-1, q) as represented by π_0. Since any line L, not in π_0, intersects π_0 in a point P_1 and since the line connecting P_0 and P_1 and the line L determine a unique plane, each line not in π_0 is covered. Those lines in π_0 are covered by the (b, 2, 2)-cover

developed for PG(N-1, q) which may be assumed to exist by mathematical induction because of the existence of the (b, 2, 2)-cover of PG(3, q) given in Theorem (2.4.4). The value of b is determined by noting that through each of the $\Phi(N-2, 0, q)$ lines through P_0 in π_0, there pass $\{\Phi(N-2, 0, q) - \Phi(N-3, 0, q)\}$ planes not lying in π_0. Thus, b is equal to the sum of $q^{N-2}\,\Phi(N-2, 0, q)$ and the number of planes needed to cover the (N-1)-flat π_0. Proceeding backwards in a recursive manner, we have $b = \sum_{\alpha=0}^{N-2} q^{\alpha}\,\Phi(\alpha, 0, q)$.

Theorem (2.4.6). There exists a (b, 2, m)-cover for the geometry PG(N,q) with $b = \sum_{\alpha}^{N-m} q^{\alpha}\,\Phi(\alpha, 0, q)$, where q is a prime power and $N \geq m \geq 2$.

Proof: Consider all (m-1)-flats, in a given (N-1)-flat π_0, through some fixed (m-2)-flat π_1. There are $\Phi(N-m, 0, q)$ such (m-1)-flats. Form the (b, 2, m)-cover of PG(N,q) by taking all m-flats through these (m-1)-flats other than those lying in π_0 together with a (b, 2, m)-cover of PG(N-1, q) as represented by π_0. Since any line L, not in π_0, intersects π_0 in a unique point P_1 which lies in an (m-1)-flat π such that $\pi_1 \subseteq \pi \subseteq \pi_0$ and since L and π determine a unique m-flat, each line not in π determine a unique m-flat, each line not in π_0 is covered. Those lines belonging to π_0 are covered by the (b, 2, m)-cover similarly developed for PG(N-1, q), the existence of which may be assumed by mathematical induction since a (b, 2, m)-cover exists for PG(m, q). In particular, for the case N = m, b = 1; for the case N = m+1, b = q(q+1) + 1; and in general b is the sum of $\Phi(N-m, 0, q)\,\{\Phi(N-m, 0, q) - \Phi(N-m-1, 0, q)\}$ and the number of m-flats needed to cover the (N-1)-flat π_0. Proceeding recursively, we have $b = \sum_{\alpha=0}^{N-m} q^{\alpha}\,\Phi(\alpha, 0, q)$.

Theorem (2.4.7). There exists a (b, 3, m)-cover for the geometry PG(N, q) with $b = \sum_{\beta=0}^{N-m} \sum_{\alpha=0}^{\beta} q^{\alpha+\beta} \Phi(\alpha, 0, q)$ where q is a prime power and $N \geq m \geq 3$.

Proof: Consider the (m-1)-flats belonging to a ($\bar{b}$, 2, m-1)-cover for the geometry PG(N-1, q) as represented by an (N-1)-flat π_0. Form the (b, 3, m)-cover of PG(N, q) by taking all m-flats through these (m-1)-flats other than those lying in π_0 together with a (b, 3, m)-cover of PG(N-1, q) as represented by π_0. Since any plane P, not in π_0, intersects π_0 in a unique line lying in an (m-1)-flat π which belongs to the (b, 2, m-1)-cover in π_0 defined above and since P and π determine a unique m-flat through π, each plane not in π_0 is covered. Those planes belonging to π_0 are covered by the (b, 3, m)-cover similarly developed for PG(N-1, q), the existence of which may be assumed by mathematical induction since a (b, 3, m)-cover exists for PG(m, q). In particular, for the case N=m, b=1; for the case N=m+1, $b = (q^2 + q + 1)\, q + 1 = q^3 + q^2 + q + 1$; and in general, b is the sum of

$$\{\sum_{\alpha=0}^{N-m} q^{\alpha} \Phi(\alpha, 0, q)\} \; \{\Phi(N-m, 0, q) - \Phi(N-m-1, 0, q)\}$$

and the number of m-flats needed to cover the (N-1)-flat π_0. Proceeding recursively, we have

$$b = \sum_{\beta=0}^{N-m} q^{\beta} \sum_{\alpha=0}^{\beta} q^{\alpha} \Phi(\alpha, 0, q) .$$

<u>Theorem (2.4.8)</u>. There exists a (b, 4, m)-cover for the geometry PG(N,q) with $b = \sum_{\gamma=0}^{N-m} \sum_{\beta=0}^{\gamma} \sum_{\alpha=0}^{\beta} q^{\alpha+\beta+\gamma} \Phi(\alpha, 0, q)$ where q is a prime power and $N \geq m \geq 4$.

<u>Proof</u>: Consider the (m-1)-flats belonging to a (b, 3, m-1)-cover for the geometry PG(N-1,q) as represented by an (N-1)-flat π_0. Form the (b, 4, m)-cover of PG(N,q) by taking all m-flats through these (m-1)-flats other than those lying in π_0 together with a (b, 4, m)-cover of PG(N-1,q) as represented by π_0. Since any 3-flat π_3, not in π_0, intersects π_0 in a unique plane lying in a (m-1)-flat π which belongs to the (b, 3, m-1)-cover of π_0 defined above and since π_3 and π determine a unique m-flat through π , each 3-flat not in π_0 is covered. Those 3-flats belonging to π_0 are covered by the (b, 4, m)-cover similarly developed for PG(N-1,q), the existence of which may be assumed by mathematical induction since a (b, 4, m)-cover exists for PG(m,q) In particular, for the case N=m, b=1; for the case N=m+1, $b = q(q^3 + q^2 + q + 1) + 1 = q^4 + q^3 + q^2 + q + 1$; and in general, b is the sum of

$$\{\sum_{\beta=0}^{N-m} \sum_{\alpha=0}^{\beta} q^{\alpha+\beta} \Phi(\alpha, 0, q)\} \{\Phi(N-m, 0, q) - \Phi(N-m-1, 0, q)\}$$

and the number of m-flats needed to cover the (N-1)-flat π_0 .

Proceeding recursively, we have

$$b = \sum_{\gamma=0}^{N-m} q^{\gamma} \sum_{\beta=0}^{\gamma} \sum_{\alpha=0}^{\beta} q^{\alpha+\beta} \Phi(\alpha, 0, q) .$$

By proceeding in a step-wise fashion according to the previous theorems, a (b, t, m)-cover may be readily constructed for PG(N,q) where q is a prime power and $N \geq m \geq t-1$. It is reasonable to believe that

$$b = \sum_{\alpha_t=0}^{N-m} \sum_{\alpha_{t-1}=0}^{\alpha_t} \cdots \sum_{\alpha_1=0}^{\alpha_2} q^{\alpha_1+\alpha_2+\dots+\alpha_t} \Phi(\alpha_1, 0, q) \text{ for these cases.}$$

As indicated earlier, no claim of optimality is made for these constructions. However, they are easily formed and should provide useful bases for combinatorial filing systems until a more optimal class of covers or other type of configuration is developed.

2.4.2. Configurations based on non-linear surfaces in finite-geometries.

A subset S of points in PG(N,q) is called a cap of order d where $0 \leq d \leq N+1$ if no subset of d points from S lie in a (d-2)-flat. Bose [5] used the concept of caps in the design of factorial experiments. He termed the problem of determining the maximum number of points on a cap of order d the packing problem. Solutions to this problem are only available in a few special cases - e.g., d=2, N and q arbitrary; d=3, q=2, N arbitrary; d=3, N=2 or 3, q arbitrary.

On the other hand, the results on Bose-Chaudhuri [14] codes represent a basis from which a useful series of caps may be constructed. They prove the following theorem.

Theorem (2.4.8). Let q be a prime power and let n_0 be an integer relatively prime to q. Then there exists a cap of order d with n_0 points on it in the geometry PG(N-1, q) where N=ud and u satisfies $q^u - 1 = cn_0$. These points are the columns of the matrix H when considered with respect to GF(q).

$$H = \begin{bmatrix} 1 & \theta^{g} & \theta^{2g} & \cdots & \theta^{(n_0-1)g} \\ 1 & \theta^{g+h} & \theta^{2(g+h)} & \cdots & \theta^{(n_0-1)(g+h)} \\ \cdots & & & & \\ 1 & \theta^{g+(d-1)h} & \theta^{2(g+(d-1)h)} & \cdots & \theta^{(n_0-1)(g+(d-1)h)} \end{bmatrix}$$

In the above $\theta=\beta^c$ where β is a primitive element of $GF(q^m)$, g is arbitrary, h is arbitrary except for being relatively prime to n_0, and $2 \leq d \leq n_0-2$. Moreover, if the rank of H is N_0, then when all rows of H except N_0 independent ones are deleted, the resulting array represents a cap of order d in $PG(N_0-1, q)$.

A proof of Theorem (2.4.8) may be found in either Bose and Ray-Chaudhuri [14], [15], in Peterson [33], or in Ray-Chaudhuri [43]. Caps of order d may be used in the construction of configurations as follows.

Theorem (2.4.9). For any integer v, there exists an (Ω, k, t, b) combinatorial configuration based on a cap of order d.

Proof: Let the attributes correspond to the points of a cap of order d in $PG(N-1,q)$ where q is a prime power relatively prime to v and where $N \geq d$, $v \geq d$. The existence of such a cap follows from Theorem (2.4.7). Let $(\pi_1, \pi_2, \ldots, \pi_b)$ be m-flats of a (b,t,m)-cover of $PG(N-1,q)$ where $N-1 \geq m \geq t-1$. Define the sets B_h by $B_h = \pi_h \cap \Omega$. The sets B_h are d-caps if $|B_h| \geq d$; otherwise all points in B_h are independent. The number of points in the blocks does not exceed the maximum number of points belonging to a d-cap in $PG(m,q)$. Finally any set of t points belonging to Ω must also be covered by one of the m-flats π_h and hence by one of the blocks B_h.

The previous theorem was given by Ray-Chaudhuri [43]. Although it represents a potentially large and interesting class of combinatorial configurations, it is difficult to apply. This results from the necessity of first finding the relevant d-caps and then finding flat spaces which cover the various subsets of t elements from the d-caps. Finally, the resulting configurations may lack the symmetry of other types of configurations and hence require more blocks. The answers to these questions will require additional research involving perhaps the use of covers based on quadrics or higher degree surfaces instead of flat spaces.

2.5. The use of caps to construct combinatorial configurations in the multi-level attribute case.

In this section, we shall consider a method of construction due to Bose, Abraham, and Ghosh [9], of an (Ω, k, t, b) configuration for a situation in which there are v attributes, each with s=q levels where q is a prime power. The attributes A_i are identified with linear functions L_i given by

$$A_i: \quad L_i = h_{i1} x_1 + h_{i2} x_2 + \dots + h_{iN} x_N \qquad i = 1, 2, \dots, v$$

where the vectors $\underset{\sim}{h}_i' = (h_{i1}, h_{i2}, \dots, h_{iN})$ transposed correspond to the points of a cap of order t in PG(N-1,q) which has at least v points. The construction of such caps follows from Theorem (2.4.7). The various subsets of t or fewer levels of distinct attributes can be identified with sets of equations expressing that corresponding linear functions equal corresponding levels.

For example, if $u_{i_1}, u_{i_2}, \dots, u_{i_g}$ where $g \leq t$ correspond to levels of the i_1-th, i_2-th, ..., i_g-th attributes, then we have the equations

$$h_{i_1 1} x_1 + \dots + h_{i_1 N} x_N = u_{i_1}$$

$$h_{i_2 1} x_1 + \dots + h_{i_2 N} x_N = u_{i_2}$$

$$\dots \qquad \dots$$

$$h_{i_g 1} x_1 + \dots + h_{i_g N} x_N = u_{i_g}$$

Because of the construction of H, the rank of the coefficient matrix in the above equations is g. Hence, the equations can be reduced to an echelon form by vector addition and scalar multiplication as follows:

i. the first non-zero coefficient on the left hand side of each equation is unity

ii. if the first non-zero coefficient in the i-th equation is x_{c_i}, then $c_1 < c_2 < \dots < c_g$

iii. the coefficient of x_{c_i} is zero in every equation except the i-th where $i \leqq g$.

