Information Systems

Application Manual

Integrated Data Store

V. PAUL COLALUCA
MANAGER - DATA CENTER
EASTERN REGION

$\int Idt$

A New Concept In Data Management

GENERAL ELECTRIC

AS-CPB-483A (REV. 7-67)

INTEGRATED DATA STORE

A NEW CONCEPT IN DATA MANAGEMENT

APPLICATION MANUAL

July 1967

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

AS-CPB-483A

(2.4M 7-67)

CONTENTS

APPENDIXES

INTEGRATED DATA STORE -- A NEW CONCEPT IN DATA MANAGEMENT

I. INTRODUCTION

I-D-S (Integrated Data Store) is a unique new way of organizing business data. Using disc storage units, I-D-S is a technique for describing the complex information structure of a business, a way to manipulate data in conformance with that structure, and a computer processing language to implement the design.

The use of I-D-S sharply reduces the high systems and programming costs associated with implementing an integrated business system. Business management has long recognized the need for current, accurate, and immediately available information. However, the long lead-time necessary to design a system that would take into consideration all the complexities of the problem was prohibitive. With I-D-S total integration can be attained in much less time than was previously thought possible because I-D-S facilitates good system design. The graphic technique illuminates information structures and points out opportunities for optimization. And, the language serves to translate that system structure into efficient computer programs with a minimum of clerical effort on the user's part.

II. INTEGRATED SYSTEMS DESIGN

The design of integrated data systems is highly complex. The conventional system is organized function-by-function, each with a bulky file. Mandatory cross-referencing to show interaction between functional operations becomes highly intricate and, in many cases, necessitates carrying redundant information within the files. Many computer runs are necessary to process these files. The results must be coordinated and correlated. The concept of Integrated Data Store significantly simplifies the design of integrated systems by providing a single file organized around the data rather than around the function. A comparison between the conventional approach and the I-D-S approach to systems design illustrates the benefits of the latter.

When designing an information system, many questions must be resolved. Some of these are:

** What information must be stored in the system?
** How should this information be stored (volume, sequence, accessibility, etc.)?
** What cross references must exist between information in the various files?
** How will information be retrieved in response to information processing requirements?

A. Conventional File Organization

In conventional approach to file organization, the problem of designing a purchasing information system, for example, may involve the planning of three files: a vendor file with related open purchase orders for general processing; a purchase order file for expediting purposes; a file by inventory item number for production inquires. Fig. 1 illustrates this organization using disc storage.

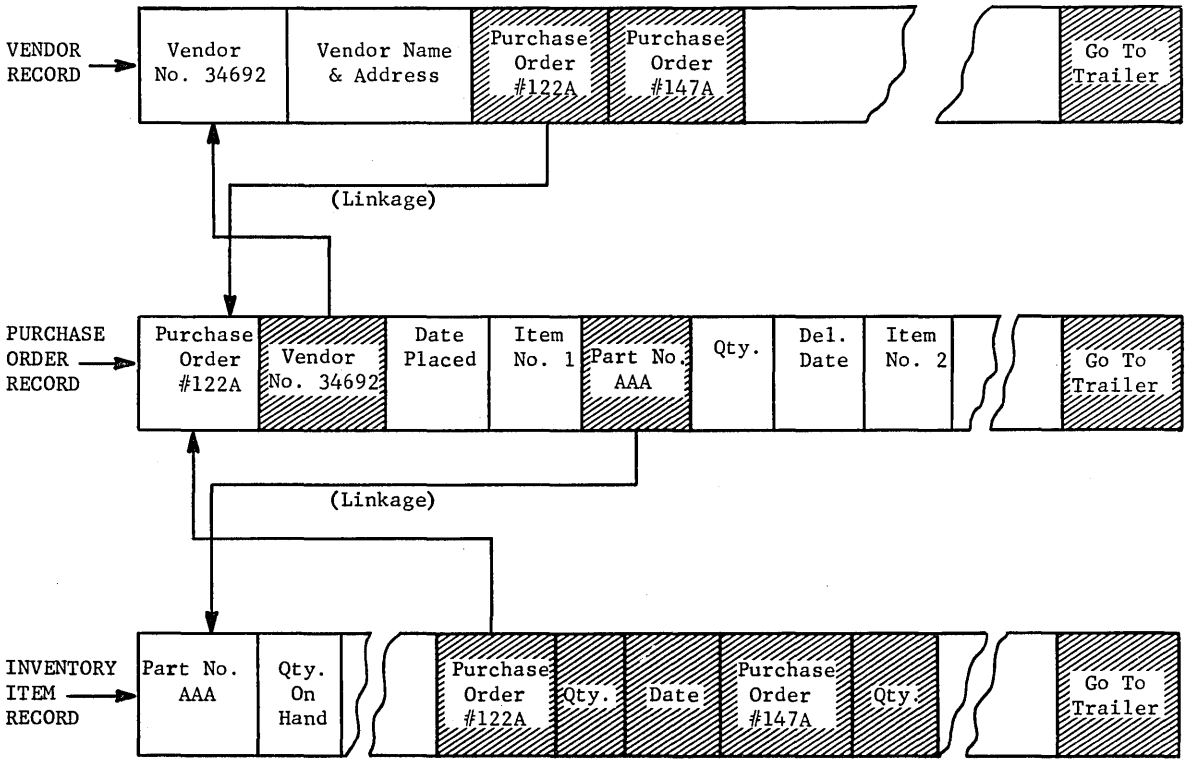


Figure 1. Conventional Record Formats

The shaded fields indicate redundant data in the series of records shown. These records are stored in separate areas of the file unit with trailers in separate overflow areas (see Fig. 2). (Overflow areas are an extension of the normal file area. They are used to store, for example, the records with the duplicate disc addresses often created when using randomizing techniques for record storage).

The systems design and programming effort involved in designing the file layout, record content, and data associations, to take full advantage of the hardware capabilities, is both complex and lengthy.

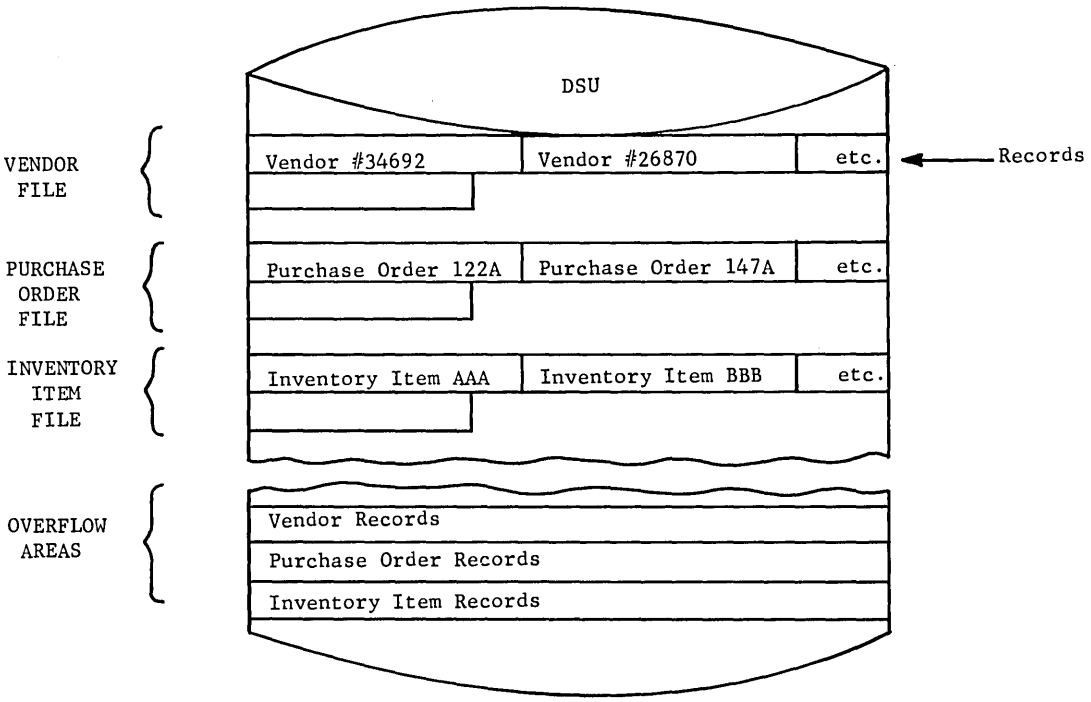


Figure 2. DSU Layout--Conventional

Also, the processing and maintenance of stored records is cumbersome and time-consuming. Related information is packed in different sections of the file unit, and records must be individually accessed in each location to complete the maintenance function. (See Fig. 3).

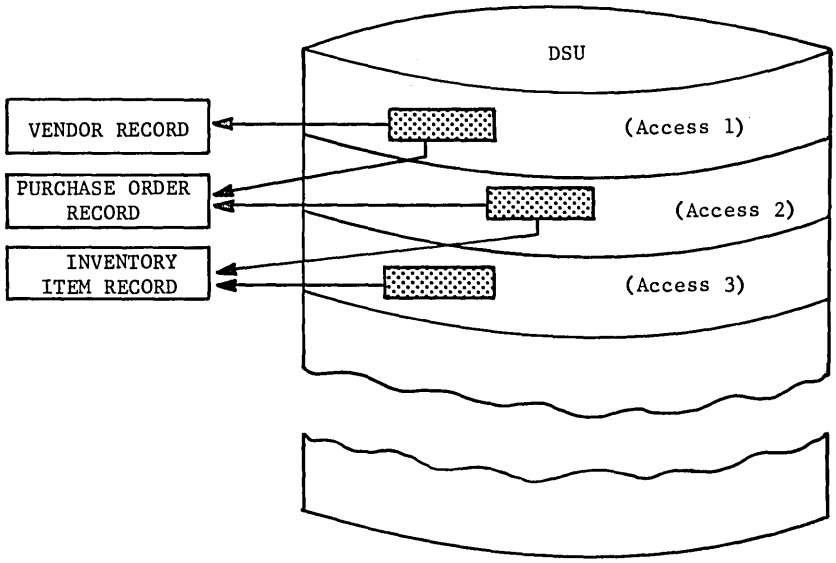


Figure 3. Record Access

If there are several hundred or several thousand open purchase orders at any one time, and the activity (changes, new orders, receipts, etc.) is high, then maintaining these files requires considerable processing time.

B. I-D-S File Organization

I-D-S alleviates the systems programming and storage problems inherent in the conventional approach to data organization and facilitates the subsequent processing of this data.

Fig. 4 shows the I-D-S record constructions comparable to the examples of conventional file organization.

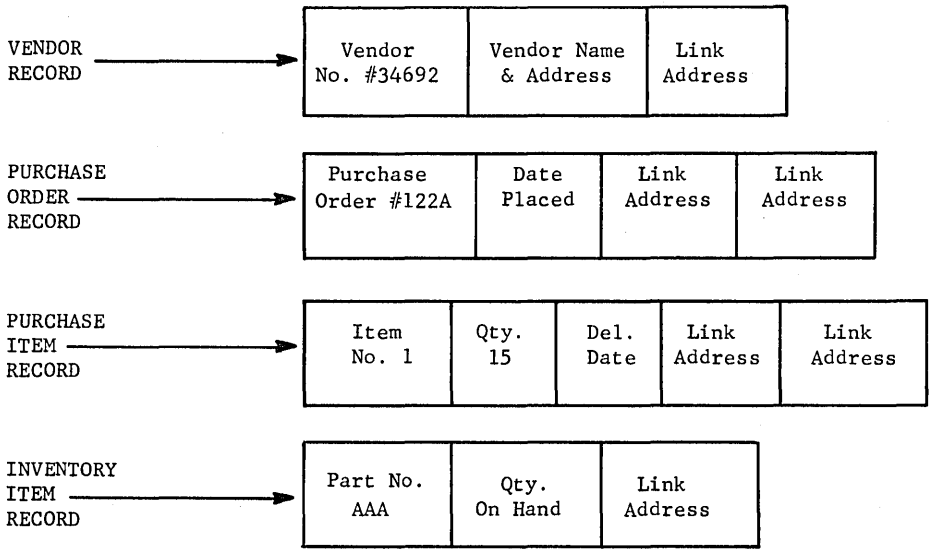


Figure 4. I-D-S Record Formats

You can see that there is no redundant information in the records. Therefore, the purchase order record, to convey its full meaning must be properly associated with (1) the vendor record which describes the vendor with whom the order was placed, (2) the item records which describe the quantity ordered and the delivery date, and (3) the inventory records which describe the items being purchased.

