On designing generalized file records for management information systems

by FRANK H. BENNER
Bell Telephone Company of Pennsylvania
Philadelphia, Pennsylvania

INTRODUCTION

The centrality of the files in a MIS (Management Information System), or in any other large scale computer application, has been traditionally considered as self-evident. The new problems of operating systems and the languages for use in the 3rd generation computers, plus the relatively unknown world of integrated data communications may tend to remove the files from the limelight. This should be avoided. Most business applications will continue to be "file bound," or I/O limited, even with random access to the data base.

A file reorganization to correct design deficiencies can be catastrophic in its impact on programs that were "on-the-shelf." The majority of the system hardware cost is probably charged to file devices. The continuing requirements for an optimum design for the complete system is recognized and honored. It is suggested that the search for this elusive design solution should begin in the files area. Here the payoff is handsome for success, and the fiscal and operational deficit is unrelenting for failure.

In a changing business environment, the mortality of a special-purpose file organization is expected to be high. The prudent approach for the designer is to make no a priori assumptions about the way in which the data base will be used, new uses will arise; or the limit of the required descriptors, new data will be added. Both R. V. Head[1] and W. H. Desmonde,[2] while describing different real-time systems, are strong in their emphasis on file organization as a limiting factor on throughput and cost, which requires rigorous optimizing techniques. A new approach to file design seems required which takes advantage of the "naturalness" that may exist in families of information. Paraphrasing Alexander,[3] the difficulties attended upon third generation computer applications are much more subtle and complex than in the past. As in the usual case, the temptation to fall back on some arbitrarily chosen order or design technique is almost overwhelming. But, if we continue to apply essentially punched card or tape file design techniques to the direct access file problem, we will live with essentially an unsolved problem and pay for the designers comfort.

Design criteria

This paper reports on a method for analyzing the logical record requirements and designing the physical records to be housed on direct access storage devices. This methodology has been followed in the Pennsylvania Company to design the file system for BIS (Business Information System), a specific term used in the Bell System to identify an indigenous MIS. This system is characterized by large direct access files, a variety of real-time activity to the files with "background" work during the business day, and intensive and periodic batch processing of the files on non-prime time. These characteristics are, in many ways, similar to the corporate MIS for many businesses.

A MIS usually consists of two operational categories of programs; real-time and batch. The file system to support these processes must at least consider;

1. Security of information.
2. Recovery of system operation after failure, and restoration of data when mutilated.
3. Key and Addressing schemes that will result in densely populated storage.
4. Need for additions and changes to the data base to meet changing requirements.
5. Performance requirements of all types of activity.
6. Reduction of wasted mass storage, consistent with performance requirements.
7. Coordination of the file system with the programming system.

Of the above, security, recovery, restoration and addressing do not lend themselves to a universal treatment. The speed requirements for recovery of operation and

correction of mutilated data is not the same for all systems. Some can tolerate times in terms of days; others demand action in minutes. Economics is also a factor in designing for system assurance. Duplexed files are expensive, but not always required. Each application has its own unique security problems and scheme for identifying records. The requirement for an optimum file record is present in all applications however.

The record designer must be able to qualitatively describe this optimum record before attempting his work. When this record is designed, his search is ended.

The optimum record will:

1. Utilize the greatest amount of the space allocated at the home address of the storage device.
2. Permit selection of data from within the record.
3. Be structured so that data fields can be added, eliminated, and rearranged.
4. Identify records by using a logical description, in addition to absolute identification.
5. Be as short as possible and yet, in the selected length, supply the greatest amount of file data to the programs in one "seek."

Logical record considerations

Most business applications will be dealing with a logical record in the data base. In an Inventory Application, the logical record may be the set of all the subsets whose elements contain information primarily about a particular Piece Part. In an Order and Billing Application, the logical record is usually a Customer Account. With a total systems approach as used in a MIS, those applications which were previously separate bounded systems, now are subsumed into the MIS or are subsystems. If we were to arbitrarily retain the demarcation between say, an Inventory System and an Order and Billing System in an MIS environment, there would be unnecessary redundancy of data and a high likelihood of subjectively favoring one of the subsystems of the MIS at the expense of the others.

The amount of information making up a logical record is usually not homogeneous throughout the universe of the records. Some of our inventory items or our customers are more active or larger than others. Thus, the logical record size will most likely vary. We could make each physical record the size required to accommodate information in our largest logical record. This would give excellent performance, but low utilization of storage would result. We could make our record length cater to the "average." This would give us good storage utilization. Since it is usually the larger and complex entities that are most active, "average" record length results in lower system performance. A disciplined approach to record design is required to properly weigh all factors.

Attributes of record components

Analysis of the natural components of a logical record is aided by the diagram, Logical Record Components, Figure 1. We see that a Logical Record is the superset of functionally related Main and Auxiliary Records uniquely identified with a particular entity. The subsets of the Main and Auxiliary Records are the Data Segment(s) and one Record Control Segment. A Data Segment is a collection of fields that cover a particular aspect of the Logical Record. Then, the elements of the Data Segment are Data Fields, with possible intersection shown. Each Logical Record has one Main Physical Record and zero or more Auxiliary Physical Records, depending upon the amount and kind of data making up the record.

There has been an increased interest in memory systems where information is stored or retrieved on the basis of content. This is in contrast with using specific addresses to reference locations containing data that must be examined. This rationale is generally referred to as being "associative" in orientation.[4] Since about 1961 the term "associative criterion" has been used to define the list of descriptors for the desired group of data or record. In this context an item for/of entry in an associative memory is described by as many characteristics as are required. The object record is located or identified at the intersection of all these descriptors. Using the associative philosophy of the memory system designer, it is a short step into a new logical organization for file records.[5] Each record can still be specifically identified, but it also may be understood from the associative logical criterion. Each set of descriptors then describes an object, but the object described is not always congruent with only one specific record. This idea permits the accommodation of more than one logically consistent file in the same physical file structure. The value of such an organization is its generality of logical flexibility.