By putting $x_\alpha = 0$ if $\alpha \neq c_1, c_2, \dots, c_g$, a canonical solution vector is obtained with at most g non-zero coordinates. Hence, a correspondence may be defined between the various subsets of t-plets and the set Ω_0 of N-vectors with at most t non-zero coordinates. Since there are at most

$$b_0 = 1 + (q-1)\binom{N}{1} + \dots + (q-1)^t\binom{N}{t}$$

such N-vectors and since the total number of possible t-plets (involving different levels of different attributes) is

$$q\binom{v}{1} + q^2\binom{v}{2} + \ldots + q^t\binom{v}{t}$$

where v is usually greater than N; in this case, there will be a number of different t-plets corresponding to each element of Ω_0.

Let $\underset{\sim}{x}' = (x_1, x_2, \ldots x_N)$ denote a solution vector with at most t non-zero co-ordinates. The block $B(\underset{\sim}{x})$ corresponding to $\underset{\sim}{x}$ in the configuration is given by $B(\underset{\sim}{x}) = \{A_{1u_1}, A_{2u_2}, \ldots, A_{vu_v}\}$ where $u_i = L_i(\underset{\sim}{x})$ for i=1,2,...,v. More than one of the vectors $\underset{\sim}{x}$ may have the same corresponding $B(\underset{\sim}{x})$. However, if the rank of the matrix H with rows $\underset{\sim}{h}_1', \underset{\sim}{h}_2', \ldots \underset{\sim}{h}_v'$ is N, then the sets $B(\underset{\sim}{x})$ and $B(\underset{\sim}{x}^*)$ are different if $\underset{\sim}{x}$ and $\underset{\sim}{x}^*$ are different. This results from the fact that $B(\underset{\sim}{x}) = B(\underset{\sim}{x}^*)$ implies $L_i(\underset{\sim}{x}) = L_i(\underset{\sim}{x}^*)$ and hence that $L_i(\underset{\sim}{x} - \underset{\sim}{x}^*) = 0$ for i = 1,2,..., v; but when the rank of H is N, this means that $\underset{\sim}{x} - \underset{\sim}{x}^*$ must be a null vector.

The previous remarks lead to the following theorem of Bose, Abraham and Ghosh [9].

<u>Theorem (2.5.1).</u> There exists a combinatorial configuration (Ω, v, t, b) for the case in which Ω consists of v attributes, each with s = q levels where q is a prime power. The structure of the configuration is based on the properties of v points lying on a cap of order t in PG(N-1,q) with N being an appropriate function of v, t, q. Finally,

$$b \leq b_0 = \sum_{\alpha=0}^{t} (q-1)^{\alpha}\binom{N}{\alpha}.$$

The filing scheme which they base on Theorem (2.5.1) is somewhat different from that outlined in Section 2.2 in the sense that the blocks $B(\underset{\sim}{x})$ are not ordered; and hence a record is stored in the corresponding bucket if it has any elements at all in common with the block. This leads to increased redundancy. On the other hand, the retrieval scheme they employ makes use of the mechanical abilities of the computer filing system to solve the linear equations associated with queries in order to determine the bucket. This type of operation may be more quickly performed possibly than that of determining the first block set which contains a given query set. Thus, any comparison of the two types of systems will depend upon the properties of the computers to be used and hence will require empirical study.

In the next chapter, we shall consider an alternative method of constructing combinatorial configurations for the multi-level attribute case. The procedure is not quite as general as the one previously described; however, it is fairly simple to apply and provides a reasonably efficient cover with small redundancy.

CHAPTER III

COMBINATORIAL CONFIGURATIONS OBTAINED BY COMPOSITION

3.1. The combinatorial problem.

As was indicated in Section 2.5, the problem of constructing combinatorial configurations with $k = v$ is equivalent to the problem of forming an array of ordered v-tuples (in which each co-ordinate corresponds to a unique attribute) in such a way that every possible ordered combination of t co-ordinates occurs at least once. For the case in which Ω consists of v attributes, each with s levels, and in which all t-plets occur exactly once, such a construction is called an orthogonal array of strength t, constraints v, and index unity and is represented by (b, v, s, t). Such orthogonal arrays have been discussed by Bose and Bush [10], Bush [21] as well as many others. However, for large v, the construction of orthogonal arrays of index unity becomes a very difficult, if not impossible, problem of combinatorial mathematics.

On the other hand, the configurations of interest to us do not require that every t-plet be covered exactly once, but rather at least once. As a result, in some situations the concept of partially balanced array, as defined by Chakravarti [24], [25] is useful.

Definition (3.1.1). A partially balanced array of strength t in b blocks, v attributes with s levels each, is equivalent to a $(b \times v)$ matrix in which among the rows of each t-column sub-matrix, every

possible permutation of the values in the vector $(u_1, u_2, \ldots, u_t)$ occurs exactly $\lambda(u_1, u_2, \ldots, u_t)$ times, independent of which t columns are chosen.

The partially balanced arrays which are of the most interest here are those in which a majority of the $\lambda(u_1, u_2, \ldots, u_t)$ are equal to unity. As with orthogonal arrays, the problem of constructing partially balanced arrays for large v with λ's near unity is another very difficult problem.

The preceding remarks suggest the following method of attack. First attempt to construct a number of efficient orthogonal arrays and **partially** balanced arrays for the cases in which v is small. Then, for larger v, develop a method called composition which expands some of the properties of a small design to the larger ones. The resulting arrays may not have the same symmetry properties as the smaller ones on which they are based. However, when properly formed, they will satisfy the covering requirements appropriate to the corresponding combinatorial configuration. Moreover, in some instances, these methods can be applied to cases in which different attributes assume different numbers of levels (i.e., when the n_i are not necessarily equal). In the subsequent sections of this chapter, the method of composition will be illustrated for combinatorial configurations of orders 2, 3, and 4.

3.2. The construction of configurations of order 2 with k = v.

Let us assume that there are v attributes, each with s levels. A combinatorial configuration with k = v, t = 2 can be represented by a (b x v) matrix in which among the rows of each 2-column sub-matrix, each of the s^2 possible ordered 2-tuples (u_1, u_2) occurs at least once. In this section, a method of constructing such matrices will be discussed for three cases of interest:

Case I : $s = 2$

Case II : $s = q$ where q is a prime power

Case III: s is not a prime power

Finally, some consideration is given to the application of the basic approach used to some situations in which the number of levels assumed by the attributes are not necessarily equal; i.e.,

Case IV : the n_i are not necessarily equal

3.2.1. Case I: $s = 2$.

If $v = 3$, then the orthogonal array (4, 3, 2, 2) of index unity given by

$$\begin{array}{c} 000 \\ 011 \\ 101 \\ 110 \end{array}, \quad b = 4 \qquad (3.2.1)$$

represents an optimal configuration with each of the possible values 00, 01, 10, 11 occurring exactly once among the rows of each 2-column sub-matrix. Similarly, if $v = 4$, then the partially balanced array of strength $t = 2$ given by

$$\begin{array}{c} 0000 \\ 0111 \\ 1011 \\ 1101 \\ 1110 \end{array}, \quad b = 5 \qquad (3.2.2)$$

with $\lambda(1, 0) = \lambda(0, 1) = \lambda(0, 0) = 1$, $\lambda(1, 1) = 2$ represents an optimal configuration since the assignment of two 1's and two 0's to four-tuples in each way possible would lead to six blocks. The above constructions represent efficient configurations for the two small values of v considered. The design (3.2.1) may be extended to the cases v = 6, v = 9, and v = 12 by the following compositions

```
        00 00 00        000 000 000        0000 0000 0000
        00 11 11        000 111 111        0000 1111 1111
        11 00 11        111 000 111        1111 0000 1111
        11 11 00        111 111 000        1111 1111 0000     (3.2.3)
        --------        -----------        --------------
        01 01 01        011 011 011        0111 0111 0111
        10 10 10        101 101 101        1011 1011 1011
                        110 110 110        1101 1101 1101
                                           1110 1110 1110

          b = 6            b = 7              b = 8
```

The first four blocks of each of these arrays are formed by dividing the attributes into three groups and assigning the i-th column of (3.2.1) to each of the attributes in the i-th group. As a result, any pair of values of attributes from different groups is covered in one of these blocks. Also, the pairs of values 00 and 11 are covered for pairs of attributes belonging to the same group. The remaining blocks are then formed by duplicating either (3.2.1) or (3.2.2) (except for the vector of 0's therein) or simply the two pairs 01 and 10 within each group depending on whether there there are 3, 4, or 2 attributes respectively associated with each group. Continuing in a similar manner, the method of composition yields the following designs for the cases v = 18, v = 27, and v = 36.

```
   000000 000000 000000      000000000 000000000 000000000
   000000 111111 111111      000000000 111111111 111111111
   111111 000000 111111      111111111 000000000 111111111
   111111 111111 000000      111111111 111111111 000000000
   --------------------      -----------------------------
   001111 001111 001111      000111111 000111111 000111111
   110011 110011 110011      111000111 111000111 111000111
   111100 111100 111100      111111000 111111000 111111000
   010101 010101 010101      011011011 011011011 011011011
   101010 101010 101010      101101101 101101101 101101101
                             110110110 110110110 110110110

          b = 9                        b = 10
```

```
000000000000 000000000000 000000000000
000000000000 111111111111 111111111111
111111111111 000000000000 111111111111
111111111111 111111111111 000000000000
--------------------------------------
000011111111 000011111111 000011111111
111100001111 111100001111 111100001111
111111110000 111111110000 111111110000
011101110111 011101110111 011101110111
101110111011 101110111011 101110111011
110111011101 110111011101 110111011101
111011101110 111011101110 111011101110
```

$b = 11$

From the nature of the previous constructions, one can see that the composition can be subsequently extended for higher values of v. These results may be stated in terms of the following theorem.

Theorem (3.2.1). For the case of v attributes with two levels each, there exists a second order combinatorial configuration with $k = v$ and b as follows

$$\begin{array}{lll} b = 3u + 1 & \text{if} & 2 \cdot 3^{u-1} < v \leq 3^u \\ b = 3u + 2 & \text{if} & 3^u < v \leq 3^u + 3^{u-1} \\ b = 3u + 3 & \text{if} & 3^u + 3^{u-1} < v \leq 2 \cdot 3^u \end{array}$$

where $u = 1, 2, \ldots$.

Proof: First of all the basic constructions are for the cases in which v has one of the three forms $v = 3^u$, $v = 3^u + 3^{u-1}$, or $v = 2 \cdot 3^u$. When this does not hold true, then the construction is based on the smallest value v^* larger than v which has one of the three indicated forms, in which case the last $(v^* - v)$ columns of the resulting array are deleted.

Suppose v has one of the three forms $v = 3^u$, $v = 3^u + 3^{u-1}$, or $v = 2 \cdot 3^u$, each of which is divisible by 3 when $u > 0$. Then the first group of blocks is given by

$$\begin{array}{ccc} 0^{(v/3)} & 0^{(v/3)} & 0^{(v/3)} \\ 0^{(v/3)} & 1^{(v/3)} & 1^{(v/3)} \\ 1^{(v/3)} & 0^{(v/3)} & 1^{(v/3)} \\ 1^{(v/3)} & 1^{(v/3)} & 0^{(v/3)} \end{array}$$

where $w^{(v/3)}$ means that the value "w" is repeated $(v/3)$ times. Since $(v/3)$ has one of the three forms $(v/3) = 3^{u-1}$, $(v/3) = 3^{u-1} + 3^{u-2}$, or $(v/3) = 2 \cdot 3^{u-1}$, the remaining blocks can be formed by repeating the construction appropriate to $(v/3)$ with the vector of 0's excluded. The existence of such designs has already been demonstrated for the cases $u = 1, 2$. Thus, the result follows by induction.

Finally, since the expression for the number of blocks $b(u)$ as a function of u satisfies the relation $b(u) = 4 + (b(u-1) - 1) = 3 + b(u-1)$, we have the equation $b(u) = 3u + b(0)$ where $b(0)$ is equal to 1, 2, or 3 according to the form of v.