C. Associating Records Into Chains

The chaining feature is the fundamental structuring tool of I-D-S. Chains are made up of all the information about a particular function (Fig. 5 gives an example). A chain must contain one master record (the vendor record) and can contain any number of detail records (other data directly related to the vendor). All the interlocking relationships of pertinent information are cross-referenced by the chains. A master record in one chain may be a detail record

in another. There is no redundancy in the storing, processing, or maintaining of data. Fig. 5 illustrates the chaining network of logical records as they might appear in an information system. With the information system structured in this manner, file interrogation and processing is conveniently simplified.

In examining the records and chaining associations in Fig. 5 you will see that there are four types of records: Vendor, Purchase Order, Item and Inventory. To provide meaning to the system, these logical records are associated in chains:

All the purchase order records -- for a vendor
All the item records -- for a purchase order
All the order item records -- for an inventory item

There can be a variable number of records within each chain. These records can be linked into any number of chains. An information system organized in this way allows all functions of a business to interrogate records and data stored only ONCE in I-D-S. During each disc access, a block of information is transferred to memory. You can store related records within the same block on the disc storage unit. With proper data organization, the probability is high that with a single access to the disc for a particular record, most of the related information will also be available in the block.

With this feature in mind, and by using Fig. 5, it is easy to see how the purchasing function of the business can obtain all information pertinent to vendor status. For example, if information is requested for vendor number 34692, the processing sequence of Fig. 5 is:

1. Retrieve vendor record (vendor code is 34692).

The basic vendor data is now available in memory for processing. (In each vendor record there is a chain link to the first purchase order record).

2. Retrieve next purchase order record in purchase order chain.

Process data of purchase order number 122A. (In each purchase order record there is a chain link to the first item record in the item chain and a link to the next purchase order record in the purchase order chain.)

3. Retrieve next item record in item chain.

Process data of item number 1. (The last item record contains a chain link back to the purchase order chain.)

4. Processing of purchase order 147A with its item records 1, 2, and 3 occurs in the same manner.

5. Following processing of purchase order number 207A, the purchase order chain links back to the vendor record, which completes the cycle for this vendor.

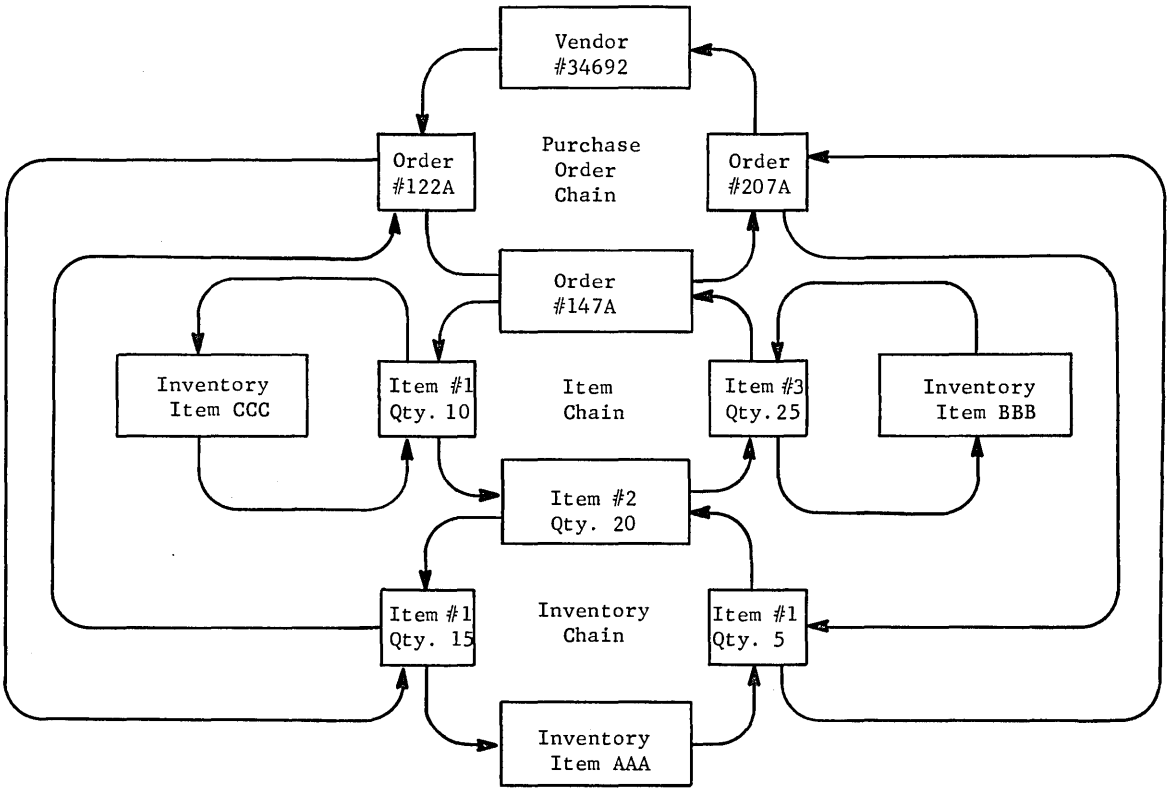


Figure 5. Chaining Example
(Schematic Diagram)

Composite vendor information is stored and retrieved through these chains, which provide a meaningful association of related data. Redundancy is eliminated. For example, vendor number is carried only in the vendor record and associated with its purchase orders through chains.

Flexibility of the I-D-S system organization can best be illustrated by looking at a second method of processing the records in Fig. 5. The production control functions of the business might, for example, inquire as to the inventory status of item AAA, both on hand and scheduled on order. The processing sequence is as follows:

1. Retrieve inventory item (inventory item number is AAA).

Process the inventory on hand balance. (Each inventory record contains a chain link to the first item in the inventory on order chain).

2. Retrieve next record in the inventory on-order chain.

Process item number 1 (quantity 5). (NOTE: The item record is in two chains -- the inventory on order chain and the item chain).

3. Repeat step 2 until the last item record in the inventory on order chain is processed and the chain terminates back, at inventory item AAA.

When the inventory status indicates the need for vendor expediting of an order item, this can be done simultaneously with the above processing. This is possible because the items in the inventory on-order chain are linked into the order chain and into the purchase order chain. By traversing these chains, all necessary purchase order and vendor data can be obtained.

III. I-D-S ENVIRONMENT

The Integrated Data Store is essentially independent of both the processor and storage media. A general knowledge of a disc storage unit is helpful in understanding the over all organization and concept of I-D-S.

A. I-D-S Pages

In the I-D-S system, a page (or block) of records consists of a fixed number of characters assigned by the program implementor. A page may contain any combination of logical record types. Each record type has its own specific length defined by the implementor. Related record types may be associated and linked according to their data content and may be stored within the same page. Automatic packing of these records within the page fully utilizes the space. Whole pages are read into memory during processing. Therefore, with proper data organization, the probability is high that much of the related information will be available in a single file access. Fig. 6 is an example of an I-D-S page.

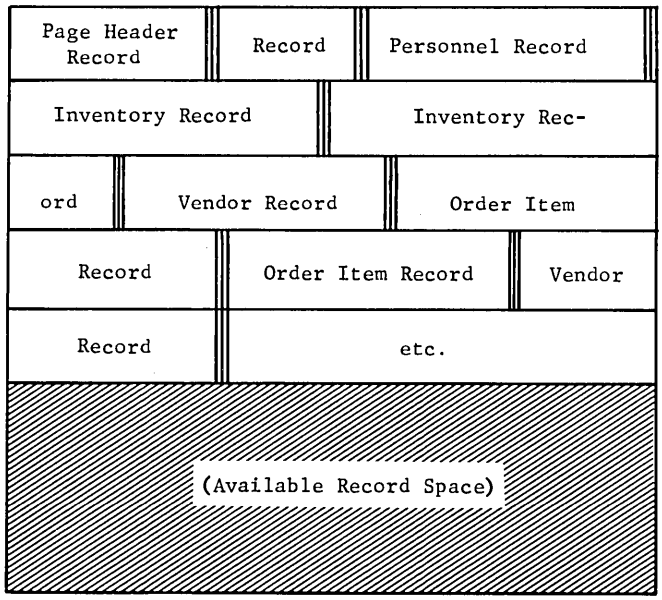


Figure 6. I-D-S Page

The organization of the page facilitates file maintenance and processing, while making it possible to place a variable number of record types in any given page. Every page begins with a unique Page Header record which contains several control fields used by the system, as follows:

1. Reference address of the page (page number).

2. Space available in the page for additional records.

3. I/O control, indicating whether page has been altered since retrieval.

4. Chain field, indicating address of the first record of a chain of calculated records, all of which are randomized to this page.

5. Line numbers available for assignment within the page (a line is equivalent to a data record).

Data Records within the page contain the following control fields in addition to their user-specified data fields:

1. Line number (1 character space).
2. Record type (2 character space).
3. Record length (2 character space).
4. Chain fields (4 character space per chain field).

B. Reference Code

The reference code is a relative addressing code permanently assigned to each record. It can be considered to be a logical address rather than an address that defines where the record is physically located in the file. Thus, the reference code can be used for direct record retrieval on a long-term basis.

The reference code consists of the page number and the line number of a data record as explained below:

1. The page number portion, (a sequential number permanently assigned to each page) defines what proportion of the way through the I-D-S environment the page is stored. Each record stored within a page shares the same page number as part of its reference code. Page numbers are converted to actual disc addresses by the I-D-S mapping routine at execution time. The page number portion occupies three character positions and permits 262,143 pages per I-D-S file.

2. The line number portion identifies a record within a page. As records are stored in a page, an available line number (line numbers are reused as records are deleted) is assigned to the record and is combined with the page number to complete the reference code. The line number portion occupies one character and permits 63 lines per page.

IV. DATA ORGANIZATION

Data organization refers to the inter-record relationships established within I-D-S. The record is the basic unit of data. Record association is achieved through chains which provide cross-reference linkages between records. These chains provide the integrating force which is implied in the name Integrated Data Store.

As shown below, the I-D-S record contains a set of data fields which collectively describe the event, activity, status, or plan that the record represents. I-D-S augments these records with additional fields called identification and chain fields, as shown in Fig. 7.

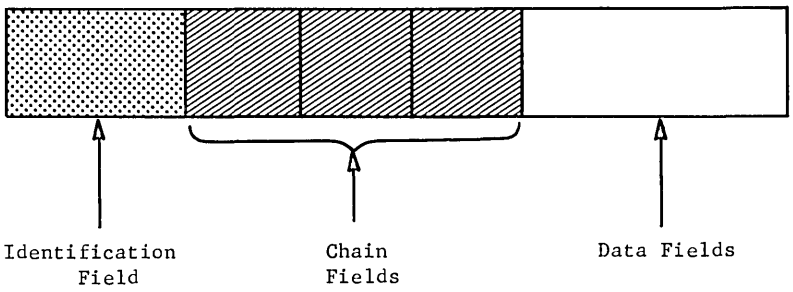


Figure 7. I-D-S Record

The chain fields contain pointers to other I-D-S records. They point from one record to the next, creating a circular association of records (Fig. 8).

These associations are processed according to the data descriptions and the procedural statements executed. The arrows in Fig. 8 indicate the linking actually carried out through the storing of the pointer of one record in the body of the prior record.

A. Data Records and Fields

I-D-S records are fixed-format, fixed-length in the COBOL tradition; that is, the length and format of a specific type of record, such as payroll or inventory record, are fixed by the specifications of the systems designer. Records of many different types, each with its own length and format may be used in the system. To maintain control, each record has the same identification fields at the very beginning. These fields are: (1) line number portion of the reference code, (2) record type (such as inventory, payroll, etc.), and(3) record length. The rest of the record consists of data and chain fields to suit the application requirements.