Fitting the physical records into such a file requires an examination of the record elements. In a direct access record, the basic building block is the data field. Close ties between fields are reflected by the creation of data segments. We must know specific information about the data fields so we can determine the particular components of an optimum main record These attributes relate to:

1. How often the field even exists to contain significant data?
2. When it does exist, how long is the field?

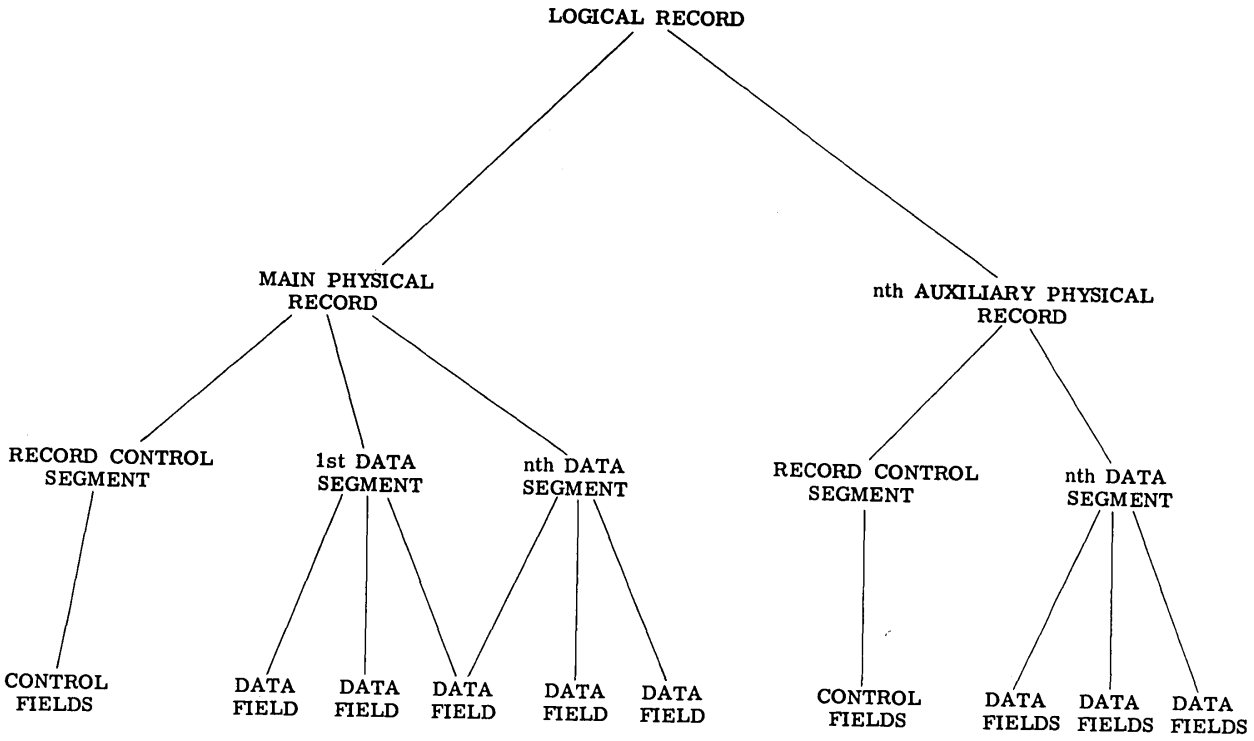

Figure 1—Logical record components

3. How often is the field required when the Logical Record is processed?
4. When it is required for use, what other fields are also needed?

The information pertaining to how often a field significantly exists aids in attaining efficient use of storage. The length of a field and its variance help determine the use of storage, and how often the system will be required to access "overflow" data. The way in which, and how often, the data is used in servicing transactions or processing requirements will permit a design that lets each application "stand up and be counted" on its true need.

These facts about a mythical record are shown on Data Segment and Field Attributes, Figure 2. Each column on the chart contains specific information about the data field or data segment that appears on a particular row. Note that Segments A and D have no component fields shown. This is because the fields are a tight family, (exist with practically the same probability and level of activity) so they are considered as being one large field. The column headed Prob. refers to the probability that the field will contain significant data in the Logical Record.

$$P_i = \frac{E_i,}{N} \quad P_i \leqq 1.00 \qquad (1)$$

where: E is number of Logical Records in which i^{th} field $\neq$ O,

and: N is number of Logical Records in the data set

The Mode column refers to the characteristics of the field in terms of being fixed (F) in length or variable (V). In fixed length cases, only the Min. column is used. The details of how a length varies are required only for variable length fields or segments. This is done by expressing the field length and frequency as a series of coordinates in the columns under Data Length headed Pt. 1, Pt. 2, Pt. n. In these columns, L and f (L) are shown for the various points. The Min column pertains to the shortest length the field was found to have.

Activity on the system is expressed on Figure 2 in terms of the frequency of need in a given time period for the particular fields and segments. The Real-Time column pertains to the transactions to be conducted in real-time. In the illustration, two different kinds of real-time transactions have been shown.

Type 1 requires Segment A, Segment B Field 1, and Segment C, Fields 1 & 2

Type 2 requires Segment A, Segment C Field 1, and Segment D

The Batch column refers to the requirements for

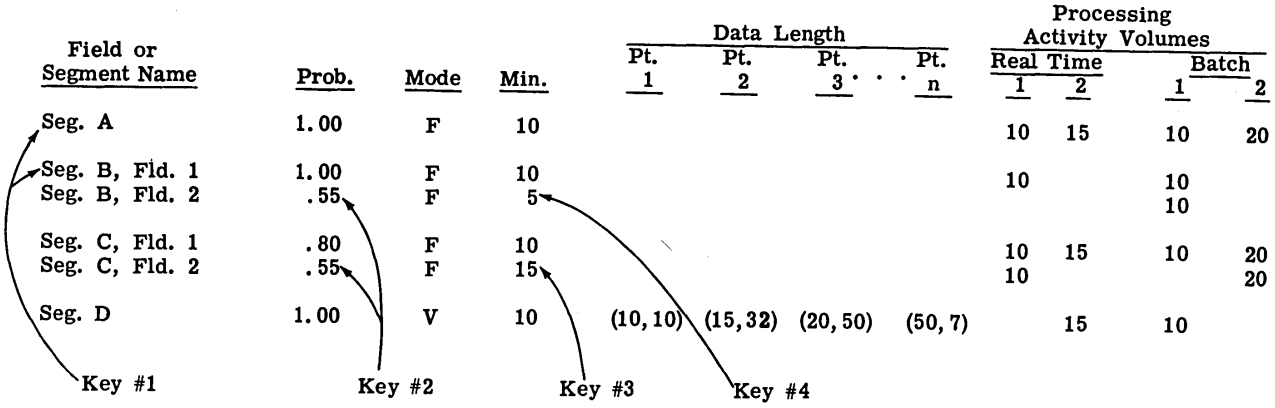

Figure 2—Data segment and field attributes

information to meet these similarly appropriate needs. The volumes shown must be taken for the same period of time. Total activity to a Segment or Field is simply:

$$A_t = E \sum_{i=1}^{n} RT_i + \sum_{j=1}^{n} BTCH_j \qquad (2)$$

where: i is a real-time transaction
j is a batch transaction
E is $\geqq 1$

and: A_t is Total Weighted Activity

The strategy to be employed later will involve certain computations that use the length, probability, and the activity for each of the data segments. Most of these file systems must serve both real-time and batch processes.

If the file is to have no particular orientation, all of values L, P, and A can be used as they are found. If there is to be an orientation toward the real-time requirements, the records should be so designed. The factor E in the above equation (2) is used to effect this orientation. It is a number, proportional to the relative importance of the real-time activity. Arriving at this value is similar to assigning the priority or the transaction limits for the use of the control package or operating system. For instance, the control program might be arranged to permit a limit of say three or four real-time transactions to be served and then one background transaction. This same limit can be effectively used as E.

From an examination of raw activity, shown on Data Segment and Field Attributes, Figure 2, and, if we wish to emphasize the influence of our Real-Time activity by E = 2 as discussed earlier, equation (2), we have weighted values:

Segment or Field

	Weighted Real-Time	Batch	A_t*
Segment A	50	30	80
Segment B, Field 1	20	10	30
Field 2	—	10	10
Segment C, Field 1	50	30	80
Field 2	20	20	40
Segment D	30	10	40

*Total Activity

By using this Total Weighted Activity, along with the related probability and the length we will be able to evaluate the relative importance of a field or segment.

Utilization of mass storage

The designer faces an implied constraint in the form of the economic use of storage. This is evidenced in his reluctance to provide space in the record for fields that occur infrequently. In tape records, this problem could frequently be taken care of by the placement of these fields where "zero suppression" could keep the tape record shorter if the field contained no significant data. In direct access devices there are as yet no such aids, so the decision cannot be so easily determined. The designer must have an objective in terms of storage utilization. Storage utilization is simply the probability that a position in storage will contain significant data.

Fixed length data fields

From the data on Figure 2, the designer has no problem with fixed length Segment A and Field 1 of Segment B, since there is a P equal to 1, Key #1. This tells him that they always significantly exist. The P for Field 2 Segment B, and Field 2 Segment C, Key #2, with P equal to .55, present a dilemma. If his objec-

tive is a utilization say of .70, the designer is tempted not to reserve space for these fields. This is fallacious if we consider only the lengths of the fields, since if a field might not exist, the problem programs must be able to test a control field associated with the object field to see if it is present.

Pan[6] in his pioneering work provided useful formulae for applying linear programming techniques to help solve the record design problem. In this particular example, after Pan, we can test the fixed length fields Field 2 Segment B, Field 2 Segment C, Key #2, and determine if we should always provide space in our physical record for these fields. To do this, the term Weighted Storage Utilization is used which is essentially the probability for the field, weighted by the ratio of a control length to the data length of the field.

$$W_i = \frac{L_i + C}{L_i} (P_i) \qquad (3)$$

where: L_i is the data length of the ith field
P_i is the probability of the ith field
C is the length of the control field used in the programming system.

If the programming system requires 3 digits for control to provide field identification and length, we have for Field 2 Segment C, on Figure 2:

$$W_i = \frac{L_i + C}{L_i}(P_i)$$

$$= \frac{15 + 3}{15}(.55)$$

$$= .66$$

where: L = 15 per Key #3, Figure 2.
For Field 2 Segment B, on Figure 2:

$$W_i = \frac{5 + 3}{5}(.55)$$

$$= .88$$

where: L_i = 5 per Key #4, Figure 2

Assume the rule, "Always allot space for data in the physical record when the Weighted Storage Utilization of the field equals or exceeds the design objective." Recalling our objective of .70, space will always be provided for Field 2 Segment B, $W_i = .88$; but none will be reserved for Field 2 Segment C, $W_i = .55$. If the application programs are "process" limited, borderline cases may be judged accordingly. This is due to some additional processing that is required to locate data using a control field, rather than by direct use of a "label" that designates specific positions in memory.

Variable length data

The problem with variable length fields, in terms of utilization of storage, is more complex. Here, in addition to treating the probability of the field, we also face the problem of deciding on a data length to provide on the first "read" of the record containing the field. Most methods of organizing records for direct access devices use a "chaining" or "overflow" address when data exceeds the space initially allotted. This "chaining address" field, incidentally, should be treated as any other fixed length field since it significantly exists only when overflow exists. The use of the additional detail on variable length fields is fairly obvious. It wlll be used to produce a probability function for each particular field. The skew present in a frequency distribution of the data has shown this essentially social data transforms, with good fit, into a form of the Gompertz Curve.[7] This curve fits well into a probability function where we encounter the probability showing an increasing ratio of decline as field length increases, but the ratio is not changing by either a constant amount or percentage. While this curve:

$$Y = ka^{b^x} \qquad (4)$$

is mathematically attractive, it cannot easily be managed for solution and integration in a digital computer. Raw data is used directly for this purpose.

The need for integration is to develop the storage utilization for variable length data. Recall that this is the probability that a position of allocated storage will contain significant data, or the ratio of the allotted area to the occupied area. For variable length fields:

$$W_i = \frac{\sum_{i=0}^{n} f(L_i)dL + C}{LA_i}(P_i) \qquad (5)$$

where: i is a particular variable length field,
LA is length of allotted area, and,
C is the length of chaining address data.