The value of Theorem (3.2.1) is that it provides constructions for which the value of b increases at an additive linear rate as the value of v increases at a multiplicative exponential rate; i.e., for a given v and b, the number of blocks appropriate for 3v is b + 3. In particular, the following table indicates the relative sizes of b and v.

v	b	v	b	v	b
3	4	27	10	243	16
4	5	36	11	324	17
6	6	54	12	486	18
9	7	81	13	729	19
12	8	108	14	972	20
18	9	162	15	1458	21

The filing schemes based on combinatorial configurations derived from Theorem (3.2.1) have a relatively small redundancy R since R is necessarily less than or equal to b (where one recalls that the structure of such schemes allows the accession number of a pertinent record to be stored in exactly one address within any given bucket) . On the other hand, with k = v, the number of sub-buckets can become overwhelmingly large for large v. Fortunately, the effects of this problem can be substantially reduced by using appropriate multi-stage schemes which will be discussed in the next chapter.

3.2.2. Case II: s = q where q is a prime power.

Let θ denote a primitive element of the Galois field GF(q). Consider the (q + 1) x 2 matrix

$$H_2' = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & \theta \\ 1 & \theta^2 \\ \cdots & \cdots \\ 1 & \theta^{q-2} \end{bmatrix} \qquad (3.2.4)$$

which has the property that no two rows are linearly dependent. Let G_2 denote a (q + 1) x (q - 1) matrix, the columns of which are a basis of the vector space which is orthogonal to the columns of H_2'. For example,

$$G_2 = \begin{bmatrix} -1 & -\theta & -\theta^2 & \dots & -\theta^{q-2} \\ -1 & -1 & -1 & \dots & -1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Form the q^2 x (q + 1) matrix in which each of the rows is orthogonal to the columns of G_2 ; i.e., if $(x_1, x_2, \ldots, x_{q+1})$ denotes a row of the array, then the following equations are satisfied.

$$\begin{aligned}
x_3 &= x_1 + x_2 \\
x_4 &= \theta \cdot x_1 + x_2 \\
x_5 &= \theta^2 \cdot x_1 + x_2 \\
&\cdots \\
x_{q+1} &= \theta^{q-2} x_1 + x_2
\end{aligned}$$

Hence, if all q^2 possible **pairs** of values are assigned to (x_1, x_2) in the first two columns of the matrix, then the remaining columns can be determined from the above equations. Since every $(q + 1) \times 1$ vector generated by the columns of G_2 has at least three non-zero co-ordinates (for otherwise, there would be two dependent rows of H_2 and hence a contradiction), all equations which are linear combinations of the defining equations for the array involve at least three co-ordinates . As a result, any pair of co-ordinates is free to assume all q^2 possible pairs of values. Thus the constructed $q^2 \times (q + 1)$ matrix represents an orthogonal array $(q^2, q+1, q, 2)$ of index unity. It appears as follows

$$\left[\begin{array}{cc|ccc|}
0 & 0 & 0 & \ldots & 0 \\
0 & 1 & 1 & \ldots & 1 \\
0 & \theta & \theta & \ldots & \theta \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \theta^{q-2} & \theta^{q-2} & \ldots & \theta^{q-2} \\
\hline
1 & 0 & & & \\
1 & 1 & & & \\
1 & \theta & & & \\
\cdots & \cdots & & & \\
1 & \theta^{q-2} & & \text{Remainder} & \\
\theta & 0 & & \text{determined} & \\
\theta & 1 & & \text{from} & \\
\theta & \theta & & \text{arithmetic} & \\
\cdots & \cdots & & \text{of GF(q)} & \\
\theta & \theta^{q-2} & & & \\
\theta^{q-2} & 0 & & & \\
\theta^{q-2} & 1 & & & \\
\theta^{q-2} & \theta & & & \\
\cdots & \cdots & & & \\
\theta^{q-2} & \theta^{q-2} & & & \\
\hline
\end{array}\right] \qquad (3.2.5)$$

The previously indicated construction of the array is based on methods given by Bose [5] in connection with the design of factorial experiments. In addition, use is made of the fact that when q is a prime power, there exists a complete set of (q - 1) orthogonal Latin squares. The relationship between complete sets of Latin squares and orthogonal arrays has been considered by Bose and Bush [10].

If the first column of the matrix in (3.2.5) is deleted, we obtain a $(q^2, q, q, 2)$ orthogonal array in which the first q rows are a vector of 0's, a vector of 1's, a vector of θ's, ..., and a vector of θ^{q-2}'s respectively. Let this array be denoted by ($3.2.5^*$). The arrays (3.2.5) or ($3.2.5^*$) or the arrays obtained by deleting additional columns from (3.2.5) represent efficient configurations for the cases $v \leq q + 1$.

The designs (3.2.5) and ($3.2.5^*$) may be extended to the cases $v = q\ (q + 1)$ and $v = (q + 1)^2$ by the following composition. The first q^2 blocks of the arrays are formed by dividing the attributes into (q + 1) groups and assigning the i-th column of (3.2.5) to each of the attributes (columns) in the i-th group. As a result, any pair of values for attributes from different groups is covered in one of these blocks. Also, the pairs of values 00, 11, $\theta\theta$, ... $\theta^{q-2}\theta^{q-2}$ are covered for pairs of attributes belonging to the same group. The remaining blocks are then formed by duplicating within each group either

i. all but the first row of (3.2.5) for the case $v = (q + 1)^2$, i.e., all vectors there except the vector of 0's.

ii. all but the first q rows of ($3.2.5^*$) for the case $v = q\ (q + 1)$, i.e., all vectors there except the vectors of 0's, of 1's, of θ's, ..., and of θ^{q-2}'s.

When this is done, the number of blocks $b_q(v; 2)$ required for the resulting configurations are

$$b_q(\,(q+1)^2,\ 2) = q^2 + (q^2 - 1) = 2q^2 - 1 \quad \text{for (i)}$$
$$b_q(\,q(q+1),\ 2) = q^2 + (q^2 - q) = 2q^2 - q \quad \text{for (ii)}$$

The fact that the composition procedure can be extended for higher values of v is indicated in the following theorem.

<u>Theorem (3.2.2)</u>. For the case of v attributes with q levels each where q is a prime power, there exists a second order combinatorial configuration with $k = v$ and $b = (u_1 + u_2)\, q^2 - (qu_2 + u_1 - 1)$ where u_1 and u_2 are integers such that $(q + 1)^{u_1} q^{u_2}$ is as small as possible but still exceeds v.

<u>Proof</u>: As indicated in the proof of Theorem (3.2.1), only the case $v = (q + 1)^{u_1} q^{u_2}$ need be considered. The first group of blocks is formed by dividing the attributes into $(q + 1)^{u_1 - 1} q^{u_2}$ groups of $(q + 1)$ attributes each if $u_1 \geq 1$ or $(q + 1)^{u_1} q^{u_2 - 1}$ groups of q attributes each if $u_2 \geq 1$. For each attribute in the i-th group, assign the i-th column of the array appropriate to $v = (q + 1)^{u_1 - 1} q^{u_2}$ or $v = (q + 1)^{u_1} q^{u_2 - 1}$ as the case may be. The remaining blocks are then formed according to the previously indicated methods (i) or (ii) respectively depending on whether there are $(q + 1)$ or q attributes in each group. Either approach gives rise to $b_q(v; 2) = (u_1 + u_2)\, q^2 - (qu_2 + u_1 - 1)$ blocks.

As was the case with Theorem (3.2.1), the value of Theorem (3.2.2) is that it provides constructions for which b increases at a linear rate as v increases at an exponential rate. In particular, the following tables provide an indication of the relationship between b and v for different values of q.

q = 3		q = 4		q = 5	
v	b	v	b	v	b
3	9	4	16	5	25
4	9	5	16	6	25
12	15	20	28	30	45
16	17	25	31	36	49
36	21	80	40	150	65
48	23	100	43	180	69
64	25	125	46	216	73
108	27	320	52	750	85
144	29	400	55	900	89
192	31	500	58	1080	93
324	33	625	61	1296	97
432	35	1280	64	3750	105
576	37	1600	67	4500	109
972	39	2000	70	5400	113
1296	41	2500	73	6480	117
1728	43	5120	76	7776	121

q = 7		q = 9	
v	b	v	b
7	49	9	81
8	49	10	81
56	91	90	153
94	97	100	161
392	133	810	225
448	139	900	233
512	145	1000	247
2744	175	7290	297
3136	181	8100	305
3584	187	9000	313
4096	193	10000	321

Example (3.2.1). Suppose q = 3. The methods of Theorem (3.2.2) provide the following constructions

0	000
0	111
0	222
1	012
1	120
1	201
2	021
2	102
2	210

v=4, b=9

000	000	000	000
000	111	111	111
000	222	222	222
111	000	111	222
111	111	222	000
111	222	000	111
222	000	222	111
222	111	000	222
222	222	111	000
012	012	012	012
120	120	120	120
201	201	201	201
021	021	021	021
102	102	102	102
210	210	210	210

v=12, b=15

0000	0000	0000	0000
0000	1111	1111	1111
0000	2222	2222	2222
1111	0000	1111	2222
1111	1111	2222	0000
1111	2222	0000	1111
2222	0000	2222	1111
2222	1111	0000	2222
2222	2222	1111	0000
0111	0111	0111	0111
0222	0222	0222	0222
1012	1012	1012	1012
1120	1120	1120	1120
1201	1201	1201	1201
2021	2021	2021	2021
2102	2102	2102	2102
2210	2210	2210	2210

v=16, b=17

0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	1111	1111	1111	1111	1111	1111	1111	1111	1111
0000	0000	0000	2222	2222	2222	2222	2222	2222	2222	2222	2222
1111	1111	1111	0000	0000	0000	1111	1111	1111	2222	2222	2222
1111	1111	1111	1111	1111	1111	2222	2222	2222	0000	0000	0000
1111	1111	1111	2222	2222	2222	0000	0000	0000	1111	1111	1111
2222	2222	2222	0000	0000	0000	2222	2222	2222	1111	1111	1111
2222	2222	2222	1111	1111	1111	0000	0000	0000	2222	2222	2222
2222	2222	2222	2222	2222	2222	1111	1111	1111	0000	0000	0000
0000	1111	2222	0000	1111	2222	0000	1111	2222	0000	1111	2222
1111	2222	0000	1111	2222	0000	1111	2222	0000	1111	2222	0000
2222	0000	1111	2222	0000	1111	2222	0000	1111	2222	0000	1111
0000	2222	1111	0000	2222	1111	0000	2222	1111	0000	2222	1111
1111	0000	2222	1111	0000	2222	1111	0000	2222	1111	0000	2222
2222	1111	0000	2222	1111	0000	2222	1111	0000	2222	1111	0000
0111	0111	0111	0111	0111	0111	0111	0111	0111	0111	0111	0111
0222	0222	0222	0222	0222	0222	0222	0222	0222	0222	0222	0222
1012	1012	1012	1012	1012	1012	1012	1012	1012	1012	1012	1012
1120	1120	1120	1120	1120	1120	1120	1120	1120	1120	1120	1120
1201	1201	1201	1201	1201	1201	1201	1201	1201	1201	1201	1201
2021	2021	2021	2021	2021	2021	2021	2021	2021	2021	2021	2021
2102	2102	2102	2102	2102	2102	2102	2102	2102	2102	2102	2102
2210	2210	2210	2210	2210	2210	2210	2210	2210	2210	2210	2210

v=48, b=23

3.2.3. Case III: s is not a prime power.

Let q denote the smallest prime power exceeding s. Consider the series of configurations developed by the methods of Sub-section 3.2.2 for the situation of v attributes with q levels each. Suppose the q levels are denoted by $0, 1, \theta, \theta^2, \ldots, \theta^{s-2}, \theta^{s-1}, \ldots, \theta^{q-2}$. If $\theta^{s-1}, \theta^{s}, \ldots, \theta^{q-2}$ are replaced by $\theta^{s-2}, \theta^{s-3}, \ldots, \theta^{(s-2)-(q-s)}$, then the resulting array is equivalent to a second order configuration involving s levels. Moreover, in some cases it may be possible to reduce the number of blocks required by deleting any rows of the matrix which are made identical with some other row by the preceding transformation. Finally, one should note that for certain small values of v, other methods of constructing second order configurations may lead to a smaller number of blocks. In these cases, such constructions may be used to supplement the composition procedure in a fashion similar to the method outlined in the proof of Theorem (3.2.2).