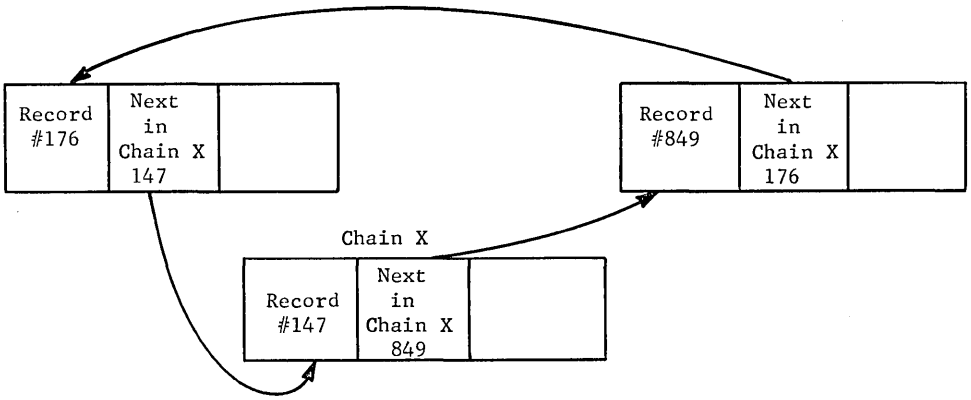


Figure 8. Chain Association

Records may have any number of data fields, each defined as some number of decimal, alphabetic, or alphanumeric characters. Fields may vary in size from one character up to many characters, as for a drawing, or a part number, or an employee's name. These fields are specified by the systems designer.

Chain fields contain the reference code for addressing. They are defined for each chain in which a record participates. Experience indicates that the average record is in only two chains. An occasional pivotal record in an integrated system may be in six or eight chains. There is no upper limit on the number of data or chain fields except that provided by the maximum page size. Average record size has been 30-40 characters in total length, with an occasional record of 120-180 characters.

I-D-S records are stored only once. Conventional approaches to file organization often require records (or certain fields in the records) to be repeated in several files. With the ability to link records into any number of chains (as required by the system), the same data fields are available for all chains processed. This technique of eliminating redundant data has four important advantages:

1. Additional space required for duplicate records is eliminated, resulting in a reduction in the total capacity required.

2. The work of data maintenance is greatly reduced, as there is only one record to retrieve and modify.

3. The possibility that one of the copies of a record will not be properly modified is eliminated. Since there is only one copy of a record, any incorrect information will be quickly recognized and corrected.

4. All reports drawn from the file will be consistent, since there is only one set of facts (records).

B. I-D-S Chains

Fig. 9 shows an I-D-S chain. Its characteristics are:

** It consists of one master record and any number of detail records.

** It links records together in a circular fashion.

** It relates records in meaningful sequences.

Records must also be defined as to their relationship within a chain--master or detail records. These record relationships are specified in the Data Description when the chain is defined.

This master-detail relationship of records is analogous to the conventional header-trailer file sequence of records. The master record of the chain contains the fixed information (header type) pertaining to the variable number of detail records in the chain. The detail records in the chain contain the variable information.

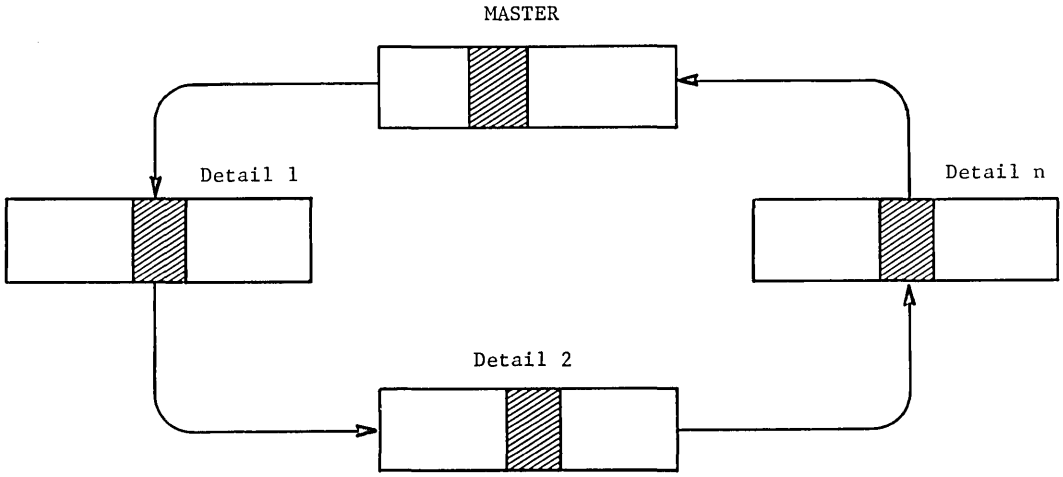


Figure 9. I-D-S Chain

Chains exist for two separate but closely related reasons. First is the case where a portion of the information appears in multiples. For example, a personnel record with a variable number of deductions and work experiences. This type of information is easily structured by building a personnel master record. Two chains are created containing the personnel record as the master record. As many deduction records as necessary are linked into the deduction chain as details. Work experience for the employee involved is handled in a like manner. Both chains are now linked to the same master record as shown in Fig. 10.

The second case for chain structuring of information involves the association of several related units of information. Relating all the purchase orders for a given vendor is an example. Fig. 5 (page 6) illustrates this example. The purchase order chain associates all of the purchase order records with their vendor record.

I-D-S chains have several structural aspects which should be emphasized:

** A chain type is named with a symbolic name. There will be as many chains of the chain type as there are master records for that chain type.

** Each chain has only one master record. The record type of the master record is the same for all chains of the same type.

** A chain is created whenever its master record is stored in the I-D-S. The chain ceases to exist when the master record is deleted.

A chain may contain more than one type of detail record. Any number of detail records may be in a chain.

Whenever a master record is deleted, all of its detail records are automatically deleted (and their detail records too).

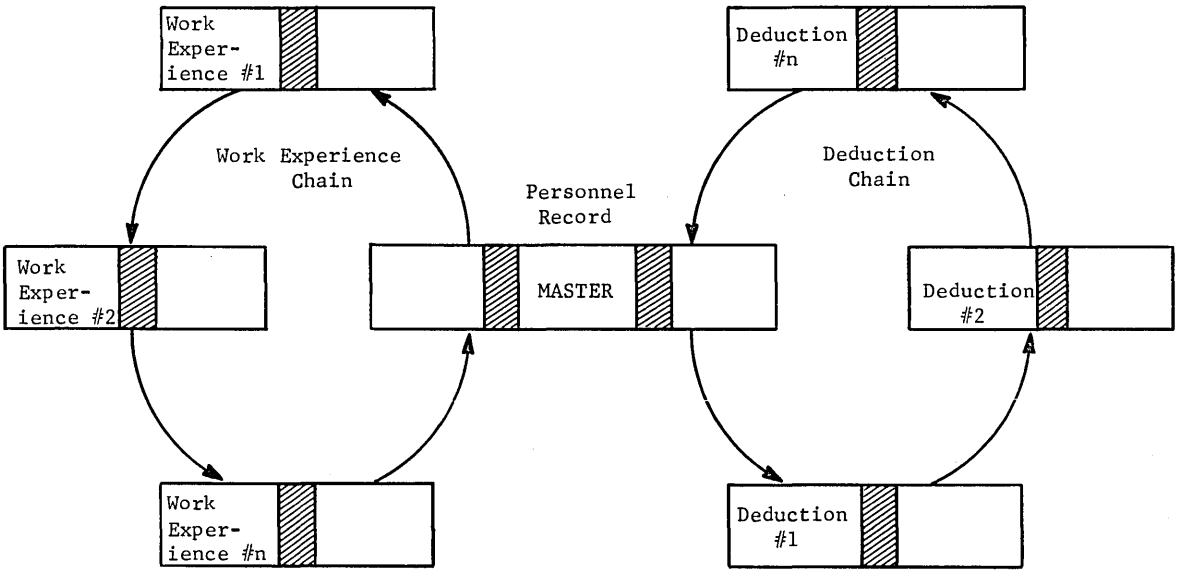


Figure 10. Master Record of Two Chains

All records in a chain are associated in an endless loop with the last detail chained back to the master.

The master record of the chain stores the reference code of the first detail in the chain.

A record (master or detail) may be defined to be in as many chains as are required. It may be defined as master in one chain and detail in another.

As records are stored in the system, they are automatically linked into their defined chains.

When a record is deleted, the chains in which it is a detail record are automatically adjusted to relink around the deleted record.

C. Chain Fields

Chain fields are provided for each chain in which a record participates. Chain fields are used by I-D-S for storage and retrieval purposes. Chain fields are the links of the chain. They provide the paths that I-D-S uses in operating on the data.

To this point all chaining illustrations have shown one type of linking.

There can be up to three chain fields per chain. There must be at least one and can be as many as three. These fields are named NEXT, PRIOR, and MASTER.

The chain field NEXT always appears in a record for each chain in which the record participates.

There may be a PRIOR chain field in every record (in addition to the chain field NEXT) in the chain. The chain field PRIOR is included in the record when the chain is defined to be PRIOR processable.

Chains that are defined as PRIOR processable can be traversed in either direction. Note that all figures up to now have shown chains with their arrows pointing in one direction. A chain that has been defined as PRIOR processable would appear as in Figure 11.

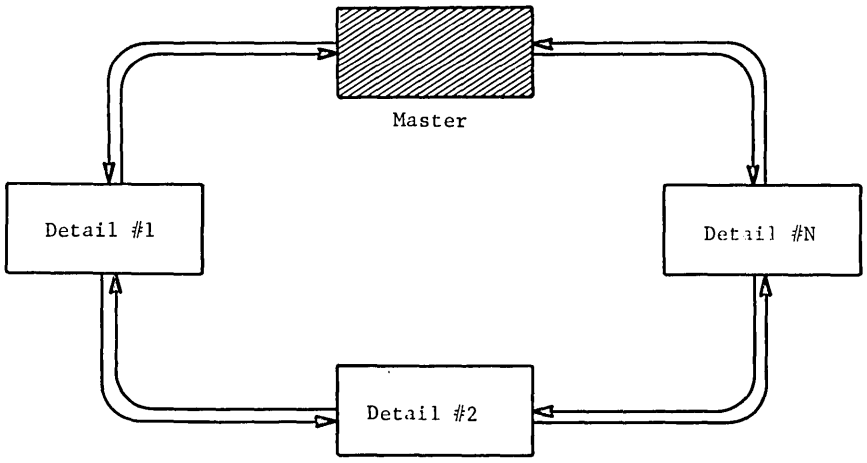


Figure 11. PRIOR Processable Chain

There is a third chain field that may be called for in the chain description. This one links a detail record directly to its master. When all details in a chain contain this chain field, then the chain is defined as HEADED. (See Fig. 12).

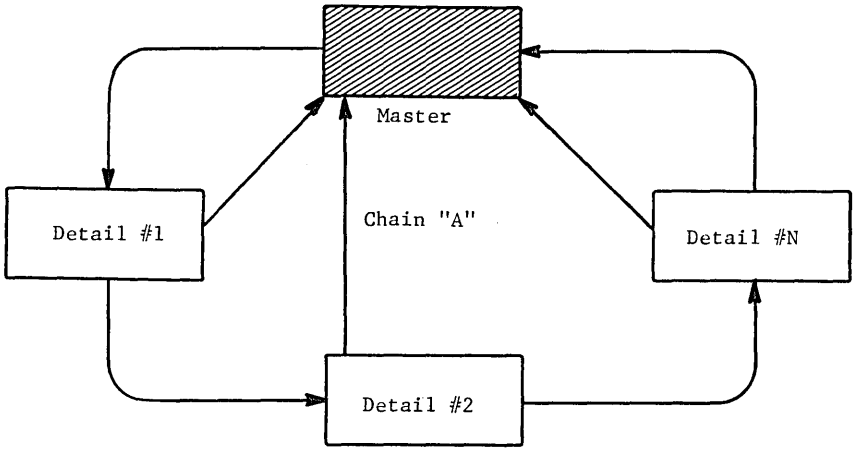


Figure 12. HEADED Chain (Linked to Master)

D. Chain Tables

Chain Tables are used internally by I-D-S. A chain table is built for each chain defined. The knowledge of what they are and how I-D-S uses them can be of help to the programmer.