Morse[8] treated a similar problem in computing Mean Service Time by using Taylor's Theorem. While our problem is not one of queues, it can be treated in a parallel manner. Figure 3 is a graph of the Probability Function of a Variable Length Field. First, consider the probability that the field length, for any record in which the field appears, is greater than length L. This is the difference between the probability that the field will be of length (L) and (L + dL).

$$P(L) - P(L+dL) = P(L) - P(L) - \frac{dP}{dL}\,dL = p(L)dL \qquad (6)$$

where: $p(L) = -\frac{dP}{dL}$ or $P(L) = \int_0^\infty p(L)dL$

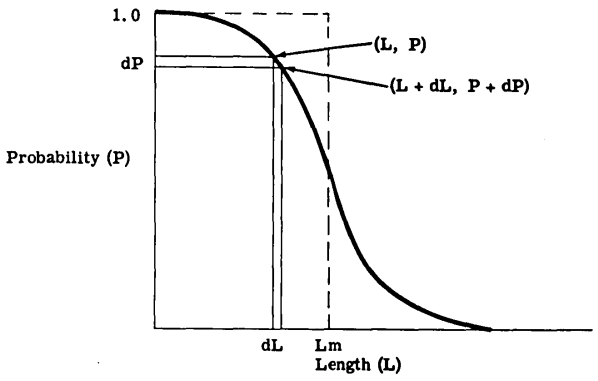

Figure 3—Probability function of a variable length field

This quantity p(L) is the probability density that a field is of length L. This is a rate or change in probability per length. Then p(L) is a measure of the mean rate at which Fields are meeting the length (L). The average length of the fields is:

$$L_m = \int_0^\infty L\,p(L)dL$$

$$= -\left[L\,P(L) \right]_0^\infty + \int_0^\infty P(L)dL$$

$$= \int_0^\infty P(L)dL \qquad (7)$$

where: L_m is the average length.

Discarding the quantity in the square brackets is justified since it is zero when (L) equals zero; and when $(L)\rightarrow\infty$, P(L) has already reached zero for all practical purposes.

Using the Trapezoidal Rule for approximation of the definite integral of the Probability Density in Figure 4, Functions for Segment D:

$$A \approx \int_{X_o}^{X_n} f(x)\,dx \qquad (8)$$

$$\approx \frac{\Delta x}{2}(Y_o + 2Y_1 + 2Y_2 + 2Y_{n-1} + Y_n)$$

$$\approx \frac{5}{2}(1 + 2 + 2 + 1.88 + 1.52 + .96 + .54 + .38 + .26 + .16 + .04)$$

$$\approx \frac{5}{2}(10.74)$$

$$\approx 26.85 \approx 27$$

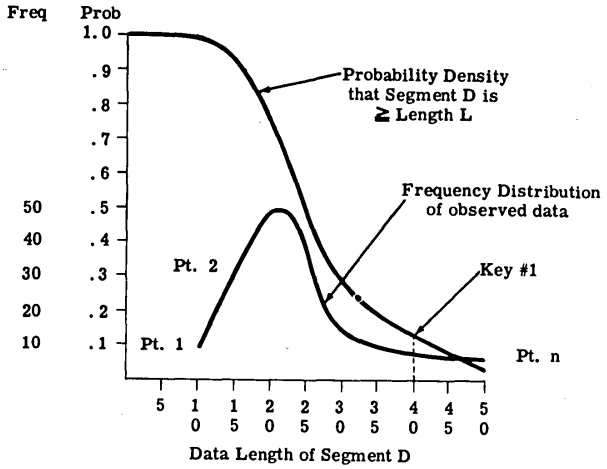

Figure 4—Functions for segment D

Applying this result to equation (5) for Segment D, and assuming "C" equals 3 digits,

$$W_i = \frac{\sum_{i=0}^{n} f(L_i)\,dL + C}{LA_i}(P_i) \qquad (9)$$

$$= \frac{27+3}{50}(1.00)$$

$$= .60$$

This would be true if LA_i is set equal to the maximum length the field was found to have. Therefore, the length of the allotted space should be reduced until W_i equals the objective, if possible. Through an iterative process we arrive at:

$$W_i = \frac{26+3}{40}(1.00) \qquad (10)$$

$$W_i = .72$$

So that if we allot 40 characters to Segment D, we will have a Storage Utilization of .72, the field will always appear in the physical record. From Figure 4, Key #1, the probability that Segment D≧40 characters is .12 or; about 12% of the times Segment D is required, an additional read would be required to secure the excess data. The number of device read commands that must be given to completely satisfy the need for specific data is an important factor on through-put. Therefore, it is necessary to keep file action to a minimum. The ability of a file record to meet the needs for data is expressed as "Performance," and is calculated as follows:

$$C_r = \{S_1, S_2, \text{----}S_n\} \qquad (11)$$

$$P(C_r) = \frac{\sum_{i=1}^{n} H_i N_i}{\sum_{i=i}^{n} N_i} \qquad (12)$$

Where: S_i is a Data Segment provided for in the combination C_r being rated for Performance, and, $P(C_r)$ is the Performance of C_r.

Where: i is the number of different segment combination required by transactions.
N is the number of times a particular combination is requested by processing needs.

and: H_i is a "hit" factor
$H_i = 1$ if all required segments are found with C_r
$H_i = 0$ if one or more of the required segments are not with C_r

Example:

$C = \{a, b, c, d, e\}$ is the combination being rated for the main physical record.