To see more clearly, however, the basic approach of this sub-section, let us consider as an example the situation in which $s = 6$. In this case, $q = 7$. For $v = 7$ or 8, there exists a second order configuration with $b = 49$ blocks. If θ^5 is replaced by θ^4 in this, then two of the blocks have the same form and hence one may be deleted. As a result, for $v = 7$ or 8 and $s = 6$, there exists a second order combinatorial configuration with $b = 48$. Using the structure of these arrays according to Theorem (3.2.2), the following table may be formed to indicate the relationship between v and b.

v	b
7	48
8	48
56	90
64	95
392	132
448	137
512	142
2744	174
3136	179
3584	184
4096	189

The above table may be supplemented by noting that for v = 3, a configuration with b = 36 blocks may be formed by assigning the 36 possible ordered pairs to the first two columns of the array and then forming the third column as the mod 6 sum of the first two columns. In addition, for v = 4 and v = 5, arrays may be based on constructions appropriate for q = 4. This is achieved by first forming the relevant array with q = 4 three times, and then identifying the symbols in the first array with the levels 0, 1, 2, 3; the ones in the second array with 0, 1, 4, 5; and the ones in the third array with 2, 3, 4, 5; and finally deleting any block , so obtained, that is identical to some other block or which are redundant in the sense that pairs covered by it are covered elsewhere. When v = 4, this approach leads to a configuration with b = 42 while for v = 5, b = 44. These may be used to supplement the preceding table relating v and b as follows.

v	b	v	b
3	36	128	120
4	42	168	125
5	44	224	126
12	77	256	131
16	78	672	161
24	83	896	162
32	84	1176	167
96	119	1568	168

For other non-prime power values of s, similar methods and tables can be developed. However, as indicated at the onset, the simplest approach is to work with arrays based on the smallest prime power q exceeding s.

3.2.4. Case IV: the n_i are not necessarily equal.

Let q denote the smallest prime power exceeding each of the n_i's $i = 1, 2, \ldots, v$ where n_i denotes the number of levels associated with the i-th attribute. Any second order configuration which is appropriate to the case of q levels can be applied to this situation by defining for each attribute a correspondence which transforms the symbols $0, 1, \theta, \ldots, \theta^{q-2}$ into the n_i levels of the attribute. In addition, in some cases, it may be possible to reduce the number of blocks by deleting any rows of the array which are made identical with some other row by the preceding transformation. Finally, as was indicated for the situation of Sub-section 3.2.3, one can attempt to construct appropriate second order configurations for small values of v and then expand them by a composition procedure.

To illustrate the approach here, let us consider a situation in which one half of the attributes have two levels each and the other half have three each. For the case v = 6, the following configuration is appropriate

$$\begin{array}{cc}
011 & 000 \\
011 & 111 \\
011 & 222 \\
101 & 012 \\
101 & 120 \\
101 & 201 \\
110 & 021 \\
110 & 102 \\
000 & 210
\end{array} \quad , \quad b = 9$$

where each of the first three attributes has two levels while each of the last three has three levels. This may then be extended to the case v = 18 as follows.

$$\begin{array}{ccc@{\quad}ccc}
000 & 111 & 111 & 000 & 000 & 000 \\
000 & 111 & 111 & 111 & 111 & 111 \\
000 & 111 & 111 & 222 & 222 & 222 \\
111 & 000 & 111 & 000 & 111 & 222 \\
111 & 000 & 111 & 111 & 222 & 000 \\
111 & 000 & 111 & 222 & 000 & 111 \\
111 & 111 & 000 & 000 & 222 & 111 \\
111 & 111 & 000 & 111 & 000 & 222 \\
000 & 000 & 000 & 222 & 111 & 000 \\
\hline
011 & 011 & 011 & 012 & 012 & 012 \\
011 & 011 & 011 & 120 & 120 & 120 \\
101 & 101 & 101 & 201 & 201 & 201 \\
101 & 101 & 101 & 021 & 021 & 021 \\
110 & 110 & 110 & 102 & 102 & 102 \\
110 & 110 & 110 & 210 & 210 & 210
\end{array} \quad , \; b = 15$$

where each of the first nine attributes has two levels while each of the last nine has three levels. From the above, one can see that the composition procedure can be readily continued to yield configurations for higher values of v.

3.3. The construction of configurations of order 3 with k = v.

As was the case in the previous section, we shall here be mostly concerned with the situation in which each of v attributes has s levels. Methods will be given for the construction of a (b x v) matrix such that among the rows of each 3-column sub-matrix, each of the s^3 possible ordered three-tuples occurs at least once. The basic composition approach will be discussed for the following three cases of interest:

Case I : s = 2

Case II : s = q where q is a prime power

Case III: s is not a prime power

Some aspects of the solutions for the above cases may also be applied to the situation in which the numbers n_i are not necessarily equal. However, the basic approach to this situation is not as clear-cut here as it was for second order configurations In any event, some remarks as to how one should proceed will be indicated under the heading of Case IV.

3.3.1. Case I: $s = 2$.

When $v = 4$, an orthogonal array of strength three and index unity may be constructed by forming an (8×4) matrix in which the first three columns represent all possible ordered three-tuples and the fourth column is the mod 2 sum of the first three columns. The resulting array appears as follows

$$\begin{matrix} 0000 \\ 0011 \\ 0101 \\ 0110 \\ 1001 \\ 1010 \\ 1100 \\ 1111 \end{matrix} \quad , \quad b = 8 \qquad (3.3.1)$$

It represents an optimal configuration of order three. Moreover, the first three columns of (3.3.1) is optimal for $v = 3$.

Efficient partially balanced arrays may be derived by noting that for any ordered choice of three attributes, the set of v-tuples which contain exactly one "1" provide a cover for the three-tuples 000, 001, 010, and 100 while the set of v-tuples which contain exactly one "0" provide a cover for the three-tuples 011, 101, 110, and 111. The number of blocks required in such arrays is hence $b = 2v$. In particular, for $v = 5$, we have the array

$$\begin{array}{c} 10000 \\ 01000 \\ 00100 \\ 00010 \\ 00001 \\ 01111 \\ 10111 \\ 11011 \\ 11101 \\ 11110 \end{array} , \quad b = 10 \qquad (3.3.2)$$

in which $\lambda(1, 0, 0) = \lambda(1, 1, 0) = 1$ and $\lambda(0, 0, 0) = \lambda(1, 1, 1) = 2$. For general values of v, $\lambda(1, 0, 0) = \lambda(1, 1, 0) = 1$ also, but $\lambda(0, 0, 0) = \lambda(1, 1, 1) = (v - 3)$.

The previously indicated constructions for v = 3, 4 and v = 5 may be extended to v = 6, v = 8, and v = 10 by the following composition procedure. Let the v attributes be divided into $v^* = (v/2)$ consecutive pairs. The array of order three which is appropriate to v^* attributes with two levels each is then written down with the adjustment that "0" is replaced by "00" and "1" is replaced by "11". For any choice of attributes coming from different pairs, all possible ordered three-tuples are covered because such a choice is equivalent to a selection of three attributes from the v^* for which a third order configuration already exists. To complete the construction, additional rows need to be added to the array so that for any choice of three attributes with two coming from the same pair, all possible three-tuples are covered. For the cases where the levels associated with the attributes coming from the same pair are equal (i.e , "00" or "11"), the initial set of blocks already provides a cover because the resulting three-tuples are equivalent to two-tuples of v^* attributes and hence are accounted for by the corresponding third order array. The remaining uncovered three-tuples involve either "01" or "10" being associated with a pair. These may be taken care of by writing down the array of order two which is appropriate to v^* attributes with two levels each with the adjustment that "0" is replaced by "01" and "1" is replaced by "10". Because any choice of three attributes with two coming from the same pair and having different levels is equivalent to a corresponding two-tuple of v^* attributes, the additional blocks complete the construction of the configuration of order three.

In order to see more clearly the structure associated with the previously given method of composition, let us consider the following examples for $v = 6$, $v = 8$, and $v = 10$.

```
00 00 00        00 00 00 00        11 00 00 00 00
00 00 11        00 00 11 11        00 11 00 00 00
00 11 00        00 11 00 11        00 00 11 00 00
00 11 11        00 11 11 00        00 00 00 11 00
11 00 00        11 00 00 11        00 00 00 00 11
11 00 11        11 00 11 00        00 11 11 11 11
11 11 00        11 11 00 00        11 00 11 11 11
11 11 11        11 11 11 11        11 11 00 11 11
--------        -----------        11 11 11 00 11
01 01 01        01 01 01 01        11 11 11 11 00
01 10 10        01 10 10 10        --------------
10 01 10        10 01 10 10        01 01 01 01 01
10 10 01        10 10 01 10        01 10 10 10 10
                10 10 10 01        10 01 10 10 10
                                   10 10 01 10 10
                                   10 10 10 01 10
                                   10 10 10 10 01

 b = 12           b = 13            b = 16
```

The above constructions can be readily extended to $v = 12$, $v = 16$, and $v = 20$, etc by re-applying the composition algorithm. In summary, if we let $b_2(v; 3)$ denote the number of blocks so obtained for the configuration of order three with $k = v$ and appropriate to the situation of v attributes with two levels each, then we may state the following theorem.

Theorem (3.3.1): By successively applying the previously indicated method of composition, combinatorial configurations of order three with k = v may be constructed for any v attributes with two levels each. The number of blocks required is given by the relation $b_2(v; 3) = b_2([v/2]_+; 3) + b_2([v/2]_+; 2)$ where it is presumed that v exceeds 6.

Using Theorem (3.3.1), the following table may be constructed to indicate the relationship between v and b

v	b	v	b	v	b
3	8	24	26	192	62
4	8	32	29	256	68
5	10	40	34	320	74
6	12	48	36	384	78
8	13	64	40	512	85
10	16	80	46	640	91
12	18	96	48	768	96
16	20	128	53	1024	104
20	24	160	59	1280	110

From the above, one can observe that the required b increases at a much slower rate than v.

3.3.2. Case II: s = q where q is a prime power.

Using properties of non-degenerate conics in PG(2, q) according to methods given in Bose [5] and Bose and Bush [10], one can construct orthogonal arrays of strength three and index unity for $3 \leq v \leq q + 1$ if q is odd and $3 \leq v \leq q + 2$ if v is even. The basic approach is to form the matrix H_3 with

$$H_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 1 & \theta & \theta^2 & \dots & \theta^{q-2} \\ 0 & 1 & 1 & \theta^2 & \theta^4 & \dots & \theta^{2q-4} \end{bmatrix} \quad \text{if q is odd}$$

$$H_3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & 1 & \theta & \theta^2 & \cdots & \theta^{q-2} \\ 0 & 0 & 1 & 1 & \theta^2 & \theta^4 & \cdots & \theta^{2q-4} \end{bmatrix} \quad \text{if } q \text{ is even}$$

where θ is a primitive element of GF(q). The matrix H_3 has the property that no three columns are linearly dependent. The desired array may be formed by writing down all q^3 possible linear combinations of the rows of H_3 with respect to GF(q). This process may be compactly written as the matrix product ΞH_3 where Ξ is a $(q^3 \times 3)$ matrix, the rows of which are all possible three-tuples occurring once.

An alternative mechanical means of obtaining the desired array from H_3 is to form the matrix G_3 whose rows are orthogonal to the rows of H_3 and then to identify the rows of G_3 with homogeneous linear equations in variates corresponding to the columns of the array. The other aspects of this approach are similar to what was outlined in Sub-section 3.3.2.

The orthogonal arrays previously described represent optimal configurations of order three. From a general point of view, they will be the primary starting point for the composition procedure. However, in some particular cases, it may also be useful to similarly extend certain partially balanced arrays. This aspect will be discussed later.