A chain table contains four entries: MASTER, PRIOR, CURRENT and NEXT. To illustrate, let us consider a chain consisting of four records (Chain A) as shown in Figure 13.

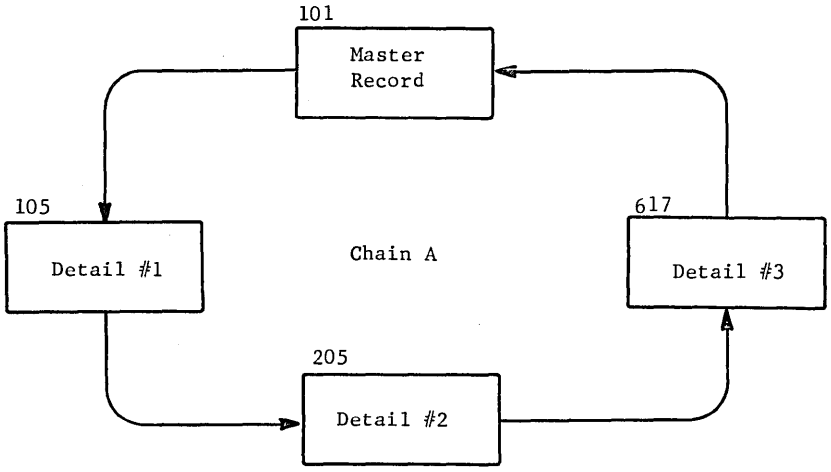


Figure 13. Chain A

Assume the master record of Chain A has just been retrieved; the contents of the chain table are shown in Figure 14.

Chain A Table	
MASTER	101
PRIOR	000
CURRENT	101
NEXT	105

Figure 14.

When I-D-S is to RETRIEVE NEXT RECORD of chain A, it uses the reference code in the "next" entry of the chain A table to determine which record to retrieve.

Following the retrieval of record #105, the table will appear as shown in Fig. 15.

Chain A Table	
MASTER	101
PRIOR	101
CURRENT	105
NEXT	205

Figure 15.

If a RETRIEVE PRIOR of chain A were executed at this point, I-D-S would use the "prior" entry in the chain table.

For chain table updating details, see Appendix C.

E. Inserting A New Detail Record In A Chain

Two steps are required to insert a new detail record in a chain:

- The appropriate master and its chain must be selected.
- The record must be inserted in that chain according to the chain ordering rules.

F. Master Record Selection

Whenever a record is stored as a detail, the chain into which it will be linked must be found. If there is only one such chain and master record, then the problem is easily solved. However, if there are many such chains with their master records, then criteria must be established for selecting the correct one. The correct master record may be determined in two distinct ways:

- Based upon the chain table--Select Current Master--This rule uses the master of the current entry as the master record of the new detail.

- Based upon the retrieval of a master record as defined by the match key fields of the chains--Select Unique Master--This rule uses the identification criteria established in the data definition, and the data values currently stored in the match key field of Working-Storage.

G. Chain Ordering

All chains in the Integrated Data Store system are ordered in one of six methods. A method is specified by the systems designer in the chain-order clause.

The chain-order clause must be used in each Master Chain Definition entry.

The six options are:

1. Chain-order Sorted Within Type--Records of the chain are maintained in sort key sequence within record type.

2. Chain-order Sorted--The various types of records of the chain are maintained in a single sequence regardless of the number of record types in the chain. Control fields of the various records must be of identical size and class.

   NOTE: When either of the sorted options is specified, details are inserted into the chain based upon the content of the defined sort control fields of the detail records.

3. Chain-order First--Inserts the detail as the first detail record in the chain relative to the master record.

4. Chain-order Last--Inserts the detail as the last detail record in the chain relative to the master record.

5. Chain-order Before--Inserts the detail record just before the current record in the chain. Used only in conjunction with the "Current Master" selection rule.

6. Chain-order After--Inserts the detail record just after the current record of the chain. Used only in conjunction with the "Current Master" selection rule.

V. INPUT/OUTPUT CONTROLLER

The major function of the Input/Output Controller is to manage the flow of pages of records between main memory and mass storage in response to STORE, RETRIEVE, MODIFY, and DELETE statements. It also controls the page processing function within memory, for example, finds the relevant record in the page, and (if so instructed) moves it to Working Storage. Figure 16 shows a schematic of these functions.

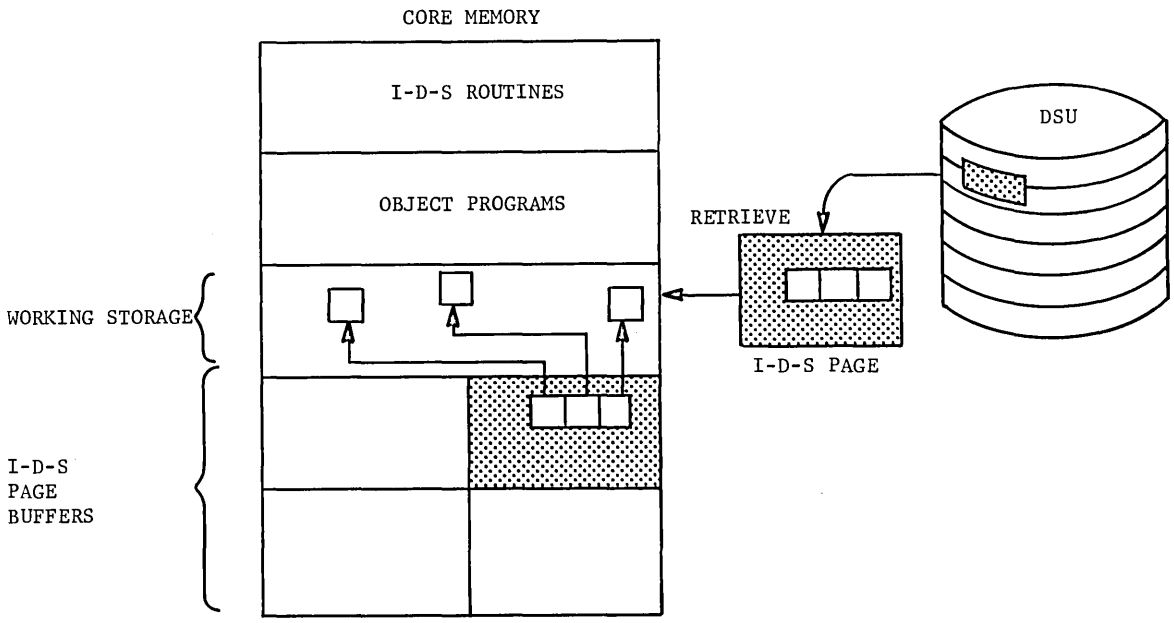


Figure 16. Input/Output Controller

A. Data Buffer Management

An inventory of data pages is maintained in memory to minimize the mass storage seek and transfer time. These pages are stored in numerous buffers in memory. The number of buffers depends on the amount of space available after the I-D-S subroutines and the problem-solving routines have been loaded.

The greater the number of data pages stored in core memory, the greater the possibility that the one needed next will already be there. To further improve the possibility of finding the page wanted in memory, the Input/Output Controller keeps track of the sequence of page utilization (record activity) and holds the most recently active pages in memory. Pages which are infrequently accessed are retired from memory as others are called in. The Input/Output Controller notes which pages have been modified and writes only the modified pages back to mass storage.

Each time a new page is brought into memory, its number is placed at the head of a page list. If a page already in memory is reused, it moves to the head of the list. Thus, this list tends to hold the most frequently used pages at the top of the list and the pages with little or no recent use at the bottom of the list. Page space is maintained to allow for the input of a new page. Following input, the page at the bottom of the new page list is automatically written back to the disc storage file (providing there has been activity updating that page).

B. Record Unpacking

The Input/Output Controller locates the record called for in the page. The fields from the record will be unpacked into working storage in response to a MOVE TO WORKING-STORAGE statement.

VI. RECORD STORAGE CONSIDERATIONS

I-D-S permits the user considerable freedom to control the record storage process and thereby minimize subsequent retrieval time. These aspects of I-D-S have been provided because of the pseudo-random access nature of all known disc, drums or other mass storage units. All of these aspects of the I-D-S language are directed toward achieving higher processing speed.

One means of achieving this result is the physical clustering of records that have a high potential of sequential retrieval. Several statements in the record definition area, such as "PLACE NEAR...." and "PAGE RANGE IS...." are provided to assist the file designer.

The PLACE NEAR statement stores a detail record in the same page as one of its master records, or an adjacent page if the master's page is full. Then the records in the chain can be retrieved with minimum access to mass storage. The PAGE RANGE controls the storage of a given record type over a limited portion of the file. This enhances the retrieval performance when a serial type search of the file is anticipated and also eliminates that record's interference with plans to cluster other records in a different PAGE RANGE.

It is often desirable to store records in control field sequence for optimum batch updating. Where batch updating is planned and the activity rate is high, the serial file structure will reduce total update time. The serial storage of records does not limit their ability to be integrated into other parts of the data base for random or chain processing.

VII. DATA STRUCTURE DIAGRAMS

A special graphic technique is used to display records and their master-detail (chain) relationships. Its use is particularly important in developing an overall view when planning an information system. This technique uses a block shape to designate a record

type--employee (record type 1) and deduction (record type 2)--and an arrow connecting two blocks to designate a chain. The arrow points from the master to the detail, as shown in Fig. 17.

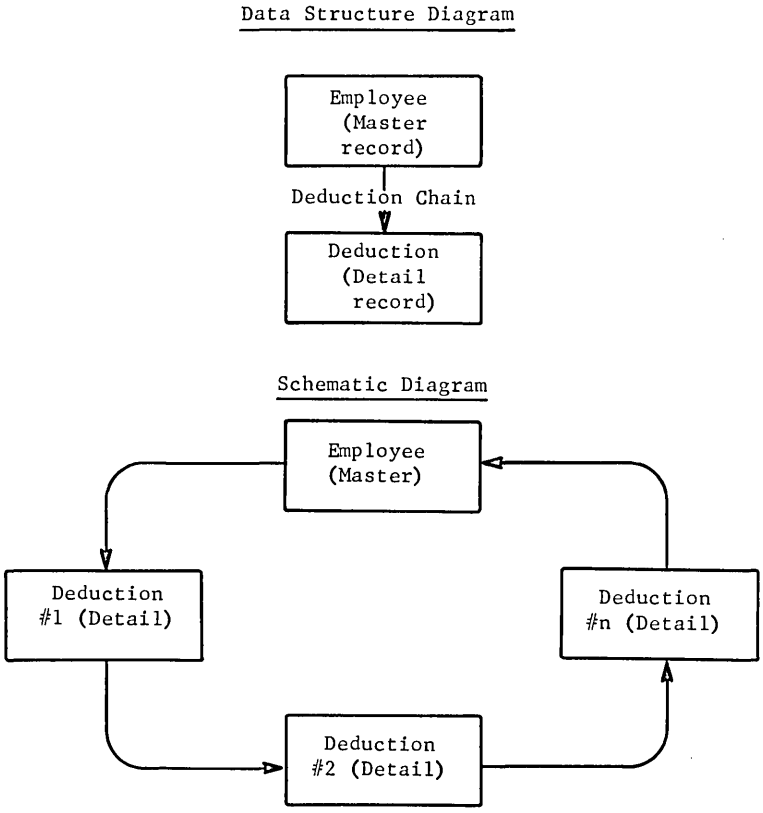


Figure 17.

In the foregoing example, the vertical block-arrow-block sequence carries the following message:

1. There are a number of records in the system of the master type (one for each employee).
2. Each of these records is the master of a chain of the specified type (deduction).
3. There are a number of records of the detail type (deductions 1, 2, 3, 4, etc.) in each such chain.

The purchase order data structure diagram (Fig. 18) represents the example shown in Fig. 5, using this technique.