The different combinations requested:

Combination	H_i	Processing Volumes
$1 = \{a, b, c,\}$	$H_1 = 1$	$1 = 50$
$2 = \{a, c, e,\}$	$H_2 = 1$	$2 = 40$
$3 = \{a, b, f,\}$	$H_3 = 0$	$3 = 10$

then:

$$P(C_r) = \frac{\sum_{i=1}^{n} H_i N_i}{\sum_{i=1}^{n} N_i} \qquad (13)$$

$$= \frac{(1).(50) + (1).(40) + (0).(10)}{50 + 40 + 10}$$

$$= \frac{90}{100} = .9$$

Storage Utilization is a similar rating that measures the expectation that a character position of storage will be occupied by useful intelligence from the data base.

$$SU = \frac{\sum_{i=1}^{n} L_i W_i}{\sum_{i=1}^{n} L_i} \qquad (14)$$

where:

L_i is Segment length
W_i is Weighted Storage Utilization for the segment

Record design strategy

The designer will naturally attempt to produce alternate main record designs that will differ in:

1. Length
2. Performance
3. Utilization of Storage

There are a number of techniques that efficiently utilize available storage addresses.[9,10] In this strategy, we attempt to make the best use of the capacity available at the addresses. The combination of segments chosen for the Main Physical Record will determine the number of times that required data is not initially available to the system activity. In this procedure, the designer first expresses these factors:

1. The length of the control fields to be used by the Programming System for Variable Format Fields (Figure 5, Memory Map of Data Segment, Fields I_i and L_i).
2. The length of the chaining fields when used for variable length data.
3. The factor "E", as discussed earlier, gives emphasis to real-time activity in the design.

The basic strategy is shown on the flow chart on Figure 6, Design Procedure. When the Weighted Storage Utilization is low, most fields will be made fixed in format, and variable length fields will seldom require a second access to overflow. The record will be long and the Storage Utilization will be low. For each value of Weighted Storage Utilization there will be some length where the Performance is acceptable. The designer then plots these Performance and Storage Utilization results for each of the feasible lengths. The optimum record length is selected for implementation; the contents of the record is the combination of segments used to develop the selected results. On Figure 7, Graph of Performance and Storage Utilization, is shown the plot of Performance (solid) and Storage Utilization (dotted) that were generated from information for a portion of the BIS File System.

As record length increases, Performance increases since more segments are present in the record. Conversely, Storage Utilization decays since data with lower P are being included. There is a minimum performance that the selected record length must provide. The decision on exceeding that length is a tradeoff between the increase in performance that will be enjoyed, versus the decrease in the utilization of storage that will be encountered. For instance, on Figure 7, if