An array of order three appropriate to v^* attributes with q levels each may be extended to $v = cv^*$ where $2 \leq c \leq q$ as follows. Let the v attributes be divided into v^* consecutive c-plets. The array of order three which is appropriate to v^* attributes is then written down with the adjustment that each symbol is replaced by a c-plet in which the symbol is repeated c times; eg "0" is replaced by "00...0"; "1" is replaced by "11...1"; "θ" is repaced by "$\theta\theta...\theta$"; etc. For any choice of three attributes coming from different c-plets, all possible ordered three-tuples are covered because such a choice is equivalent to a selection of three attribute levels from the v^* for which the third order configuration already exists. To complete the configuration, additional rows need to be added to the array so that for any choice of three attributes with two or three coming from the same c-plet, all possible three-tuples are covered. For the cases in which the levels associated with the same c-plet are equal (eg., "00" or "11"; or "000" or "111"; etc.), the initial set of blocks already provides a cover because the resulting three-tuples are equivalent to two-tuples or one-tuples of v^* attributes and hence are covered by the corresponding third order array. The remaining uncovered three-tuples are accounted for in two steps. First $(q - 1)$ arrays of order two which are appropriate to v^* attributes with q levels each are formed with the adjustment that each symbol therein is replaced by a corresponding c-plet. The q c-plets for the first array may be taken as the rows of the sub-matrix associated with the second through $(c + 1)$-th columns and the $(q + 1)$-th through $(2q)$-th rows of the orthogonal array of strength two given in (3.2.5). The q c-plets for the next such array may be formed similarly from the $(2q + 1)$-th through $(3q)$-th rows of (3.2.5). This process can be continued with the q c-plets in the $(q - 1)$-th such array being so taken from the last q rows of (3.2.5). For the case of $c = q$, the above process is equivalent to associating the $(q - 1)$ sets of q c-plets with the rows of $(q - 1)$ mutually orthogonal $q \times q$ Latin squares. These additional arrays account for the remaining three-tuples in which exactly two attributes come from the same c-plet. The final part of the array is formed by repeating for each of the v^* groups corresponding to c-plets, a c-column matrix the rows of which (when taken together with the c-plets associated with the (q - 1) second order arrays and the c-plets with the same symbol repeated) lead to a configuration of order three appropriate for c attributes with q levels each. Since $c \leq q$, this can be done with $(q^3 - q^2)$ additional blocks by locating the $(q^3 \times c)$ sub-matrix of ΞH_3 which represents an orthogonal array of strength three which contains the orthogonal array of strength two associated with the q^2 different types of c-plets already appearing in the array.

The structure associated with the previously given method of composition can be seen more clearly in terms of a series of examples. Hence, let us consider the case q = 3. When v = 4, the following orthogonal array of strength three and index unity may be formed.

$$\begin{array}{ccc} 0000 & 1002 & 2001 \\ 0012 & 1011 & 2010 \\ 0021 & 1020 & 2022 \\ 0102 & 1101 & 2100 \\ 0111 & 1110 & 2112 \\ 0120 & 1122 & 2121 \\ 0201 & 1200 & 2202 \\ 0210 & 1212 & 2211 \\ 0222 & 1221 & 2220 \end{array} , \quad b = 27; \qquad (3.3.3)$$

it represents an optimal configuration of order three. Moreover, the first three columns of the matrix in (3.3.3) represents an optimal array for v = 3. Finally, the following partially balanced array is of interest for v = 5

$$\begin{array}{cccc} 00012 & 00021 & 01111 & 21111 \\ 00102 & 00201 & 10111 & 12111 \\ 00120 & 00210 & 11011 & 11211 \\ 01002 & 02001 & 11101 & 11121 \\ 01020 & 02010 & 11110 & 11112 \\ 01200 & 02100 & 02222 & 12222 \\ 10002 & 20001 & 20222 & 21222 \\ 10020 & 20010 & 22022 & 22122 \\ 10200 & 20100 & 22202 & 22212 \\ 12000 & 21000 & 22220 & 22221 \end{array} , \quad b = 40 \qquad (3.3.4)$$

where $\lambda(0, 1, 2) = \lambda(0, 1, 1) = \lambda(0, 2, 2) = \lambda(1, 2, 2) = \lambda(1, 1, 2) = 1$; $\lambda(0, 0, 1) = \lambda(0, 0, 2) = \lambda(0, 0, 0) = 2$; $\lambda(1, 1, 1) = \lambda(2, 2, 2) = 4$.

The array given in (3.3.3) can be extended to $v = 8$ with $b = 45$ as follows

00 00 00 00	11 00 00 22	22 00 00 11	01 01 01 01	02 02 02 02
00 00 11 22	11 00 11 11	22 00 11 00	01 12 12 12	02 10 10 10
00 00 22 11	11 00 22 00	22 00 22 22	01 20 20 20	02 21 21 21
00 11 00 22	11 11 00 11	22 11 00 00	12 01 12 20	10 02 10 21
00 11 11 11	11 11 11 00	22 11 11 22	12 12 20 01	10 10 21 02
00 11 22 00	11 11 22 22	22 11 22 11	12 20 01 12	10 21 02 10
00 22 00 11	11 22 00 00	22 22 00 22	20 01 20 12	21 02 21 10
00 22 11 00	11 22 11 22	22 22 11 11	20 12 01 20	21 10 02 21
00 22 22 22	11 22 22 11	22 22 22 00	20 20 12 01	21 21 10 02

Similarly, it may be extended to $v = 12$ and $b = 63$

000 000 000 000	111 000 000 222	222 000 000 111
000 000 111 222	111 000 111 111	222 000 111 000
000 000 222 111	111 000 222 000	222 000 222 222
000 111 000 222	111 111 000 111	222 111 000 000
000 111 111 111	111 111 111 000	222 111 111 222
000 111 222 000	111 111 222 222	222 111 222 111
000 222 000 111	111 222 000 000	222 222 000 222
000 222 111 000	111 222 111 222	222 222 111 111
000 222 222 222	111 222 222 111	222 222 222 000

012 012 012 012	021 021 021 021	001 001 001 001	002 002 002 002
012 120 120 120	021 102 102 102	010 010 010 010	011 011 011 011
012 201 201 201	021 210 210 210	022 022 022 022	020 020 020 020
120 012 120 201	102 021 102 210	100 100 100 100	101 101 101 101
120 120 201 012	102 102 210 021	112 112 112 112	110 110 110 110
120 201 012 120	102 210 021 102	121 121 121 121	122 122 122 122
201 012 201 120	210 021 210 102	202 202 202 202	200 200 200 200
201 120 012 201	210 102 021 210	211 211 211 211	212 212 212 212
201 201 120 012	210 210 102 021	220 220 220 220	221 221 221 221

By continuing to extend smaller arrays to larger ones by the above illustrated method of composition, one can form the following table to indicate the relationship between v and b.

v	b	v	b
3	27	144 = 3 x 48	191
4	27	192 = 3 x 64	219
5	40	216 = 3 x 72	225
6	45	288 = 3 x 96	241
8 = 2 x 4	45	324 = 3 x 108	243
12 = 3 x 4	63	432 = 3 x 144	267
16 = 2 x 8	75	576 = 3 x 192	299
24 = 3 x 8	93	648 = 3 x 216	309
32 = 2 x 16	109	864 = 3 x 288	325
36 = 3 x 12	111	972 = 3 x 324	327
48 = 3 x 16	127	1296 = 3 x 432	355
64 = 2 x 32	151	1728 = 3 x 576	391
72 = 3 x 24	153	1944 = 3 x 648	405
96 = 3 x 32	169	2592 = 3 x 864	421
108 = 3 x 36	171	2916 = 3 x 972	423

From the above table, one can observe that except for the first few cases, b increases at a substantially slower rate than v.

The indicated method of construction can be similarly applied to other prime power values of q. In summary, if we let $b_q(v; 3)$ denote the number of blocks obtained for the configuration of order three with k = v and appropriate to a situation of v attributes with q levels each, then we may state the following theorem.

Theorem (3.3.2). By successively applying the previously described method of composition, combinatorial configurations of order three with k = v may be constructed for any v attributes with q levels each where q is a prime power. For $v \geq 2q$, the number of blocks required is given by $b_q(v; 3) = b_q([v/c]_+; 3) + (q - 1)\, b_q([v/c]_+; 2) + b_q^*(c; 3)$ where $b_q^*(2; 3) = 0$ and $b_q^*(c; q) = (q^3 - q^2)$ for $3 \leq c \leq q$.

For any prime power q, the relationship between v and b can be developed by applying Theorem (3.3.2). In doing this, the c value used in the successive steps of the composition procedure must be appropriately chosen. Some aspects of this process have already been illustrated for the case q = 3. As another example, let us consider the following table indicating the relationship between v and b for q = 4

v	b
4	64
5	64
6	64
12 = 2 x 6	112
20 = 4 x 5	160
24 = 4 x 6	196
48 = 4 x 12	244
80 = 4 x 20	292
96 = 4 x 24	337
192 = 4 x 48	412
320 = 4 x 80	460
384 = 4 x 96	514
768 = 4 x 192	616
1280 = 4 x 320	664
1536 = 4 x 384	727

3.2.3. Case III: s is not a prime power.

As was pointed out in Sub-section 3.2.3, the easiest way to construct configurations for this situation is to form the array appropriate to q levels where q is the smallest prime power exceeding s and then to apply a transformation in which one or more of the q symbols correspond(s) to exactly one of the s levels. This approach will lead to a reasonably satisfactory series of designs, particularly if some blocks can be deleted because of duplications induced by the transformation.

Alternatively, a composition method similar to the one of the previous section can be developed. In particular, the procedure when c = 2 is quite straightforward and involves the use of double symbols like "00", "11", . . . , "ss" in the first series of blocks which are associated with a known third order configuration; and the series of symbols

Series 1: 01, 12, . . . , (s-1)s

Series 2: 02, 13, . . . , (s-1)1

. . . .

Series s-1: 0(s-1), 10, . . . , (s-1)(s-2)

in the (s-1) sets of blocks which are associated with the relevant second order configurations. In some instances, the value of c can be readily increased as high as one plus the number of mutually orthogonal Latin squares of side s. When this is done, however, additional blocks have to be added to account for the three-tuples arising from the same c-plet. Finally, some efficiencies may be introduced by using systems like partially balanced arrays either with respect to initial constructions or in the formation of the c-plets. However, these questions necessarily require investigations of each case separately and hence will not be considered here.

3.3.4 Case IV: the n_i are not necessarily equal.

Again, the most direct method is to construct the array which is appropriate to q levels where q is the smallest prime power that exceeds all the n_i. The configuration can then be formed by applying to each attribute a transformation in which one or more of the q symbols corresponds to exactly one of the n_i levels. The other details of this type of approach are similar to what has been outlined in Sub-section 3.3.3.

In some cases, however, some efficiencies directed at reducing the number of blocks required can be introduced. This involves devoloping suitable arrays for small v and then expanding them by an appropriate composition procedure. However, since the method of attack depends to a large extent on the actual values of the n_i, general methods here are very difficult to develop. On the other hand, the basic principles can be seen in terms of an example. Hence, let us consider the case when each of the n_i is either two or three.

Suppose there are six attributes in which the first three have two levels each and the last three have three levels each. A suitable configuration is given by

011	000	100	001	010	002
011	111	100	112	010	110
011	222	100	220	010	221
011	012	101	010	110	011
011	120	101	121	110	122
011	201	101	202	110	200
001	021	100	022	110	020
001	102	100	100	110	101
001	210	100	211	110	212
000	000	111	111		

, $b = 29$

This may be expanded to a situation involving twelve attributes in which the first six have two levels each while the last six have three levels each

00	11	11	00	00	00	11	00	00	00	00	11	00	11	00	00	00	22
00	11	11	11	11	11	11	00	00	11	11	22	00	11	00	11	11	00
00	11	11	22	22	22	11	00	00	22	22	00	00	11	00	22	22	11
00	11	11	00	11	22	11	00	11	00	11	00	11	11	00	00	11	11
00	11	11	11	22	00	11	00	11	11	22	11	11	11	00	11	22	22
00	11	11	22	00	11	11	00	11	22	00	22	11	11	00	22	00	00
00	00	11	00	22	11	11	00	00	00	22	22	11	11	00	00	22	00
00	00	11	11	00	22	11	00	00	11	00	00	11	11	00	11	00	11
00	00	11	22	11	00	11	00	00	22	11	11	11	11	00	22	11	22
00	00	00	00	00	00	11	11	11	11	11	11						

01	10	10	01	01	01	01	10	10	02	02	02
01	10	10	12	12	12	01	10	10	10	10	10
01	10	10	20	20	20	01	10	10	21	21	21
10	01	10	01	12	20	10	01	10	02	10	21
10	01	10	12	20	01	10	01	10	10	21	02
10	01	10	20	01	12	10	01	10	21	02	10
10	10	01	01	20	12	10	10	01	02	21	10
10	10	01	12	01	20	10	10	01	10	02	21
01	01	01	20	12	01	01	01	01	21	10	02

; $b = 47$

The previously described type of procedure can be continued for higher values of v. Also, some mofifications can be introduced to base part of the composition procedure on three-plets. In any event, the result of the construction procedure is to cause the number of required blocks b to increase at a reasonably slow rate as v increases at a relatively rapid rate.