Data Structure Diagram

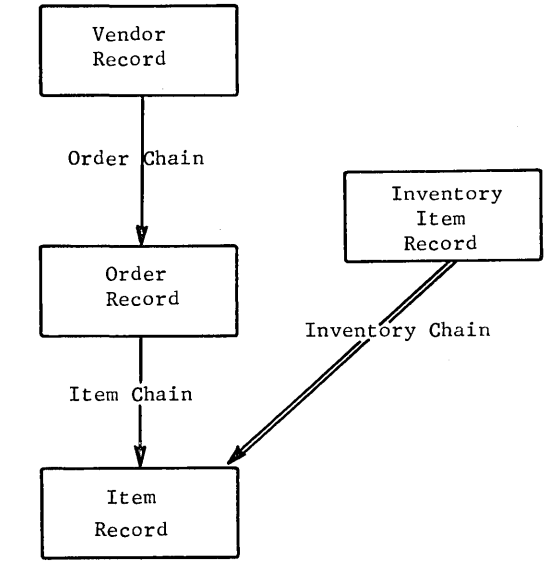


Figure 18.

The purchase order contains four groups of information.

1. Information about the vendor--such as his name, address, and vendor code.

2. Information about the order--such as the order number, due date, mode of transportation, and dollar value.

3. Information about the order item--such as delivery date, quantity, unit price, and extended dollar value.

4. Information about the inventory item--such as its identification and description.

Fig. 18 shows all four groups and their chain associations with only four blocks and three arrows. To expand this structure, four different record types would be designed to carry the information contained in the four groups:

Vendor record--There would be a vendor record for every vendor with whom the business is concerned:

1. It would be the master record of a order chain.

2. Thus, the vendor record is only a master.

Order record--There would be an order record for each order currently stored in the system:

1. It would be a detail in a order chain.

2. Each order, in turn, would be the master of an item chain.

3. Thus, the purchase order record is both a master and a detail.

Item record--There would be an item record for each item on each order.

1. It would be a detail in the inventory chain.

2. It would be a detail in the item chain.

3. Thus, the order item record is a detail in two chains.

Inventory item record--There would be an inventory item record for each inventory item currently stored in the system.

1. It would be the master record of the inventory chain.

The schematic diagram for the above records is shown in Fig. 5.

Additional information may be encoded in the data structure diagram by differentiating two types of chains with respect to storage criteria. A double line may be used to define chains where the master record controls the storage process and high-speed, in-core chain processing can be anticipated. A single line defines chains that do not control storage. In Fig. 18, the Item Record storage is controlled by the position of its selected Inventory Item record, the master record of its Inventory chain. (See Page 50 for Sample Coding of this structure).

It is possible, and frequently advantageous, to describe a chain with two or more different types of detail records. (Remember: a chain may have only one type of master). Fig. 19 illustrates such a case.

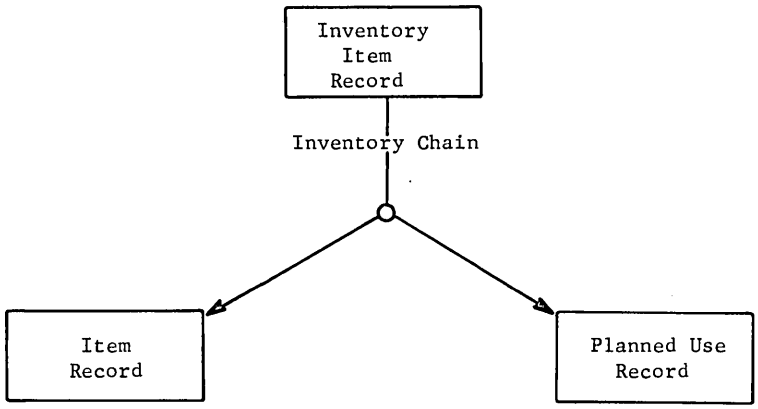


Figure 19.

The Inventory Item Record and the Item Record are the same as the example in Fig. 18. However, an additional detail record type, the Planned Use Record, has been added to the Inventory Chain. The split arrow is the means of showing such a chain.

Fig. 20 illustrates another frequently used data structure.

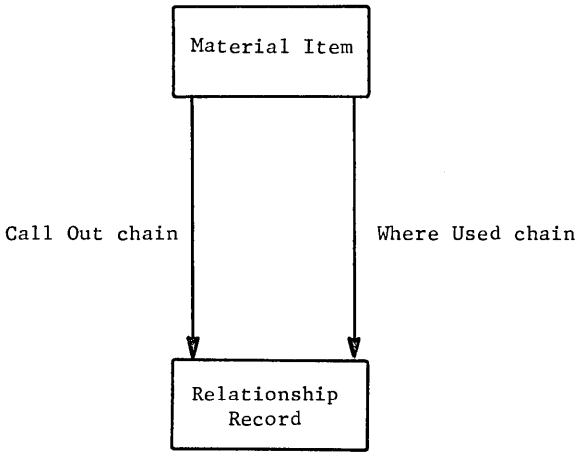


Figure 20.

This same structure (a pair of records joined by a pair of chains) is common to all file organizations where two records of the same type must be related to each other, for example, manufacturing parts lists, military organizations, PERT and CPM networks, geneological files, inverted dictionaries, etc.

This Material Item Record/Relationship Record structure can be used to define an engineering parts list assembly breakdown. A given material item is recorded only once, regardless of the number of assemblies (which are also material items) on which it is specified.

Fig. 21 illustrates another information structure. Two records, both of which have been defined as details in one chain (account chain), also have a master/detail relationship between each other which is defined by another chain. (Joint account chain). This example illustrates the direct association of each account record with a Bank Customer Record. The Joint Account Record is used to associate the Account Record with additional Bank Customer Records.

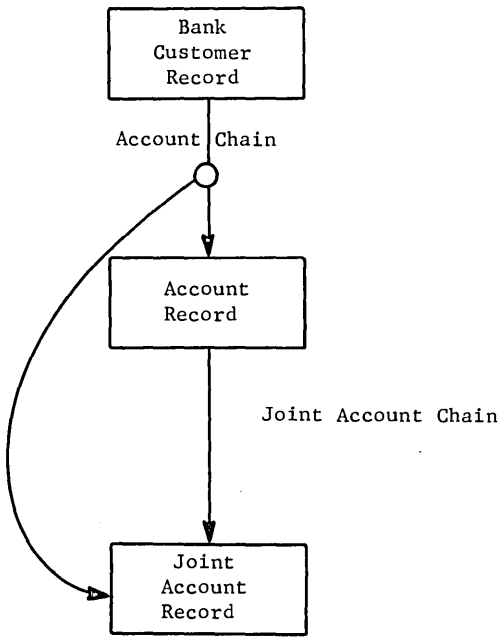


Figure 21.

It is normal in going from the system concepts to system design to system implementation to make compromises between the ideal and what is currently practical. This is also true of I-D-S data file designs where timing or space considerations may not permit the actual implementation of some data relationship. In the I-D-S data structure diagram technique, a dashed line chain represents a data relationship which conceptually exists, but which is not implemented as a chain in the actual file design. The actual implementation of the chain may be full, partial, or non-existent but is completely handled by the programmer through his data field manipulation.

Fig. 22 represents a data structure diagram where one chain is only conceptual or phantom and is not specified in the actual data description of the file.

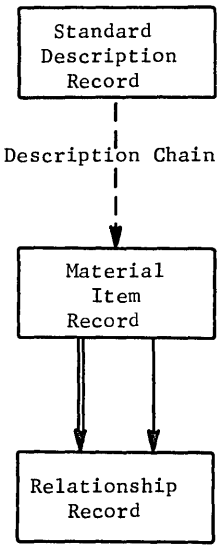


Figure 22.

In Fig. 22, a phantom chain is indicated which relates the Material Item Record to a Standard Description Record. There are three different programmer approaches to this chain.

1) The Material Item Record contains a field (containing the actual description and the Standard Description Record) which exists only in concept. When the Material Item Record is retrieved, its standard description is found within the record.

2) The system is specified using a Standard Description Record which can be retrieved on the basis of a Standard Description Code. The Material Item Record also contains the Standard Description Code corresponding to its standard description. The programmer can retrieve the standard description of a particular Material Item Record by retrieving the Material Item Record and using the contents of the Standard Description Code to then retrieving the Standard Description Record.

3) The system is again specified using a Standard Description Record which can be retrieved on the basis of a Standard Description Code. The Material Item Record this time contains a data field designed to hold the address of the appropriate Standard Description Record. When a new Material Item Record is to be stored, the appropriate Standard Description Record is retrieved and its address is placed in the address reference field prior to record storage.

   Retrieval of the Standard Description is accomplished by using the address found within the Material Item Record to retrieve the Standard Description Record.

In summary, the data structure diagrams, because of their clear visual representation, encourage the creation of intelligently designed information structures. The I-D-S programming language described in the following chapter translates and processes these structures with a minimum of programming effort.

The Integrated Data Store - Data Base Study, September 20, 1965, states a management information problem and develops a solution using I-D-S. The solution shows how to structure the information and process the file to prepare the required standard and special reports.

APPENDIX A

I-D-S Programming Language

Introduction

The source language of I-D-S is an extension of COBOL. Therefore, all formats and language specifications of COBOL must be adhered to when preparing a source program. In a few cases, certain elements of the COBOL language are not applicable to the I-D-S usage. These exceptions are mentioned in the following sections on the I-D-S language.

Identification Division

The purpose and usage of the Identification Division are identical with those defined for COBOL, with no special function for I-D-S.

Environment Division

All portions of the Environment Division, except the File-Control paragraph of the Input-Output Section, are used as defined by COBOL.

General Note

In GE-400 I-D-S unique names are to be unique within the first eight (8) characters.

FUNCTION:

To define the file name which represents the I-D-S data file, a unique version of the SELECT sentence is required as shown below:

FORMAT:

```
FILE-CONTROL                                    SELECT IDS file-name

        ASSIGN TO file-code-1 [file-code-2]
      [; BASE IS integer-3 PAGES]
      [; PROGRAM REQUIRES integer-4 BUFFERS]
```

NOTES:

1. The SELECT IDS sentence must be used only once to identify the I-D-S data file.

2. Other optional phrases of the SELECT sentence as specified for COBOL should not be used with the SELECT IDS sentence.

3. GE-400 I-D-S only.

   A. The ASSIGN-hardware-device-number designates the specific type of I/O media as well as the I/O channel and logical device number for each file. The type of hardware device assigned to a file is indicated by the first two digits of the four-digit octal device number. The logical device numbers for the data storage unit are as follows:

0501	Controller 1	File Unit 1	0511	Controller 2	File Unit 1
0502	Controller 1	File Unit 2	0512	Controller 2	File Unit 2
0503	Controller 1	File Unit 3	0513	Controller 2	File Unit 3
0504	Controller 1	File Unit 4	0514	Controller 2	File Unit 4

   B. The BASE clause allows the user to position the start of the I-D-S file on any disc. If the option is not used, I-D-S assumes the file starts on the first disc.

   C. The PROGRAM clause allows the user to select the number of pages to be retained in core. If this option is not used, program is assigned 3 buffers. Integer-4 cannot be less than 2.

4. GE-600 I-D-S only.

   A. The file code must be a two-character word consisting of two letters (A,...,Z) or a letter and a digit (0,...,9). Each file code must be unique with respect to other file codes in the program. At execution time, the object program is submitted to the General Comprehensive Operating Supervisor (GECOS) with "file cards" specifying the peripheral device for each file. The file code in the file card must be the same as that assigned in the source program. GECOS associates each of the objects program's files with its peripheral device by matching the file codes. (See the GE-625/635 Comprehensive Operating Supervisor Reference Manual, CPB-1002).

B. When multiple file units are used, an additional restriction is placed on the file code, if no I-D-S data file is physically stored on two or more file units. In this case, the allocator of GECOS assigns the file code specified by the file card to the first unit and a file code which is one greater to the second unit, etc. For example, if an I-D-S file has been physically stored on two file units and the file code of AA has been assigned, the first unit must be referenced using file code AA and the second unit, using file code AB. The user must be sure that any other files references by his program do not use a file code which may conflict with this procedure. The GECOS loader will set aside as much core as specified in the control cards loaded with the program. I-D-S will use all space unallocated by the actual program for buffer assignment.

DATA DIVISION

The description of the I-D-S data file is contained in a special section of the Data Division called the I-D-S Section. This section must physically follow the Working-Storage Section, if present, and precede the Constant Section.

The I-D-S Section contains the File Description, Record Description, Field Description, and Chain Description as required to describe the complete data file.