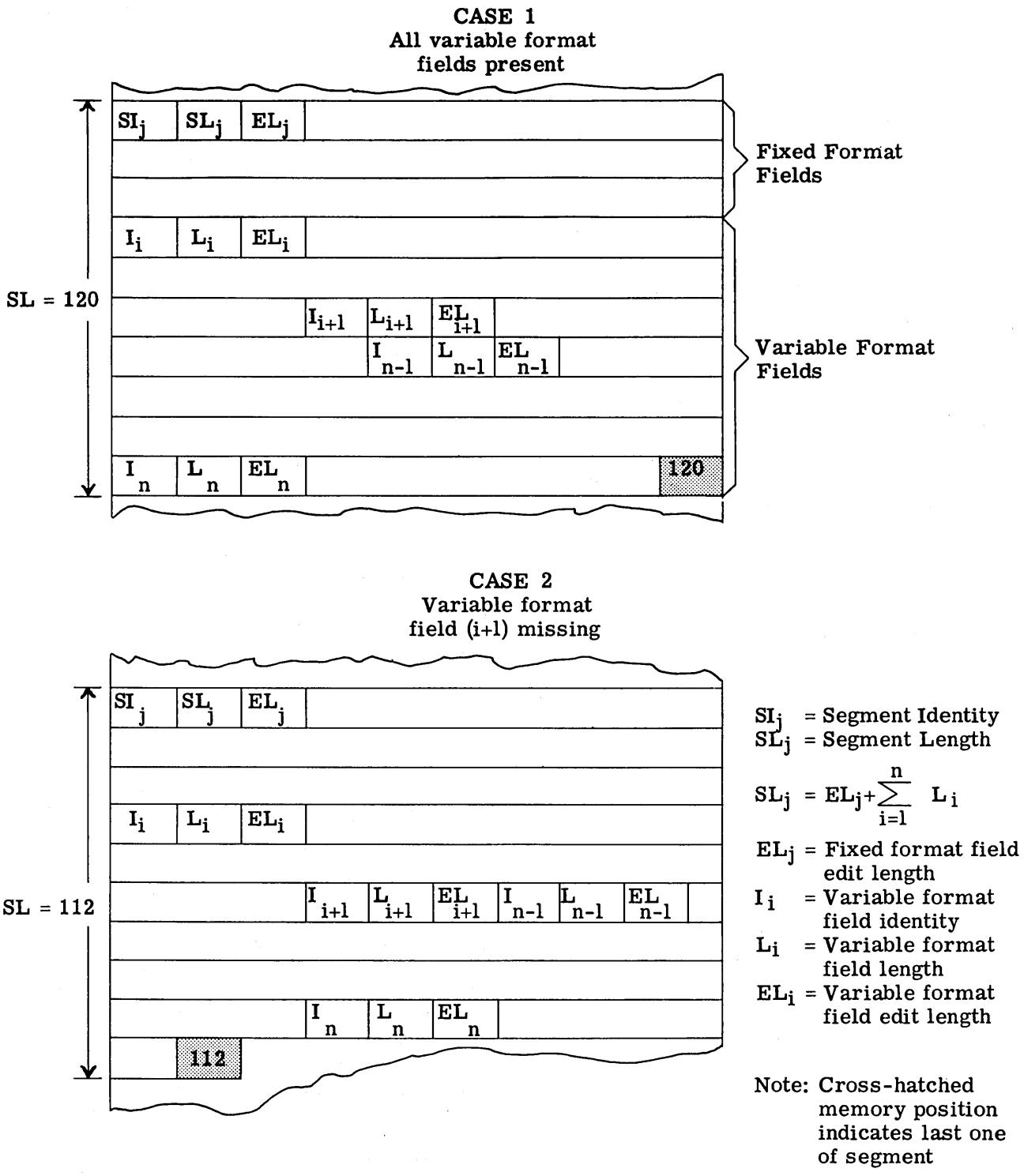

Figure 5—Memory map of data segment

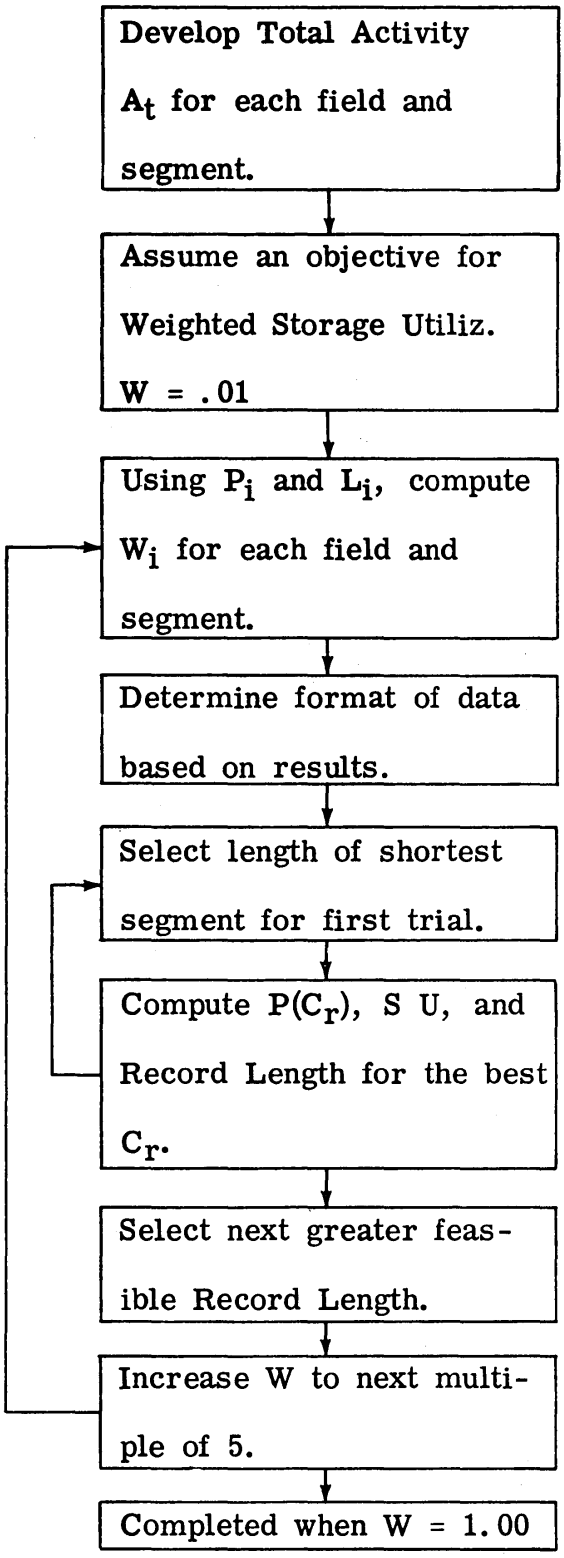

Figure 6—Design procedure

a minimum Performance of .90 is required, the length of such a record will be 600 characters. If the next feasible length is 700 characters, the decision to use this length is based on the absolute change in Performance and in Storage Utilization. A useful rule is to subject the change, or Gain ratio to a slope test:

$$G = \frac{\Delta P}{\Delta SU} = \frac{.02}{.05} = .4 \text{ If } G \geqq 1, \text{ take next length} \quad (15)$$

There are times when certain solutions might be ruled out due to address breaks, device constraints, and economics. This action should only be taken after the theoretical solution is found. These techniques may be difficult to use without some computer aids. Most designers will have a linear programming package available to them, or can have one written. Many of the ideas in this design method appear in a program[11] available from the IBM Corp and used in the Pennsylvania Company. Certain restrictions and specific understandings are in force when it is used.

After the contents of the Main Physical Record has been decided, the rest of the logical record will be in the Auxiliary Record(s). These records will contain seldom used data and the excess of the variable length data from the main record.

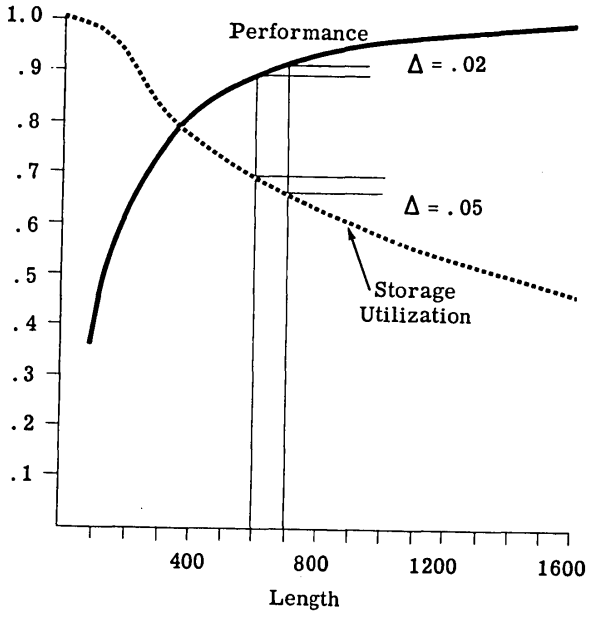

Figure 7—Performance and storage utilization

Programming considerations

Until now there has been emphasis on storage utilization, throughput and efficiency. However, it is necessary to place the resulting record in the context of the programming system. A fairly representative selection of storage devices[12,13,14] show a variety of capacities and speeds available. Capacity and high cost has been softened as a limit. On Figure 5 is shown how a segment will appear in main memory after it has been read from the device. Each segment must carry its identity (SI_j), or, type of data it contains. The total length of the segment (SL_j) must be shown for programming housekeeping. The segment length is a function of the length of the fields present.

$$SL_j = EL_j + \sum_{i=1}^{n} EL_i \quad (16)$$

The field designated EL is used to show field length and certain edit information. Note that Case 1 is longer than Case 2 for the same segment, on different records. It will also show the mode of the data:

Packed Decimal—Unsigned
Packed Decimal—Signed
Binary
Eight Bit etc.

All this information will be used by the File Interface Program to be discussed later.

Recall that each physical record will contain a Record Control Segment and one or more data segments. Within the Record Control Segment, Figure 8, Memory Map of Physical Record are fields to show the length (RCL) of this segment, the total length (RL) of the record data on the track, and the address of Auxiliary Physical Record(s). Figure 8 shows the record and some of the fields mentioned. It can be seen that:

$$RL = RCL + \sum_{j=1}^{n} SL_j \quad (17)$$

All Record Control Segments in the file system are managed the same way. After a record is read into memory, the start of the Record Control Segment will always be known. By looking at field N, the programming system can initialize to handle the number of segments possible in the subect file record. This is an aid to setting up "looping limits." Any segment can be located in the input area by referencing the positionally significant segment bit (SB_j) that pertains to the desired segment. This bit has two states:

Set = the segment is present in the record.
Reset = the segment is not in the record.