3.4. The construction of configurations of order 4 with k = v and s = 2.

In this section, we shall consider a method of constructing a (b x v) matrix such that among the rows of each four-column sub-matrix each of the 2^4 possible ordered four-tuples occurs at least once. When v = 5, an orthogonal array of strength four and index unity may be formed with b = 16. It has the following appearance

```
00000     10001
00011     10010
00101     10100
00110     10111         (3.4.1)
01001     11000
01010     11011
01100     11101
01111     11110
```

and represents an optimal configuration of order four. In addition, the first four columns of the array in (3.4.1) is optimal for v = 4.

For higher values of v, appropriate partially balanced arrays may be derived by noting that for any ordered choice of four attributes, the set of v-tuples which contain exactly two "1's" provide a cover for the four-tuples 0000, 0001, 0010, 0100, 1000, 0011, 0101, 0110, 1001, 1010, 1100 while the set of v-tuples which contain exactly one "0" provide a cover for the four-tuples 0111, 1011, 1101, 1110, 1111. The number of blocks required in these constructions is given by

$b = \binom{v}{2} + \binom{v}{1} = v\,(v + 1)/\,2$. In particular, for $v = 6$ and $v = 7$, they appear as follows

$$\begin{array}{l} 110000 \\ 101000 \\ 100100 \\ 100010 \\ 100001 \\ 011000 \\ 010100 \\ 010010 \\ 010001 \\ 001100 \\ 001010 \\ 001001 \\ 000110 \\ 000101 \\ 000011 \\ 011111 \\ 101111 \\ 110111 \\ 111011 \\ 111101 \\ 111110 \end{array}, \quad b = 21 \qquad \begin{array}{l} 1100000 \\ 1010000 \\ 1001000 \\ 1000100 \\ 1000010 \\ 1000001 \\ 0110000 \\ 0101000 \\ 0100100 \\ 0100010 \\ 0100001 \\ 0011000 \\ 0010100 \\ 0010010 \\ 0010001 \\ 0001100 \\ 0001010 \\ 0001001 \\ 0000110 \\ 0000101 \\ 0000011 \\ 0111111 \\ 1011111 \\ 1101111 \\ 1110111 \\ 1111011 \\ 1111101 \\ 1111110 \end{array}, \quad b = 28 \tag{3.4.2}$$

in which $\lambda(1, 1, 0, 0) = \lambda(1, 1, 1, 0) = 1$, $\lambda(1, 0, 0, 0) = v - 4$, $\lambda(1, 1, 1, 1) = v - 4$, and $\lambda(0, 0, 0, 0) = \binom{v-4}{2}$.

The constructions in (3.4.1) and (3.4.2) may be extended to $v = 8$, $v = 10$, $v = 12$, and $v = 14$ by the following composition procedure. Let the v attributes be divided into $v^* = (v/2)$ consecutive pairs. The array of order four which is appropriate to v^* attributes with two levels each is then written down with the adjustment that "0" is replaced by "00" and "1" is replaced by "11". For any choice of four attributes coming from different pairs, all possible ordered four-tuples are covered because such a choice is equivalent to a selection of four attributes from the v^* for which a fourth order configuration already exists. To complete the construction, additional rows need to be added to the array so that for choices of four attributes with two coming from the same pair, all possible four-tuples are accounted for. In the cases where the levels associated with the attributes coming from the same pair are equal (i.e., "00" or "11"), the initial set of blocks already are suitable because the resulting four-tuples are equivalent to either three-tuples or two-tuples of v^* attributes and hence are covered by the corresponding fourth order array. The remaining uncovered four-tuples involve either "01" or "10" being associated with a pair. These may be taken care of in two steps. First a set of blocks is added which corresponds to an array of order three in v^* attributes with two levels each but with the adjustment that "0" is replaced by "01" and "1" is replaced by "10". Because any choice of four attributes with two coming from the same pair and having different levels while the other two come from different pairs is equivalent to a corresponding three-tuple of v^* attributes, these additional blocks cover such four-tuples. The construction is completed by adding a set of blocks which is associated with an array of order two in v^* attributes with four levels each but with the adjustment that "0" is replaced by "00", "1" is replaced by "01", "θ" is replaced by "10", and "θ^2" is replaced by "11". This final part takes care of all four-tuples in which two attributes come from each of two pairs. Some reduction in the total number of blocks required for the array may be realized at this stage by deleting any blocks in which the corresponding covered four-tuples have already been accounted for in previous blocks;

eg., the blocks associated with a vector of 0's, a vector of 1's, a vector of θ's and a vector of θ^2's may be deleted from this last part.

In order to see more clearly the structure associated with the previously described method of composition, let us consider the following examples for v = 8 and v = 10.

```
00 00 00 00     01 01 01 01     00 01 10 11
00 00 00 11     01 01 10 10     01 00 11 10
00 00 11 00     01 10 01 10     10 11 00 01
00 00 11 11     01 10 10 01     11 10 01 00
00 11 00 00     10 01 01 10     00 10 11 01
00 11 00 11     10 01 10 01     01 11 10 00
00 11 11 00     10 10 01 01     10 00 01 11
00 11 11 11     10 10 10 10     11 01 00 10
11 00 00 00                     00 11 01 10   ,   b = 36
11 00 00 11                     01 10 00 11
11 00 11 00                     10 01 11 00
11 00 11 11                     11 00 10 01
11 11 00 00
11 11 00 11
11 11 11 00
11 11 11 11
```

```
00 00 00 00 00     10 01 01 01 01     00 01 01 01 01
00 00 00 11 11     01 10 01 01 01     00 10 10 10 10
00 00 11 00 11     01 01 10 01 01     01 00 01 10 11
00 00 11 11 00     01 01 01 10 01     01 01 00 11 10
00 11 00 00 11     01 01 01 01 10     01 10 11 00 01
00 11 00 11 00     01 10 10 10 10     01 11 10 01 00
00 11 11 00 00     10 01 10 10 10     10 00 10 11 01
00 11 11 11 11     10 10 01 10 10     10 01 11 10 00
11 00 00 00 11     10 10 10 01 10     10 10 00 01 11   ,   b = 40
11 00 00 11 00     10 10 10 10 01     10 11 01 00 10
11 00 11 00 00                        11 00 11 01 10
11 00 11 11 11                        11 01 10 00 11
11 11 00 00 00                        11 10 01 11 00
11 11 00 11 11                        11 11 00 10 01
11 11 11 00 11
11 11 11 11 00
```

The indicated procedure can be continued for higher values of v. In summary, if we let $b_2(v; 4)$ denote the number of blocks obtained for the configuration of order four with k = v and appropriate to a situation of v attributes with two levels each, then we may state

Theorem (3.4.1). By successively applying the previously described method of composition, combinatorial configurations of order four with $k = v$ may be constructed for any v attributes with two levels each. For $v \geq 8$, the number of blocks required is given by the formula $b_2(v; 4) = b_2([v/2]_+; 4) + b_2([v/2]_+; 3) + b_4^*([v/2]_+; 2)$ where $b_4^*([v/2]_+;2)$ is an appropriate number which does not exceed $b_4([v/2]_+;2)$.

By applying Theorem (3.4.1), the following table may be formed to indicate the relationship between v and b.

v	b	v	b
4	16	80	200
5	16	96	224
6	21	112	250
7	28	128	258
8	36	160	284
10	40	192	313
12	55	224	347
14	65	256	359
16	73	320	391
20	80	384	423
24	97	448	463
28	109	512	475
32	117	640	515
40	130	768	554
48	152	896	604
56	174	1024	619
64	182	1280	666

From the above table, one can see that b increases at a reasonably slow rate as v increases. Indeed as v becomes quite large, the rate of increase of b becomes comparably much smaller.

Methods similar to the ones outlined in this section can also be developed for cases when s = q where q is a prime power as well as general s. In addition, the basic approach can be extended to fifth and higher order configurations. However, these topics will not be discussed here as they involve continued application of previous concepts.

CHAPTER IV

MULTI-STAGE FILING SYSTEMS

4.1 Multi-stage combinatorial configurations.

In the previous chapter, we have considered a general method of constructing combinatorial configurations with $k = v$. These systems have the desirable property that the number of blocks b is of reasonably small magnitude for large v. However, because $k = v$, the number of sub-buckets associated with each bucket in the corresponding filing system can become overwhelmingly large. For example, some buckets may have as many as $(2^v - 1)$ sub-buckets. As a result, the component of retrieval time which is specific to checking whether sub-buckets pertain to some given query may reach such a considerable magnitude as to destroy any value which the filing system might otherwise have. This particular problem was one of the factors which motivated Ray-Chaudhuri [43] to introduce the concept of a multi-stage combinatorial configuration as the basis of a multi-stage filing scheme.

A multi-stage combinatorial configuration $(\Omega, k, \mathcal{G}, b, d)$ consists of a master set Ω (representing the set of attribute levels $A_{11}, \ldots, A_{1n_1}, \ldots, A_{v1}, \ldots, A_{vn_v}$), a class of subsets $\mathcal{G}$, and blocks $B_{h_1h_2\ldots h_\eta}$ with $1 \leq \eta \leq d$ such that

i. $B_{h_1h_2\ldots h_\eta} \subseteq B_{h_1h_2\ldots h_{\eta-1}}$ where $h_\eta = 1, 2, \ldots, b_{h_1h_2\ldots h_{\eta-1}}$ for $2 \leq \eta \leq d$ and $h_1 = 1, 2, \ldots, b_0$.

ii. $\left| B_{h_1 h_2 \ldots h_\eta} \right| \leq k$.

iii. For every set A in $\mathfrak{A}$, there exists $(h_1, h_2, \ldots, h_d)$ such that A is contained in the block $B_{h_1 h_2 \ldots h_d}$.

The total number of blocks involved in all stages of the configuration is given by $b = b_0 + \Sigma\, b_{h_1 h_2 \ldots h_\eta}$. If $|A| \leq t$ for each A in $\mathfrak{A}$, then the configuration is said to be of order t and is denoted as an (Ω, k, t, b, d) scheme. The blocks of the configuration can be ordered by introducing the following rule. The η-tuple $(h_1, h_2, \ldots, h_\eta)$ is said to precede the η-tuple $(h_1', h_2', \ldots, h_\eta')$ if for some ξ where $1 \leq \xi \leq \eta$, we have $h_1 \leq h_1'$, $h_2 \leq h_2'$, $\ldots$, $h_{\xi-1} \leq h_{\xi-1}'$, $h_\xi < h_\xi'$. Using the above convention, we define the η-th stage covering index $\gamma_\eta(A)$ of any given set A in $\mathfrak{A}$ to be that η-tuple which precedes all other η-tuples for which the corresponding blocks cover A. In other words, $\gamma_\eta(A) = (h_1, h_2, \ldots, h_\eta)$ if and only if A is contained in $B_{h_1 h_2 \ldots h_\eta}$ but A is not contained in $B_{h_1'' h_2'' \ldots h_\eta''}$ for $h_1'' \leq h_1$, $h_2'' \leq h_2$, $\ldots$, $h_{\xi-1}'' \leq h_{\xi-1}$, $h_\xi'' < h_\xi$ where $1 \leq \xi \leq \eta$. So that some degree of consistency is maintained in the structure of the corresponding filing systems, the discussion here will be restricted to what has been termed **simple** multi-stage schemes. A multi-stage configuration is called simple if the η-th stage covering index of a set A in $\mathfrak{A}$ contains the $(\eta-1)$-th stage index in the sense that $\gamma_\eta(A) = (h_1, h_2, \ldots, h_{\eta-1}, h_\eta)$ while $\gamma_{\eta-1}(A) = (h_1, h_2, \ldots, h_{\eta-1})$. Using the above framework, the following theorem may be proven.

Theorem (4.1.1). Suppose there exist uni-stage configurations $(\Omega_{\eta-1}, k_\eta, t, b_{\eta-1})$ for $\eta = 1, 2, \quad, d$ where $\Omega_0 = \Omega$ and where $\Omega_{\eta-1}$

may be identified with the at most $k_{\eta-1}$ attribute levels which are assigned to any given block in the $(\eta-1)$-th stage. Then there exists a simple multi-stage configuration with parameters (Ω, k, t, b, d) where $k = k_\eta$ and $b = b_0 + b_0 b_1 + \ldots + b_0 b_1 \ldots b_{d-1}$.