FILE DESCRIPTION

The File Description entry provides information regarding the physical characteristics of the I-D-S data file. The entry is used only for documentation purposes, since the Input/Output Controller module of I-D-S expects to find these same characteristics stored as a special Environment record within the disc storage unit. As this implies, the disc storage unit must be pre-conditioned with the Page Header records and the Environment record prior to the execution of any I-D-S program.

The entry consists of a level indicator, a file name, and a series of clauses which define the physical characteristics of the I-D-S file. The mnemonic level indicator MD is used to identify the start of the File Description entry and to distinguish it from the Record Description entries which will follow.

FUNCTION:

To describe the physical structure of the I-D-S file.

FORMAT:

```
MD file-name    ; PAGE CONTAINS integer-1 CHARACTERS
                ; FILE CONTAINS integer-2 PAGES
```

NOTES:

1. The file-name must be identical to the one used in the SELECT IDS sentence of the FILE-CONTROL paragraph of the Environment Division (Refer to "Environment Division" on Page 25.)

   Other optional phrases of the File Description entry as specified for COBOL do not apply to the I-D-S Section and should not be used.

2. The PAGE size (integer-1) specified may be any value up to a maximum of 4096 characters as determined by the particular system designer. However, the most efficient use of the storage capacity of Mass Storage Device involved should be considered when establishing the page size. See Appendix D for additional details.

3. The FILE clause is required to complete the physical description of the I-D-S environment. The total physical storage must be equivalent to the storage assigned by the FILE-CONTROL paragraph of the Environment Division. Integer-2 may be any number of pages. The maximum number of pages possible within the I-D-S page numbering system is 262,143 ($2^{18}-1$).

4. GE-400 BOS-MT I-D-S page size should be 1, 2, 4, 8, or 16 sectors.

5. GE-600 I-D-S only.

   The FILE clause expresses the total physical storage requirements of the I-D-S file. This value must be equivalent to or less than the capacity which has been reserved for the file by the allocation procedure of GECOS. When storage is allocated by GECOS, it is reserved in increments of 23,040 characters. This increment of storage is referred to as a "link". See the GECOS manual for a discussion of the allocation of permanent random disc or drum files.

DATA DESCRIPTION

The I-D-S Data Description entries accomplish three purposes:

1. Record Description. To name, describe and direct the placement, and define retrieval criteria of a record.

   This is the I-D-S 01 level entry.

2. Field Description. To name and describe the information content of a record.

   These are the I-D-S 02 level entries and they may be supplemented by COBOL 03-49 level entries.

3. Chain Description. To name and describe the inter-record relations (master/detail) and direct the insertion and retrieval of a record within the I-D-S structure.

   These are the I-D-S 98 level entries.

All I-D-S records are stored on the external media as fixed length records of "n" characters in length. Each record contains identification, plus at least one chain field for each chain association specified, plus as many characters of data fields as required by the totality of the 02 level field entries. The fields are packed into the records and records are packed into pages on a character oriented basis.

RECORD DESCRIPTION

In general, the format and usage of Record Descriptions required for I-D-S are comparable with those for COBOL Record Descriptions.

The Record Description entries perform three functions:

- Establish the existence of a record and assign a name and record type code to it.
- Establish parametric control over the placement of a record in the I-D-S page environment.
- Define the path of retrieval to be used by the associative RETRIEVE statement.

FUNCTION:

To specify the parameters which define an I-D-S record.

FORMAT:

Option 1

```
01 record-name;  TYPE IS integer-1
    [ ; PAGE-RANGE IS integer-3 TO integer-4 ]
                                  { field-name                 FIELD }
      ; RETRIEVAL VIA             { { chain-name-1 }           CHAIN }
                                  { { CALC         }                 }
    [ ; PLACE NEAR chain-name-2 CHAIN ]
    [ ; INTERVAL IS integer-2 PAGES ]
    [ ; AUTHORITY IS integer-5 ]
```

Option 2

```
01 record-name COPY FROM LIBRARY
```

NOTES:

1. Each of the above clauses is applicable only at the record (01) level.
2. Record-name must be unique within the total data base.
3. GE-600 I-D-S only.
   A. The AUTHORITY clause is currently implemented only on the GE-600 I-D-S.
   B. The option 2 entry is used only with the GE-600 I-D-S and only when the Record Description entry is to be copied from a library source. This library source, which is actually a file assigned to file code ". L", is searched for the level 01 entry identified by record-name. When the 01 entry is found, the Record Description entry and all of its associated entries (level 02-49 and 98) will be copied from the library. This enables the user to include only those portions of the total data description that are required for the current application. The user must be sure, however, that all records which are to be referenced by the procedure either directly or indirectly have been defined.

FUNCTION:

To define the Record Type which will be stored in each record of this type and which will be used to associate a retrieved record with its data description.

FORMAT:

```
; TYPE IS integer-1
```

NOTES:

1. This clause is required for each level 01 entry.

2. Integer-1 may be any value from 1 to 999.

3. The system permits the user to reassign the same type code to different records. This is useful under certain application situations and is safe if the record descriptions are compatible.

PAGE-RANGE

FUNCTION:

To specify a portion of the record placement parameters which provide a method for segmenting the data base, and specifying the segment in which an I-D-S record will be placed.

FORMAT:

```
; PAGE-RANGE IS integer-3 TO integer-4
```

NOTES:

1. Integer-3 and integer-4 represent the first and last page numbers of a series of pages within which records of this type are to be stored. Integer-3 may be of greater magnitude than integer-4. In this case, the range is defined as beginning with the integer-3 page and (wrapping around) extending through the integer-4 page.

2. The page numbers specified must fall within the total number of pages specified for the file by the FILE clause of the File Description entry.

3. If no PAGE-RANGE is specified, the range is assumed to be equal to the page range of the entire file.

4. Many different types of records may share the same page range.

5. The PAGE-RANGE clause delimits the action of the PLACE, NEAR, and INTERVAL clauses.

FUNCTION:

1. To specify a portion of the record placement parameters which will be used when the record is stored.

2. To specify the retrieval procedure to be used in conjunction with the "RETRIEVE record-name" form of the retrieval verb. (See RETRIEVE verb, Procedure Division).

FORMAT:

```
                                  [  field-name          FIELD  ]
; RETRIEVAL VIA                   [ {chain-name}    . .  CHAIN  ]
                                  [ {CALC      }         .      ]
```

NOTES:

1. This clause is required for each level 01 entry.

2. When the RETRIEVAL VIA field-name FIELD form is used, the field-name must be an 02 level field-name of the record and the field must be specified as an 8-digit numerical value (PICTURE IS 9 (8)). This field will not be stored in the data area of the record since its value is the record reference code. This is a primary record.

   GE-600 only. The field must be specified as SYNCHRONIZED LEFT.

3. This cell always contains the reference code of the last record processed by an I-D-S statement, unless its contents have been changed by the execution of a COBOL statement. Exception: see PLACE NEAR chain-name CHAIN clause.

   Overflow rule:

   If space is not available in the specified page, the record will be placed in the first page, in the direction of ascending page numbers, where there is available space. The pages wrap around. The PAGE-RANGE limit is observed.

4. When the RETRIEVAL VIA chain-name-1 CHAIN form is used, the chain-name must be one in which this record is specified as a DETAIL. This is a secondary record.

5. Placement of a secondary record will be in the same page as the master record of the chain-name CHAIN if there is space available. If not, the record will be placed in accordance to the overflow rule specified in Note #3.

   Exception, see "PLACE NEAR chain-name CHAIN" clause. (This applies to both primary and secondary records).

   A primary record will be placed in the page, the number of which is indicated by the contents of the communication call named DIRECT-REFERENCE.

6. When the RETRIEVAL VIA CALC CHAIN form is used, the record must contain a 98 level entry specifying that the record is a DETAIL of the CALC chain. This is a calculated record.

7. Placement of a calculated record will be in the same page as its randomized CALC chain master (Page Header record), if there is space available in that page. If not, the record will be placed in accordance with the overflow rule specified in Note #3. The randomized CALC chain master record will be determined based upon the RANDOMIZE fields specified in the 98 level entry and the standard randomizing routine furnished with I-D-S.

FUNCTION:

To specify a portion of the record placement parameters which will be used when the record is stored.

FORMAT:

[; PLACE NEAR chain-name-2 CHAIN]

NOTES:

1. This clause does not apply to calculated records.

2. When the PLACE NEAR chain-name-2 CHAIN clause is used, the record must be specified as a detail in chain-name-2.

3. Placement of a primary or secondary record under the PLACE NEAR ---- clause will supersede the record placement function specified by the RETRIEVAL VIA ....clause. The record will be stored in the same page as the master record of chain-name-2 CHAIN, if there is space available. If not, the record will be placed in accordance with the overflow rule specified under Note #3 of the RETRIEVAL VIA clause.

INTERVAL

FUNCTION:

To specify a portion of the record placement parameters which provide a method to ensure a uniform distribution of a given type of records within the I-D-S page environment.

FORMAT:

```
[; INTERVAL IS integer-2 PAGES]
```

NOTES:

1. Integer-2 may be zero or any positive value.

2. The INTERVAL clause does not apply to calculated records.

3. Placement of a primary or secondary record under the INTERVAL clause will supersede the record placement function of the RETRIEVAL VIA clause.

4. The placement of a record under the INTERVAL clause will be in the page represented by the sum of integer-2 and the page number of the last record of this type stored or retrieved, if there is space available in that page. If there is no space in the selected page, the record will be placed in accordance with the overflow rule specified in Note #3 of the RETRIEVAL VIA clause. If the procedure being executed has not yet processed a record of this type, the page number will be assumed to be zero.

FUNCTION:

To provide a method to safeguard the data contained in a record from un-authorized storage, reference or modification.

FORMAT:

[; AUTHORITY IS integer-5]

NOTES:

1. This clause conditionally inhibits the ability to use the following I-D-S verbs on the record type.

   a. STORE
   b. MOVE TO WORKING-STORAGE
   c. MODIFY
   d. DELETE
   e. HEAD

2. This clause has no effect upon the RETRIEVE verb.

3. Integer-5 may be any value not exceeding 4095 (10). The value supplied is used as a "lock" for the data contained in any record of this type. Whenever this record is referred to during execution, a "key" must have been supplied which matches the lock. The key is supplied by the OPEN statement as defined in the description of the Procedure Division.

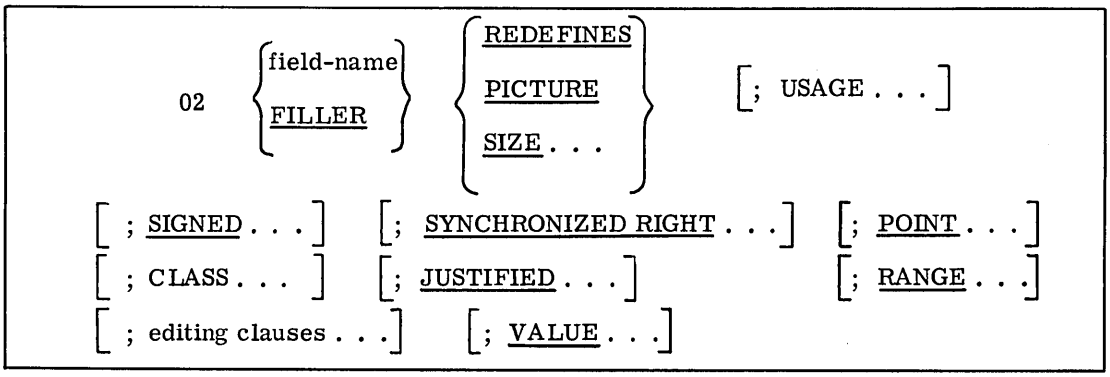
4. Implemented in GE-600 I-D-S only.

I-D-S FIELD DESCRIPTION ENTRY

FUNCTION:

To specify the characteristics of a particular data field.

FORMAT:



NOTES:

1. All fields will be left justified in a word oriented computer unless specified as SYCHRONIZED RIGHT. When SYNCHRONIZED RIGHT, the field will be moved to the right boundary in the word(s) used for its storage and extra characters on the left will be zero filled.