When the bit is found "set," reference is made to the associated Displacement Field (D_j) for the segment. This field contains the number of positions away from the start of the record where the segment is found. By consulting control fields in the segment any data can be secured.

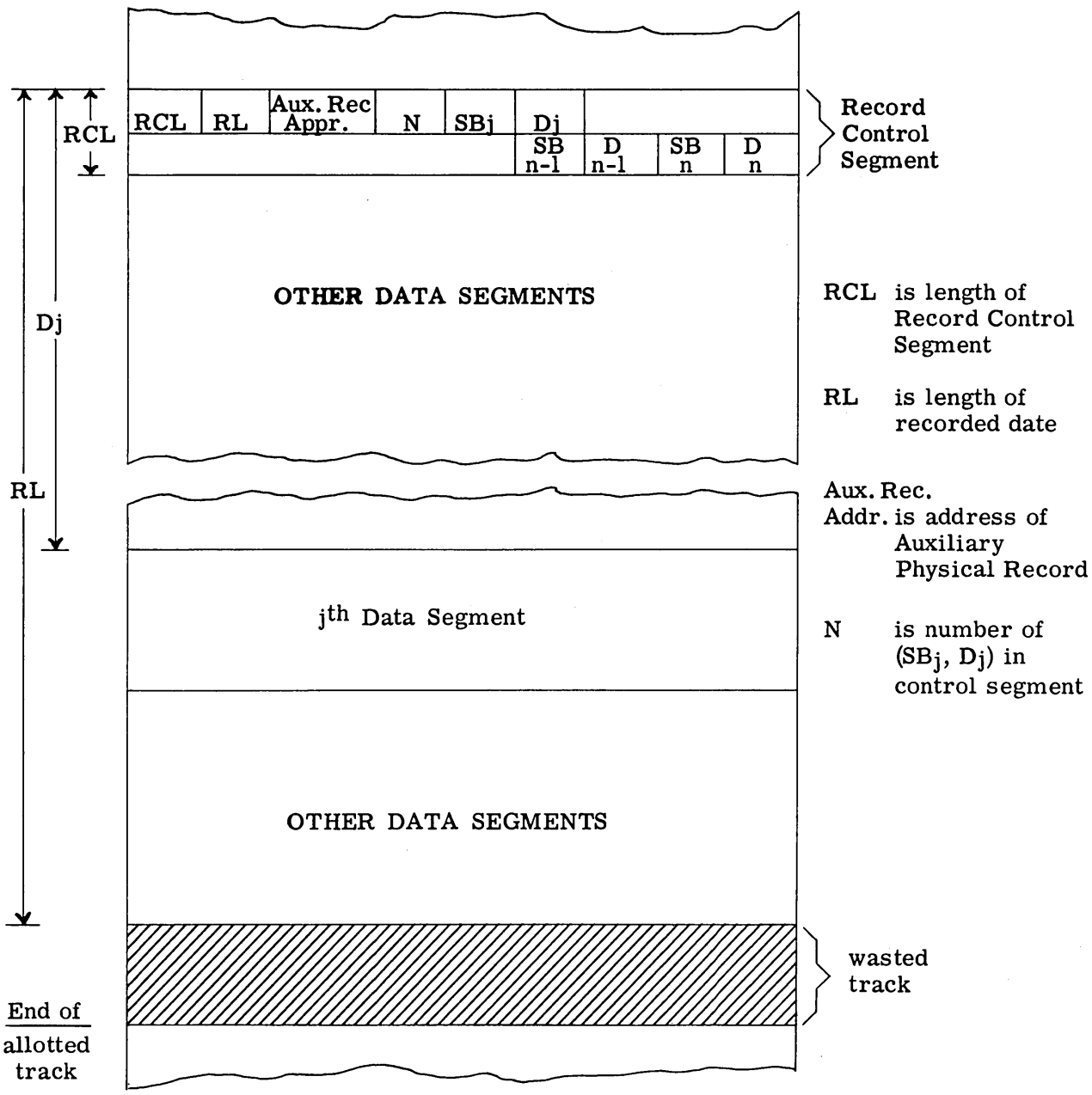

Figure 8—Memory map of physical record

In many business applications there is a constant need for studies of the file records. Frequently the records desired are not known specifically, but will meet certain criteria. This is the need for an associative capability. By placing the Record Control Segment in the Key Area of the record, the records can be located using a simple key scan program. This way the entire record does not need to be brought into memory. If the Record Control Segment is not placed in the Key Area, the entire record must be brought in, but the logic remains the same. By using the Segment Bits (SB_i) Figure 8, in a register, simple Boolean Statements will identify the desired records without lengthy special coding of "compares and branches."

There are many ways in which the contents of the data base in a MIS can be developed. Unless a disciplined approach is used,[15] subtle duplication or omission of data may result. Earlier, it was mentioned that the mortality of special purpose files is expected to be high. This is in part due to the changing needs of the business enterprise. New information must be entered into the data base. There is always a healthy concern for existing programs when the file is changed. By having the file record itself contain information about where data are located and making it relative, the impact of change is kept to a minimum.

Most programming systems will provide for the "linking" together of various subroutines. Since we can have a general way in which data can be extracted from a file record, it is natural to have a generalized routine written once. On the project where these techniques were used, the name used for this routine is File Interface Program (FIP). This program requires only the key to the desired record and a mask of the requested data. It is written in "tight" coding, which translates an efficient but complex file record into a simplified record for use of programs written in COBOL. It relieves the problem programs of all file reading and writing responsibility. Problem programs need no knowledge of specific locations of file data. As the system evolves, the value of FIP is expected to be even more apparent.