Proof: The result obviously holds for $d = 1$ by assumption. To complete the proof by induction, assume the theorem is true for $(d-1)$ stages in the sense that there exists an $(\Omega, k_{d-1}, t, b', d)$ configuration with $b' = b_0 + b_0 b_1 + \ldots + b_0 b_1 \ldots b_{d-2}$. Within any given $(d-1)$-th stage block $B_{h_1 h_2 \ldots h_{d-1}}$, there exists an $(\Omega_{d-1}, k, t, b_{d-1})$ configuration since $|\Omega_{d-1}| \leq k_{d-1}$. The blocks of these configurations may be labelled $B_{h_1 h_2 \ldots h_d}$. Since each successive stage is necessarily of order t, the total system represents an (Ω, k, t, b, d) configuration with $b = b_0 + b_0 b_1 + \ldots + b_0 b_1 \ldots b_{d-2} + b_0 b_1 \ldots b_{d-2} b_{d-1}$ The fact that this system is simple follows from the property that the η-th stage is nested inside of the $(\eta-1)$-th stage.

The above theorem is essentially the same as that given by Ray-Chaudhuri [43] except for the fact that he was concerned with situations in which retrieval pertained to only one level of each of the attributes. In the remainder of this chapter, we shall refer to this situation as the uni-level attribute case (where the prefix "uni" refers to the number of levels relevant to retrieval as opposed to the number of levels which the attribute may assume). By applying Theorem (4.1.1) to the series of configurations arising from Theorem (2.4.1), we may state

Theorem (4.1.2). There exists an (Ω, k, t, b, d) configuration for the situation of v uni-level attributes where $v = (q^{N+1} - 1)/(q - 1)$ and $k = (q^{m_d - 1} - 1)/(q - 1)$ and $b = b_0 + b_0 b_1 + \dots + b_0 b_1 \dots b_{d-1}$ with $b_\eta = b(m_{\eta-1}, t, m_\eta, q)$ for $\eta = 0, 1, 2, \dots, d-1$ and $N \equiv m_0 > m_1 > \dots > m_{d-1} > m_d \geq (t-1)$; and q is a prime power.

The proof of the theorem follows by noting that the attributes are identified with points in PG(N,q) while the first-stage blocks are identified with the m_1-flats of a (b_0, t, m_1) cover thereof. Then the attributes within any one of these blocks are considered as a set Ω_1 and are identified with the points of $PG(m_1,q)$ as it pertains to the corresponding m_1-flat. The second-stage blocks then are taken to be the m_2-flats of a (b_1, t, m_2) cover of the appropriate $PG(m_1,q)$. The process continues until the d-th stage blocks have been formed.

Ray-Chaudhuri discussed Theorem (4.1.2) for the case in which $m_0 = N$, $m_1 = N - 1$, ..., $m_d = N - d$. He also indicated that a series of multi-stage configurations may be derived by combining Theorem (2.4.9) and Theorem (4.1.1). However, because the parameters of such schemes are difficult to specify in a clear-cut fashion, we shall not describe them in any detail. In any event, the basic point of these remarks is that a multi-stage configuration can be formed by combining any theorem which provides the basis of construction for any general series of relevant uni-stage configurations with Theorem (4.1.1).

One particular type of multi-stage configuration which is of interest to us here is formed by combining the procedures of Chapter III with Theorem (4.1.2). The first-stage blocks for the attribute level set $\Omega = \{A_{11}, \dots, A_{1n_1}, \dots, A_{v1}, \dots, A_{vn_v}\}$ are formed in accordance with the method of composition. This has already been described in some detail for $t = 2, 3, 4$. The resulting first-stage blocks contain exactly one level of each attribute. These are then identified with v uni-level attributes to which the results of Theorem (4.1.2) apply. In particular, the formation of (b, t, m) covers of the geometry PG(N,q) has been explicitly described for $t = 2, 3, 4$ in Theorem (2.4.6), Theorem (2.4.7), and Theorem (2.4.8). As a result, with the appropriate construction of the different stages, the number of attribute levels contained in a block can be reduced to a form like $k = (q^t - 1)/(q - 1)$. If this number is still large, one can similarly work with a new geometry PG(N',q'), where $q' < q$ and $(q'^{(N'+1)} - 1)/(q' - 1) \geq k$. Eventually q could be reduced as low as 2 or 3 at which point further stages could be formed, if necessary, by a systematic trial and error procedure. By forming a system as outlined above, one obtains a configuration in which at any stage, the number of blocks which pertain to the next stage is not excessively large. In addition, the number of sub-buckets corresponding to each of the final-stage blocks is of a reasonable magnitude.

The above approach can be supplemented at any time by any of the useful systems considered in previous chapters. For example, when $t = 2$, BIB designs may be used where applicable while for $t = 3$, the schemes of Theorem (1.4.5) and Theorem (1.4.6) are of similar interest. The question of what is the best way to form multi-stage systems is difficult to attack because it is completely entangled with the concept of retrieval time in the corresponding filing systems. Although these concepts will be considered in the next sections, no definite conclusions can really be drawn because as stated in Chapters I and II, the components of retrieval time depend to a very large extent on the properties of the computer systems to be used with the filing schemes.

4.2. Multi-stage combinatorial filing systems.

Ray-Chaudhuri [43] has indicated that a combinatorial filing system may be based on a simple multi-stage combinatorial configuration in the following way. As in the previous sub-section, let $\gamma_d(A)$ denote the d-th stage covering index of the set A in $\mathfrak{a}$; i.e., $\gamma_d(A) = \underset{\sim}{h}$ where $\underset{\sim}{h} = (h_1, h_2, \ldots, h_d)$ if A is contained in $B_{h_1 h_2 \ldots h_\eta}$ but is not contained in $B_{h_1'' h_2'' \ldots h_\eta''}$ for $h_1'' \leq h_1$, $h_2'' \leq h_2$, ..., $h_{\xi-1}'' \leq h_{\xi-1}$, $h_\xi'' < h_\xi$ where $1 \leq \xi \leq \eta \leq d$. Let $\mathfrak{a}_{\underset{\sim}{h}}$ denote the collection of all subsets A of Ω such that $\gamma_d(A) = \underset{\sim}{h}$. To each of the $\underset{\sim}{h}$, A combinations, let there correspond sufficiently large disjoint subsets $M_{\underset{\sim}{h},A}$ of M, the set of addresses. The accession number of the i-th individual's record is stored in an element of $M_{\underset{\sim}{h},A}$ if and only if the largest set which f(I) has in common with $B_{\underset{\sim}{h}} \equiv B_{h_1 h_2 \ldots h_d}$ is the subset A in $\mathfrak{a}_{\underset{\sim}{h}}$; i.e., if $f(I) \cap B_{\underset{\sim}{h}} = A$. Let

$$M_{\underset{\sim}{h}} = \bigcup_{A \in \mathfrak{a}_{\underset{\sim}{h}}} M_{\underset{\sim}{h},A} \tag{4.2.1}$$

The sets $M_{\underset{\sim}{h}}$ represent the d-th stage buckets of the filing system while the subsets $M_{\underset{\sim}{h},A}$ correspond to the sub-buckets. In addition, the sets

$$M_{h_1 h_2 \ldots h_\eta} = \bigcup_{h_{\eta+1}} \cdots \bigcup_{h_d} M_{\underset{\sim}{h}} \tag{4.2.2}$$

$1 \leq \eta < d$ may be identified with the η-th stage buckets.

The retrieval procedure for any query simply involves initially the determination of the appropriate first stage bucket by identifying which first stage block first contains the subset specified in the query. The second and higher stage buckets would then be similarly determined. Afterwards, all sub-buckets within this final d-th stage bucket **and corresponding to subsets which contain the query set are** located and the accession numbers therein obtained. Thus, the retrieval function may be formally written for A in $\mathcal{Q}$ as

$$r(A) = \bigcup_{A \subseteq C \in \mathcal{Q}_{\underset{\sim}{h}}} M_{\underset{\sim}{h}, C} \tag{4.2.3}$$

where $\gamma_d(A) = \underset{\sim}{h}$. In the actual filing scheme, the contents of the blocks B_{h_1} are stored in locations $\ell(h_1)$. Given any query, these are searched sequentially starting at $\ell(1)$ until the first stage covering index h_1^* is determined. The contents of the second stage blocks $B_{h_1 h_2}$ are stored in locations $\ell(h_1, h_2)$. Once h_1^* has been identified, the system switches to $\ell(h_1^*, 1)$ and proceeds sequentially until the second stage covering index (h_1^*, h_2^*) is found. This is continued for each stage until the d-th stage covering index $(h_1^*, h_2^*, \ldots, h_d^*)$ has been determined at the location $\ell(h_1^*, h_2^*, \ldots, h_d^*)$. The contents of the possible subsets C in $\mathcal{Q}_{\underset{\sim}{h}^*}$ are stored in locations of the type $\ell(h_1^*, h_2^*, \ldots, h_d^*; C)$. Each of these are then checked to determine whether C contains A in which case the addresses of $M_{\underset{\sim}{h}, C}$ are noted. Once all the relevant C have been identified, then the corresponding $M_{\underset{\sim}{h}, C}$ are referred to by chaining and the accession numbers therein are extracted.

4.3. Retrieval time in multi-stage filing systems.

Here, retrieval time will be viewed as having three basic components

$T_{1,c}$ = time required to determine the d-th stage **bucket** (i.e., the d-th stage covering index)

$T_{2,c}$ = time required to search among the sub-buckets within the d-th stage bucket

$T_{3,c}$ = time for retrieval which is independent of the structure of the filing system

Proceeding along the lines of Ray-Chaudhuri [43], we assume that $T_{3,c}$ may be neglected. Let τ_η denote the time to test whether the set A in $\mathcal{A}$ is contained in an η-th stage block $B_{h_1 h_2 \ldots h_\eta}$ for $1 \leq \eta \leq d$. If $\gamma_d(A) = \underset{\sim}{h}$, then A must be compared with h_1 first stage blocks, h_2 second stage blocks, . . . , h_d d-th stage blocks. Hence, we have that $T_{1,c}$ is essentially given by

$$T_{1,c} = \sum_{\eta=1}^{d} h_\eta \tau_\eta \quad .$$

Let τ' denote an upper bound on the time to determine whether a sub-bucket contains a set A or not. If $\nu_{\underset{\sim}{h}}$ denotes the number of subsets in $\mathcal{A}_{\underset{\sim}{h}}$, then an upper bound for $T_{2,c}$ is given by

$$T_{2,c} \leq (\tau')(\nu_{\underset{\sim}{h}})$$

Hence, we may write that the retrieval time T(A) for the set A satisfies

$$T(A) \leq \sum_{\eta=1}^{d} h_\eta \tau_\eta + (\tau')(\nu_{\underset{\sim}{h}}) \qquad (4.2.4)$$

If $\tau_0 = \max \tau_\eta$, if $b_\eta = \max b_{h_1 h_2 \ldots h_\eta}$ for $1 \leq \eta \leq (d-1)$ with $b_0 = b_0$ and if $\nu_0 = \max_{\underset{\sim}{h}} \nu_{\underset{\sim}{h}}$, then

$$T(A) \leq \left(\sum_{\eta=0}^{d-1} b_\eta \right) \tau_0 + \tau' \nu_0 \qquad (4.2.5)$$

As a result, we see that an upper bound for T(A) is linearly related to the numbers of blocks in each stage $b_0, b_1, \ldots, b_{d-1}$ and the number of sub-buckets. This is one of the principal reasons why multi-stage systems lead to more efficient retrieval times than similarly structured uni-stage systems. However, the proper choice of which method of staging to use remains largely an open question until more sensitive expressions than (4.2.5) can be developed.

Alternatively, let T_0 denote the average retrieval time required for retrieval of queries A in $\mathfrak{a}$. Let $\nu_{\underset{\sim}{h}}^* = |\mathfrak{a}_{\underset{\sim}{h}} \cap \mathfrak{a}|$ and $\nu = |\mathfrak{a}|$.
Then

$$T_0 \leq \frac{1}{\nu} \{ \sum_{\underset{\sim}{h}} \sum_{\eta} h_\eta \tau_\eta \nu_{\underset{\sim}{h}}^* + \tau' \sum_{\underset{\sim}{h}} \nu_{\underset{\sim}{h}} \nu_{\underset{\sim}{h}}^* \}$$

Suppose the weights (ν_h^*/ν) are approximately equal for different $\underset{\sim}{h}$.
Then

$$T_0 \leq \tau_0 \sum_{\eta=0}^{d-1} \left(\frac{b_\eta + 1}{2} \right) + \tau' \nu_0 \qquad (4.2.6)$$

The interpretation of (4.2.6) is essentially the same as that given previously for (4.2.5).