2. I-D-S will pass each 02 level Field Description on to COBOL as a 01 level Description within the WORKING-STORAGE SECTION. The WORKING STORAGE 01 level entries are the interchange points between I-D-S and COBOL. COBOL statements must initialize those fields before a record is stored and MOVE TO WORKING-STORAGE makes field values available after an I-D-S record retrieval. Detailed discussion of each option is specified in the COBOL manual and should be used in accordance with the COBOL specification for a 01 level entry for the WORKING-STORAGE SECTION.

3. The user may specify 03-49 level items as subfields to the 02 level I-D-S fields and have their descriptions passed on to COBOL. I-D-S does not itself recognize these fields or their field-names. The description of the 03-49 level subfields must agree in every sense with the COBOL specifications applicable to 02-49 level items in the WORKING-STORAGE SECTION.

4. I-D-S will only allow uniquely named 02 level field entries within the WORKING-STORAGE SECTION. If two I-D-S records contain 02 level fields with the same field-name, only the first 02 level entry and its 03-49 level subfield entries will be passed on to the COBOL WORKING-STORAGE SECTION. The WORKING-STORAGE field thus generated will be used by all records containing that field-name.

5. FILLER provides for extra storage space on the disc for anticipated future expansion. No WORKING-STORAGE space is provided.

6. A SIZE, PICTURE, or REDEFINES clause must be specified for every 02 level field. The SIZE or REDEFINES clause must be used if subfields (03-49 level) entries are specified. The SIZE clause value must encompass all of the subfields specified including the zeroes implied by any SYNCHRONIZE clause at the 03-49 level. The field specified by the REDEFINES clause and its subfields must be equal to or smaller in SIZE than the originally defined field.

7. REDEFINES, when used, must immediately follow field-name. All other clauses may be written in any order.

8. Clauses which begin with SIGNED, SYNCHRONIZED RIGHT, POINT, PICTURE, JUSTIFIED, or RANGE, as well as the editing clauses, cannot be used if subfield (03-49 level) entries are to be specified.

9. GE-400 I-D-S only.

   The GE-400 line I-D-S requires the second, third....02 level field entries, repeating a previously defined field-name, to use a special 96 level entry in lieu of 02 to signal that it is a repeat entry. The 03-49 level subfield entries are not permitted with a 96 level entry.

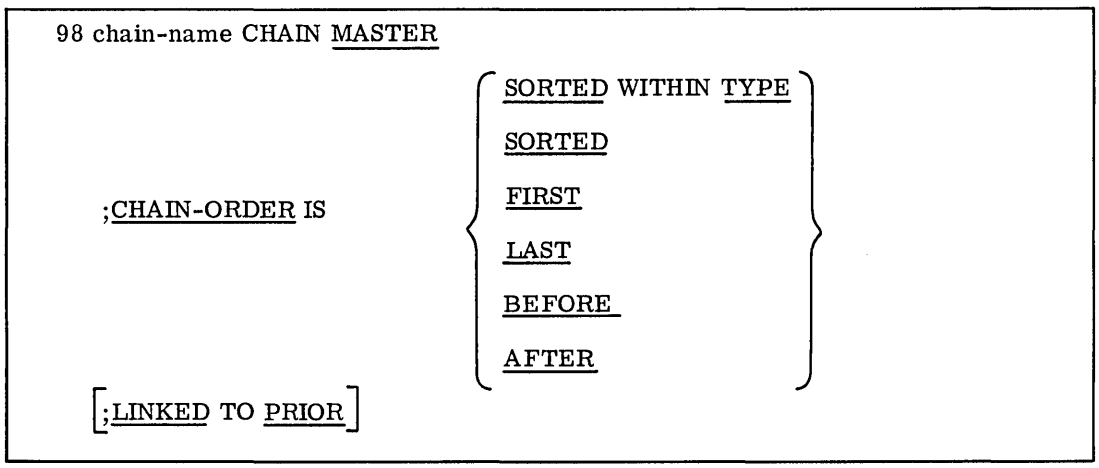
I-D-S CHAIN DESCRIPTION ENTRY

FUNCTION:

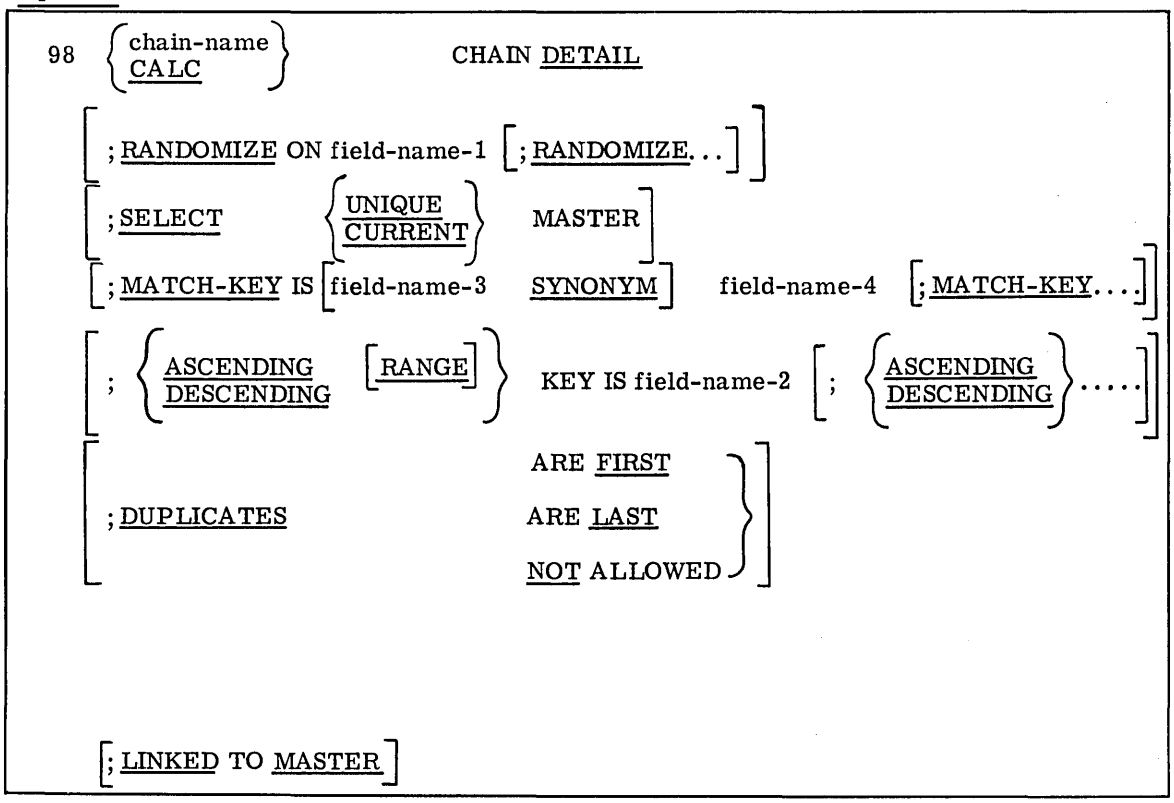
To name and describe the inter-record relations (master/detail), and direct the insertion of a record into the I-D-S structure.

FORMAT:

Option 1



Option 2



FUNCTION:

To describe the record's relationship to the chain-name CHAIN.

FORMAT:

Option 1

```
98 chain-name CHAIN MASTER
```

Option 2

```
98 { chain-name } CHAIN DETAIL
   { CALC       }
```

NOTES:

1. The chain-name must be unique within the total data base.

2. Either Option 1 or Option 2 must be used for each chain with which the currently defined record is to be associated. (See also Note #3). Only one record type may be defined as the master record of a chain. Any number of record types may be defined as detail records of a chain.

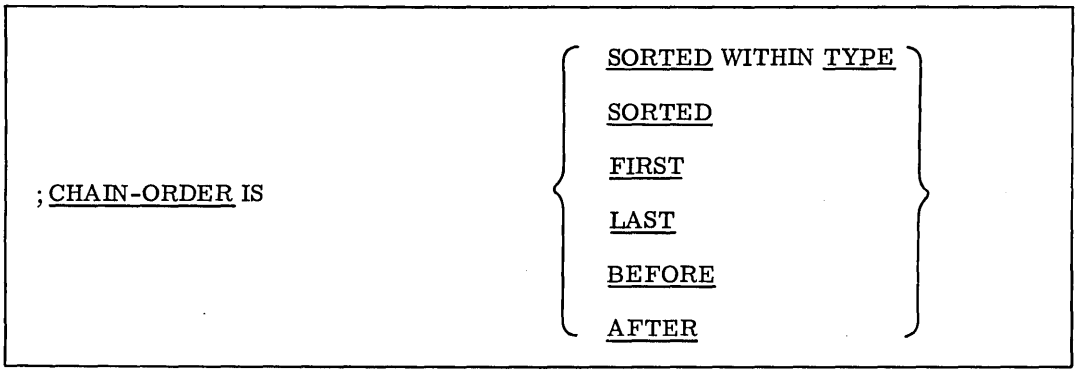
3. An Option 2 entry must be used whenever the record has been defined as a calculated record by the RETRIEVAL VIA clause. In this case, the record must be defined as a detail of the CALC CHAIN.

CHAIN-ORDER

FUNCTION:

To specify the criteria for sequencing the details of the chain.

FORMAT:



NOTES:

1. This clause must be used in each Master Chain Definition entry.

2. When either of the SORTED options is specified, details will be added to the chain based upon the contents of the defined sort control fields of the detail records. When the SORTED WITHIN TYPE form is used, records of the chain are maintained in sequence within record type, independent of other types. This does not mean that there is an implied major sort by record type code. It means only that when a given type of record is considered independently of any other detail record of that chain, it is in sequence by its own sort key(s).

   When the SORTED form is used, the various records of the chain are maintained in a single sequence regardless of the number of record types in the chain. The name, size, and class of major control fields of the various records must be identical. The major control fields are those that control inter-record-type sorting.

3. The last four forms shown for this clause cause a detail to be inserted in the chain relative to some other record in the chain. These options are:

FIRST	Insert detail in chain immediately following the master record.
LAST	Insert detail in chain immediately preceding the master record.
BEFORE	Insert record immediately preceding the current* record of chain.
AFTER	Insert record immediately following the current* record of chain.

   *The current record of a chain will always be the master record if SELECT UNIQUE MASTER has been specified. In this case, BEFORE and AFTER acts like LAST and FIRST respectively.

4. The selection of the BEFORE and LAST options causes IDS to create an extra chain field which contains the reference code of the immediately preceding record of the chain. BEFORE causes the creation of this field in all record types of the chain. LAST introduces this field in the master record type only.

FUNCTION:

To provide an extra chain field for each record of the chain which will contain the reference code of the immediately preceding record in the chain.

FORMAT:

[; LINKED TO PRIOR]

NOTES:

1. The clause provides a two-way path so that the chain can be traversed in both directions. This ability is important for several reasons:

   - Facilitates the action of the RETRIEVE PRIOR verb.
   - Facilitates the action of the MODIFY verb.
   - Enables the immediate removal of a deleted record which would otherwise stay linked in this chain until the chain is again traversed. This could save valuable space.

2. The use of chain PRIOR fields has two disadvantages. First, the record size is increased to provide space for the additional chain field. Second, the linking process is slower, because the chain PRIOR field of the next record must be adjusted when a new record is inserted.

3. Under two conditions, chain PRIOR fields are automatically provided. If the "BEFORE current record" CHAIN-ORDER is selected, all record types defined for the chain will contain a chain PRIOR chain field. If the LAST detail CHAIN-ORDER is specified, the master record (only) will contain a chain PRIOR chain field.

RANDOMIZE

FUNCTION:

To specify those fields of a calculated record which generate the page number used in record placement.

FORMAT:

```
[; RANDOMIZE ON field-name-1        [; RANDOMIZE. . . .]]
```

NOTES:

1. If this clause is not used, the page number will be the lowest within the page range specified for this record type.

2. Field-name-1 must be an 02 level field contained in the record being defined.

3. The randomizing routine of I-D-S uses as many fields as are specified.

4. The word RANDOMIZE must precede each control field specified.

5. The fields, designated as RANDOMIZE fields, will be compared at record storage time. Therefore, an attempt to store a record with identical RANDOMIZE field values will be rejected as an error.

FUNCTION:

To specify the criteria for selecting the specific master record from all master records of a given type when a detail record is being stored.