Application comments

Information about BIS is generally available[16] and has been discussed at several industry conferences. In the Pennsylvania Company, some of the phases of BIS have already been implemented. The techniques described here were developed to design the file for BIS. Earlier in 1967, the first real time aspects of the system were placed in service on a trial basis. The File Interface Program, written in assembler language, is serving the inquiry programs that retrieve data on customers' accounts for the use of Service Representatives. This inquiry takes place while the representatives are conversing with the customers, so there is a need for a fast response. The file system is list structured. Some data sets are "threaded." Others are based on the Multi-List System or on the Inverted List technique depending on the operational characteristics which are required. The management of the keys and the interior of the record makes this diversity possible. Presently, the file system contains only those segments which are required for the real time and batch processes active today. As additional segments enter the file due to new work to be done, FIP should preserve the investment in existing application programs.

These problem programs were written in COBOL to give a high degree of transferability. The data mode and organization of the file record is one that conserves storage, but is not suitable for direct use with COBOL. Therefore, the interface program edits and code translates the data to an eight bit mode. Initially, the file device used for the massive data base is the IBM 2321 Data Cell. When or if there is an increase in the use of the file beyond the performance of this device, the same records will be transferred to a faster device without serious reorganization. The record length will be chosen in accordance with the principles discussed, but there will not be changes to existing programs. In fact, due to shipping schedules, initial program testing was done on an IBM 2311 Disk using the same record discipline. An additional advantage of the optimizing techniques was in the form of providing input to the simulation models of the file system. The design loop was one of data collection, preliminary file design, simulate, analyze results, modify design, simulate, analyze, etc.

These techniques have served well. The massive files enjoy a Storage Utilization of 65%. The actual record length of the Main Physical Record is down to 796 bytes; from early designs, not using these methods, of 1800 bytes. A more favorable balance between file processes and application program time exists. This ratio is (1: 1.2). The informal objective of a mean response Time of 10 seconds for inquiries has been met. The distribution of response times is shown on Figure 9, Inquiry Response Times. This data is taken from a GPSS (General Purpose Systems Simulator) model that was used in the design effort. It has been validated by empirical measurements of reality and is deemed conformal.

Fundamental thinking still takes place in the whole problem of data management. The functions performed by FIP seem to lead to a solution using either:

1. **Macro Statements**—given in the application programs that would be assembled along with problem coding. An expanded GET. (Example: GET File Name, Record Key, Segment Name 1, Seg-

ment Name 2, Segment Name n.)

2. **Read Only Storage**— the coding of FIP could be placed here with obvious savings. In effect, a privileged command, or ROS MACRO

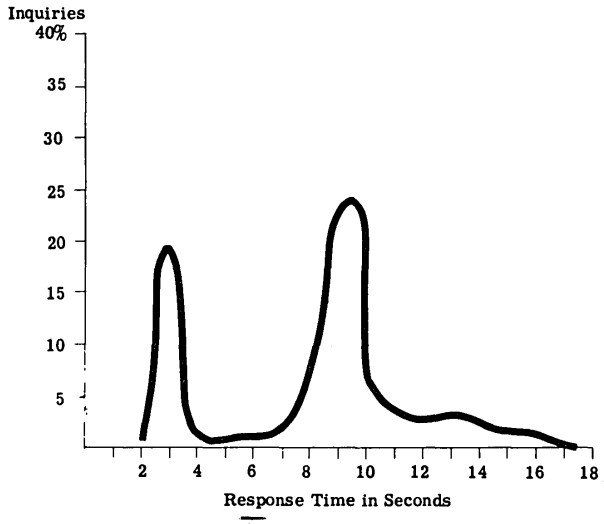

Figure 9—Inquiry response time

It is felt that File Spread Sheet Analysis or intuition would not have produced results as satisfactory as the current design. These techniques are the only ones that seem to properly consider data length, probability, and activity in determining the record design. Other schemes fail to consider them all, or treat some of them casually.

ACKNOWLEDGMENTS

The work described here represents the contribution of many other than the author. Much of the numerical analysis work was built on a solid base provided by Mr. George Pan, of the General Electric Company, Schenectady, N.Y. Messrs. Robert Lanham and Jack Lauback of IBM at Kinston, N.Y., contributed many ideas or participated in application of these techniques. Mr. Richard S. Kaufman of IBM at Philadelphia, Pa. provided the needed counsel for the associated simulation effort which was imbedded in much of this work. Mr. James Weisbecker of the Bell Telephone Company of Pennsylvania was responsible for the demanding application of the files and the development of FIP as a working program. This recognition can only understate the value of their individual talents to the work reported here.

REFERENCES

1 R V HEAD
Real-time business systems
Holt Rinehart & Winston Inc New York p 78, 79 1964

2 W H DESMONDE
Real-time data processing
Prentice-Hall Englewood Cliffs N J p 107 1964

3 C ALEXANDER
Notes on the synthesis of form
Harvard University Press Cambridge Mass p 1-5 1964

4 G J SIMMONS
Application of an associatively addressed distributive memory
AFIPS Proc vol 25 1964

5 E W FRANKS
A data management system for time-shared file processing etc
AFIPS Proc vol 28 1966

6 G S PAN
Linear programming method for optimum file design
Unpublished Master's Thesis Syracuse University N Y 1965

7 F E CROXTON D J COWDEN
Applied general statistics
Prentice-Hall Englewood Cliffs N J p 302-303 1955

8 P M MORSE
Queues, inventories and maintenance
John Wiley & Sons New York N Y p 9 1958

9 W W PETERSON
Addressing for random address storage
IBM Journal of Research and Development vol 1 no 2 1957

10 W P HEISING
Note on random addressing techniques
IBM Systems Journal vol two 1963

11 *The weighted record analysis program users guide*
Un-numbered Communications Industry Marketing IBM Corp White Plains N Y 1966

12 *Introduction to IBM System/360 direct access devices and organization methods*
C 20-1649-0 IBM Corp White Plains N Y 1966

13 *Random access devices series references manual*
70-06-500 RCA Camden N J March 1966

14 *Series 200 summary description—file storage units NP—7099*
Honeywell Inc Wellesly Hills Mass 1965

15 J C MILLER
Conceptual models for determining information requirements
AFIPS Proc vol 25 1964

16 *Business information systems*
American Telephone and Telegraph Company New York N Y 1965