Finally, let us consider the redundancy of the systems described here for the case of v attributes with s levels each under the assumption of a uniform distribution of records. Since a record is not stored in a first stage bucket if the individual has no attribute levels in common with the corresponding block, an upper bound to the redundancy R_c is

$$R_c \leq \{ 1 - (\tfrac{s-1}{s})^v \} \; b_0 \; . \qquad (4.2.7)$$

The actual redundancy bound cannot be more exactly approached because it is difficult to measure the effect of the fact that a record will not be stored in a first stage block if the intersection of that block and the record is a subset contained in one of the prior blocks.

4.4. Example.

Suppose there are $v = 256$ attributes with $s = 2$ levels each. **In** addition, suppose that a filing system **oriented toward efficient retrieval of first and second order queries is desired.** Using the method of composition as outlined in the proof of Theorem (3.2.1), we can base the first stage buckets on the $b_0 = 17$ blocks of a second order combinatorial configuration with $k = v = 256$. If the $k_0 = k = 256$ elements in each first stage block are identified with the points of EG(2, 16), then the second stage blocks may be taken to correspond to the $b_1 = 272$ lines therein, each of which pertains to $k_1 = 16$ attribute levels. Continuing in the same manner, the third stage blocks may be identified with the $b_2 = 20$ lines of EG(2, 4) where the points are associated with the k_1 elements assigned to a second stage block. As a result, there are $k_2 = 4$ elements in each third stage block. Each corresponding bucket contains as many as $\nu_0 = (2^4 - 1) = 15$ sub-buckets. Hence from (4.2.5), we have

$$\begin{aligned} T(A) &\leq (17 + 272 + 20)\ \tau_0 + 15\ \tau' \\ &\leq 309\ \tau_0 + 15\ \tau' \end{aligned}$$

Alternatively, since b_1 is somewhat large, another scheme of staging may be more worthwhile. Let the $k_0 = 256$ elements in a first stage block be divided into 16 groups of 16. Let each of these groups be identified with a point in EG(2, 4) and form second stage blocks as the $k_1 = 64$ elements corresponding to the four points on a line; this leads to $b_1^* = 20$ second stage blocks. Similarly, let the $k_1 = 64$ elements of a second stage block be divided into 16 groups of 4 and form $b_2^* = 20$ third stage blocks of $k_2 = 16$ elements each by again using the structure of EG(2, 4) in a similar fashion. The fourth stage is then obtained by proceeding once again in essentially the same way but with respect to 16 groups of 1. This gives $b_3^* = 20$ and $k_3 = 4$. Finally, as in the preceding situation, each of the fourth stage buckets has at most $\nu_0 = 15$ sub-buckets. Thus from (4.2.5), we have

$$\begin{aligned} T(A) &\leq (17 + 20 + 20 + 20)\, \tau_0 + 15\, \tau' \\ &\leq 87\, \tau_0 + 15\, \tau' \end{aligned}$$

Hence, this multi-stage system is more efficient with respect to retrieval time than the one initially outlined. On the other hand, the total number of final stage buckets here is (17)(20)(20)(20) as compared with (17)(272)(20) in the previous system; i.e., this system involves nearly (1.5) times as many final stage buckets. Whether this added magnitude causes any problems represents a question which is difficult to evaluate. In some sense, however, any solution will rest on the properties of the computer system involved.

CHAPTER V

SOME PROBLEMS FOR FUTURE RESEARCH

Even though a variety of different filing schemes have been considered here, a great deal of further research is needed. More efficient systems for cases in which different attributes assume different numbers of levels represent one area. Also, compromise designs, which are suitable for one type of query with respect to some sets of attributes and other types of queries with respect to other sets of attributes, need to be developed for the cases where they are applicable. Other types of schemes which are of interest are those suitable in situations where some types of queries are retrieved more often than others and those which enable efficient retrieval of queries involving more than one level of each attribute. Finally, before the different systems currently in existence can be effectively compared with one another, the concepts of retrieval time and redundancy need to be more explicitly developed. When this has been achieved, then one will be able to specify more completely the type of properties which are desirable for filing systems.

BIBLIOGRAPHY

[1] Abraham, C. T., Ghosh, S. P., and Ray-Chaudhuri, D. K., "File organization schemes based on finite geometries," IBM Research Report RC-1459, Yorktown Heights, New York; IBM Watson Research Center, August 1965.

[2] Baker, F. T., "Some storage organization for use with disk files," IBM Federal Systems Division Report, 1963.

[3] Bose, R. C., "On the construction of balanced incomplete block designs," Annals of Eugenics, Vol. 9 (1939) pp. 353-399.

[4] Bose, R. C., "Some new series of balanced incomplete block designs," Bulletin Calcutta Mathematical Society, Vol. 34 (1942) pp. 17-31.

[5] Bose, R. C., "Mathematical theory of the symmetrical factorial designs," Sankhya, Vol. 8 (1947), pp. 107-166.

[6] Bose, R. C., "On a resolvable series of balanced incomplete block designs," Sankhya, Vol. 8 (1947), pp. 249-256.

[7] Bose, R. C., "On the application of finite projective geometry for deriving a certain series of balanced Kirkman arrangements," The Golden Jubilee Commemeration Volume Calcutta Mathematical Society, 1958-59, pp. 341-354.

[8] Bose, R. C., "On some connections between the design of experiments and information theory," Bulletin of the International Statistical Institute," Vol. 38, Part 4 (1961), pp. 257-271.

[9] Bose, R. C., Abraham, C. T., and Ghosh, S. P., "File organization of records for multiple-valued attributes for multi-attribute queries," Proceedings of the Symposium on Combinatorial Mathematics, Chapel Hill, North Carolina: University of North Carolina Press, 1967

[10] Bose, R. C. and Bush, K. A., "Orthogonal arrays of strength two and three," The Annals of Mathematical Statistics, Vol. 23, No. 4 (December 1952), pp. 508-524.

[11] Bose, R. C. and Connor, W. S., "Combinatorial properties of group divisible incomplete block designs," The Annals of Mathematical Statistics, Vol. 23, No. 3 (September 1952), pp. 367-383.

[12] Bose, R. C. and Kishen, K., "On the problem of confounding in the general symmetrical factorial design," Sankhya, Vol. 5 (1940), pp. 21-36.

[13] Bose, R. C. and Nair, K. R., "Partially balanced incomplete block designs," Sankhya, Vol. 4 (1939), pp. 337-372.

[14] Bose, R. C. and Ray-Chaudhuri, D. K., "On a class of binary error-correcting group codes," Information and Control, Vol. 3 (1960), pp. 68-79.

[15] Bose, R. C. and Ray-Chaudhuri, D. K., "Further results on error-correcting group codes," Information and Control, Vol. 3 (1960), pp. 279-298.

[16] Bose R. C. and Shrikhande, S. S., "On the composition of balanced incomplete block designs," Canadian Journal of Mathematics, Vol. 12 (1960), pp. 177-188.

[17] Bose, R. C., Shrikhande, S. S., and Bhattacharya, K. N., "On the construction of group divisible incomplete block designs," The Annals of Mathematical Statistics, Vol. 24, No. 2 (June 1953), pp. 167-195.

[18] Bose R. C., Shrikhande, S. S., and Parker, E. T., "Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler's conjecture," Canadian Journal of Mathematics, Vol. 12 (1960), pp. 189-203.

[19] Bose, R. C. and Srivastava, J. N., "On a bound useful in the theory of factorial designs and error correcting codes," The Annals of Mathematical Statistics, Vol. 35, No. 1 (March 1964), pp. 408-414.

[20] Buchholz, Werner, "File organization and addressing," IBM Systems Journal, Vol. 2 (June 1963), pp. 86-111.

[21] Bush, K. A., "Orthogonal arrays of index unity," The Annals of Mathematical Statistics, Vol. 23, No. 4 (December 1952), pp. 426-434.

[22] Carmichael, R. D., Introduction to the Theory of Groups of Finite Order, Boston, Massachusetts: Ginn and Co., 1937.

[23] Davis, D. R. and Lin, A. D., "Secondary key retrieval using an IBM 7090-1301 system," Communications of the Association for Computing Machinery, Vol. 8, No. 4 (1965), pp. 243-246.

[24] Chakravarti, I. M., "Fractional replication in asymmetrical factorial designs and partially balanced arrays," Sankhya, Vol. 17 (1956), pp. 143-164.

[25] Chakravarti, I. M., "On some methods of construction of partially balanced arrays," The Annals of Mathematical Statistics, Vol. 32, No. 4 (December 1961), pp. 1181-1185.

[26] Chakravarti, I. M., "On the construction of difference sets and their use in the search for orthogonal Latin Squares and error-correcting codes," 35-th Session of the International Statistical Institute, 1965.

[27] Ghosh, S. P., "On the construction of balanced incomplete block designs using non-degenerate quadrics in finite projective geometry," IBM Research Report RC-1784, Yorktown Heights, New York: IBM Watson Research Center, March 1967.

[28] Ghosh, S. P. and Abraham, C. T., "Application of finite geometry in file organization for records with multiple valued attributes," IBM Research Report RC-1561, Yorktown Heights, New York: IBM Watson Research Center, March 1966.

[29] Hanan, M. and Palermo, F. P., "An application of coding theory to a file addressing problem," IBM Journal Research and Development, Vol. 7, No. 2 (April 1963), pp. 127-129.

[30] Hocquenghem, A., "Codes Correcteurs d'Erreurs," Chiffres, Vol. 2 (September 1959), pp. 147-156

[31] Johnson, L. R., "An indirect chaining method for addressing on secondary keys," Communications of the Association for Computing Machinery, Vol. 4, No. 5 (May 1961), pp. 218-222.

[32] Peterson, W. W., "Addressing for random-access storage," IBM Journal Research and Development, Vol. 1, No. 2 (April 1957), pp. 130-146.

[33] Peterson, W. W., Error Correcting Codes, Cambridge, Massachusetts: MIT Press and John Wiley and Sons, 1961.

[34] Plackett, R. L. and Burman, J. P., "The design of optimum multi-factorial experiments," Biometrika, Vol. 33 (1943-1946), pp. 305-325.

[35] Primrose, E. J. F., "Quadrics in finite geometries," Proceedings Cambridge Philosophical Society, Vol. 47 (1951), pp. 299-304.

[36] Rao, C. R., "Factorial experiments derivable from combinatorial arrangements of arrays," Journal Royal Statistical Society, Supplement, Vol. 9 (1947), pp. 128-139.

[37] Rao, C. R., "The theory of fractional replication in factorial experiments," Sankhya, Vol. 10 (1950), pp. 81-87.

[38] Rao, C. R., "A general class of quasi factorial designs," Sankhya, Vol. 17 (1956), pp. 165-174.

[39] Rao, C. R., "A study of BIB designs with replications 11 to 15," Sankhya, Series A, Vol. 23 (1961), pp. 117-127.

[40] Rao, C. R., "Combinatorial arrangements analagous to orthogonal arrays," Sankhya, Series A, Vol. 23 (1961), pp. 283-286.

[41] Ray-Chaudhuri, D. K., "Some results on quadrics in finite projective geometry based on galois fields," Canadian Journal of Mathematics, Vol. 14 (1962), pp. 129-138.

[42] Ray-Chaudhuri, D. K., "Application of the geometry of quadrics for constructing PBIB designs," The Annals of Mathematical Statistics, Vol. 33 (1962), pp. 1175-1186.

[43] Ray-Chaudhuri, D. K., "Combinatorial information retrieval systems for files," IBM Research Report RC-1554, Yorktown Heights, New York: IBM Watson Research Center, February 1966.

[44] Schay, G. and Spruth, W. G., "Analysis of a file addressing method," Communications of the Association for Computing Machinery, Vol. 5, No. 8 (August 1962), pp. 459-462.

[45] Schay, G. and Raven, N., "A method for key-to-address transformation," IBM Journal Research and Development, Vol. 7, No. 2 (April 1963), pp. 121-126.

[46] Sprott, D. A., "A study of BIB designs with replications 16 to 20," Sankhya, Series A, Vol. 24 (1962), pp. 203-207.

[47] Verblen, O. and Bussey, N. J., "Finite projective geometries," Transactions American Mathematical Society, Vol. 7 (1906), pp. 241-259.