FORMAT:

```
; SELECT { UNIQUE  }        MASTER
       { CURRENT }
```

NOTES:

1. One of the two forms of this clause must be used in each 98 level chain-name CHAIN DETAIL entry. It does not apply to CALC CHAIN DETAIL. That definition implies that the Page Header record specified by the output of the randomizing procedure is the unique master to be selected.

2. When using the UNIQUE option, the master is selected by matching the data field values in a master record with those supplied in WORKING-STORAGE. The field(s) to be initialized are those specified as MATCH-KEY fields in the 98 level entry.

3. Specifying CURRENT MASTER causes the master of the current record of this chain to be selected. If the current record of the chain is the master itself, then this is the master selected. The user's procedure must make sure that the current entry in the chain table leads to the desired master.

MATCH-KEY

FUNCTION:

To specify those data fields which must be initialized in WORKING-STORAGE to enable UNIQUE identification of the MASTER record of this chain.

FORMAT:

```
MATCH-KEY IS [field-name-3 SYNONYM] field-name-4 [MATCH-KEY....]
```

NOTES:

1. This clause must be used in conjunction with the SELECT UNIQUE clause.

2. The field-name field(s) named by this clause must match, on a field-by-field basis, the control field(s) contained in the master record as well as control fields specified or implied by the RETRIEVAL clause for each of the higher level master records, if any, in the hierarchical structure which includes this record. These fields need not be specified or present in the detail record.

   - When a master record is defined as a primary record, the data-field named by the RETRIEVAL VIA field-name FIELD must be named as a MATCH-KEY for this detail record. This terminates the record hierarchy.

   - When a master record is defined as a secondary record by the RETRIEVAL VIA chain-name CHAIN clause, and the CHAIN-ORDER is SORTED, the data fields named by the ASCENDING/DESCENDING KEY and ASCENDING RANGE KEY clauses must be named as MATCH-KEYS for this detail record. This does not terminate the record hierarchy.

   - When a master record is defined as a calculated record, the data fields defined as RANDOMIZE control fields must be named as MATCH-KEY fields for this detail record. This terminates the record hierarchy.

3. The "field-name-3 SYNONYM" clause equates the field-name-4 MATCH-KEY as used by the master record to field-name-3 when the file designer wants to specify an alternate source for the MATCH-KEY and not access the normal MATCH-KEY field.

4. GE-400 I-D-S only.

   IF field-name-3 is a field-name which is not defined in this record as an 02 or 96 level field entry, then it must be defined in this record type with a special 97 level number. This reserves space in WORKING-STORAGE.

FUNCTION:

To specify those data fields which will control the sequence of the detail records of the chain named.

FORMAT:

```
[ ; {ASCENDING [RANGE] } KEY IS field-name-2 [ ; {ASCENDING } .... ] ]
    {DESCENDING        }                       [   {DESCENDING}     ]
```

NOTES:

1. Field-name-2 must be a 02 level field entry contained in the record being defined.

2. When multiple sort control keys are required to define a chain sequence, the various field-names must be presented in sequence from major control field to minor, thus establishing the sort level of each field. Each sort control key must be independently defined as either ASCENDING or DESCENDING.

3. The sorted chain has two auxiliary purposes which require searching the named chain in the forward direction until the retrieved record's sort key value is equal to the working storage value.

   - They are used by the RETRIEVE record-name RECORD statement to assist in the associative retrieval of the named record.
   - They are used by the STORE record-name RECORD statement to assist in retrieving the UNIQUE MASTER required for linking a new detail record.

   The ASCENDING RANGE clause modifies the two search functions listed above to the extent that the retrieval search is continued until the retrieved record's sort key value is equal to or greater than the working storage value.

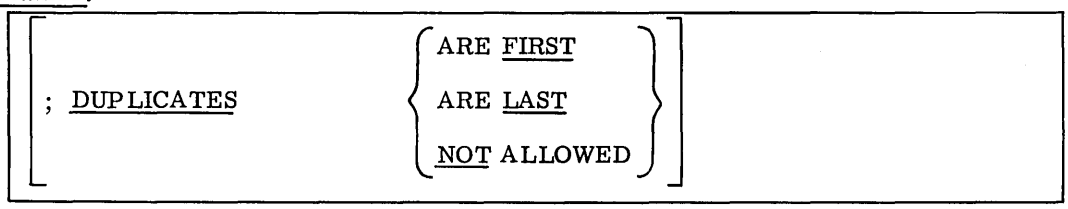
4. When more than one type of detail record is specified for a sorted chain, the major control fields of the various records must be identical in name, size, and class. The major control fields are those that control inter-record-type sorting.

DUPLICATES

FUNCTION:

To specify whether or not records with identical sort key values may exist in a chain and what ordering action should be taken, if permitted.

FORMAT:



NOTES:

1. This clause must be used and only used when the chain has been defined as a sorted chain by the CHAIN-ORDER clause.

2. When DUPLICATES are allowed, the new detail may be positioned as the FIRST or LAST of the string of records with identical sort key values.

3. When DUPLICATES are NOT ALLOWED, an error condition will be noted and the record rejected whenever storage of a record with identical sort key values is attempted.

FUNCTION:

To provide an extra chain field for each detail record of the type being defined, which will contain the reference code of the master record of the chain.

FORMAT:

```
[; LINKED TO MASTER]
```

NOTES:

1. This optional clause can improve the operation of the system by providing a direct path from each detail of the type being defined, to the master of the chain.

2. The MASTER chain field will increase the size of the record type being defined.

SAMPLE CODING. The following example describes the records shown in Figure 20 in I-D-S source language. This illustrates one method of preparing the Data Division for the purchase order problem.

```
DATA DIVISION.
IDS SECTION.
    MD  INFORMATION-FILE; PAGE CONTAINS 1920 CHARACTERS
        FILE CONTAINS 1020 PAGES.
        01  VENDOR-REC TYPE IS 100; RETRIEVAL VIA CALC CHAIN.
        02  VENDORNO SIZE IS 6 NUMERIC.
        02  VENDOR-NAME SIZE IS 18 ALPHANUMERIC.
        02  ADDRESS SIZE IS 24 ALPHANUMERIC.
        02  CITY-STATE SIZE IS 20 ALPHANUMERIC.
            98  CALC CHAIN DETAIL; RANDOMIZE ON VENDORNO.
            98  ORDER CHAIN MASTER; LINKED TO PRIOR;
                CHAIN-ORDER IS SORTED.
        01  ORDER-REC TYPE IS 105; RETRIEVAL VIA CALC CHAIN.
        02  ORDER-NO SIZE IS 5 NUMERIC.
        02  ORDER-DATE SIZE IS 3 NUMERIC.
        02  PURCHAGENTNO SIZE IS 2 NUMERIC SYNCHRONIZED RIGHT.
            98  CALC CHAIN DETAIL; RANDOMIZE ON ORDER-NO.
            98  ORDER CHAIN DETAIL; SELECT UNIQUE MASTER;
                MATCH-KEY IS VENDORNO; ASCENDING KEY IS ORDER-NO;
                DUPLICATES NOT ALLOWED.
            98  ITEM CHAIN MASTER; CHAIN-ORDER IS SORTED.
        01  ITEM-REC TYPE IS 110; RETRIEVAL VIA ITEM CHAIN; PLACE NEAR
              INVENTORY CHAIN.
        02  ITEMNO SIZE IS 2 NUMERIC.
        02  DUEDATE PICTURE 9(6).
        02  ORDER-QTY PICTURE IS 9999V9.
            98  ITEM CHAIN DETAIL; SELECT CURRENT MASTER;
                ASCENDING KEY IS ITEMNO; DUPLICATES NOT ALLOWED.
            98  INVENTORY CHAIN DETAIL; SELECT UNIQUE MASTER;
                MATCH-KEY MATLIDENT; ASCENDING KEY IS DUEDATE;
                DUPLICATES ARE LAST; LINKED TO MASTER.
        01  INVENTITEM TYPE IS 115; RETRIEVAL VIA CALC CHAIN.
        02  MATLIDENT PICTURE IS X(18).
        02  MATLDESC PICTURE IS X(30).
        02  QUANTONHAND PICTURE IS 9(8).
            98  INVENTORY CHAIN MASTER; CHAIN-ORDER IS SORTED.
            98  CALC CHAIN DETAIL; RANDOMIZE MATLIDENT.
```

PROCEDURE DIVISION

Execution of the I-D-S procedural statements will STORE, RETRIEVE, MOVE TO WORKING-STORAGE, MODIFY and DELETE records. In addition, these statements have the implicit responsibility of maintaining the structure of the data file that is created by the defined chain relationships.

The communication interface between I-D-S procedural statements and the balance of the COBOL Procedure Division, are the Working-Storage areas which are established for each 02 level field defined in the field description entries of the I-D-S Section. All COBOL references to data from the I-D-S file are to these Working Storage areas.

The procedural statements of I-D-S may appear anywhere in the context of the COBOL Procedure Division. An I-D-S sentence is always preceded by ENTER IDS. The sentence may contain any number of I-D-S statements and must be terminated by a period. A paragraph name or section name may be assigned to an I-D-S sentence in a manner consistent with normal COBOL format.

The following pages describe how these various statement formats may be used and the limitations which apply to each.

OPEN

FUNCTION:

To initialize the processing of the I-D-S data base.

FORMAT:

```
OPEN    [ FOR { RETRIEVAL } ]
              { UPDATE    }
[WITH AUTHORITY-KEY integer-1]          file-name
;  IF ERROR GO TO procedure-name.
```

NOTES:

1. This statement must be executed before any other I-D-S statement is executed.

2. The IF ERROR clause tests the occurrence of any logical or physical error as a result of the last I-D-S statement. The specific errors which may occur are a function of the statement executed. The user program may determine the type of error by referring to the communication cell named ERROR-REFERENCE.

3. If the error is a hardware error, data description error, or results from an improper use of I-D-S functions, the program will be brought to an orderly halt, the file closed and the program aborted and memory dumped with the appropriate error message typed on the console typewriter.

4. If a data dependent error is detected by I-D-S, and the user has not provided an IF ERROR clause, the function will not have been executed, the error code will be stored in ERROR-REFERENCE and control will pass to the next statements.

5. The execution of a subsequent I-D-S statement will reset a value stored in ERROR-REFERENCE.

6. GE-400 I-D-S only.

   The file-name clause must agree with the file-name specified by the MD level entry of the IDS SECTION and with the file-name in the SELECT IDS file-name clause in the ENVIRONMENT DIVISION.

7. GE-600 I-D-S only.

   A. When the I-D-S file is opened FOR RETRIEVAL, the STORE, DELETE, and MODIFY statements of I-D-S are not operative. An attempt to use these statements when I-D-S file is OPEN FOR RETRIEVAL, results in an error condition during program execution. When the FOR { RETRIEVAL / UPDATE } clause is not used, UPDATE is assumed by I-D-S.

   B. The AUTHORITY-KEY clause enables access to various record types which may be protected by a defined AUTHORITY code. (See Data Division, Record Description). The value of integer-1 may not exceed 4095. When this clause is used, each reference to a record of the I-D-S file involves a match of the AUTHORITY value defined for the record with the AUTHORITY-KEY supplied. When a valid match occurs, the I-D-S verb is allowed to function normally. Otherwise, the function of the verb is aborted and an error condition is returned to the user's program.

FUNCTION:

To place a record into the I-D-S data base and to establish any chain fields which have been defined.

FORMAT:

```
STORE record-name-1 RECORD; IF ERROR GO TO procedure-name.
```

NOTES:

1. Record-name-1 must be one defined by a 01 level entry record description in the I-D-S Section of the Data Division.

2. When this verb is used, the following is assumed:

   - Each field for this record has been initialized with the desired value in the WORKING-STORAGE area.

   - Any other control fields required to provide MATCH-KEY values of the master records of the defined chains have been initialized in their respective WORKING-STORAGE areas.

3. The record is placed into the data base in accordance with the clauses of the Record Description entry.

4. The reference code assigned to the record stored is accessible in the communication cell named DIRECT-REFERENCE after the storage process is completed.

   If the reference code is divided by 64, the quotient is the page number and the remainder is the line number of the record.

5. The record stored is recorded as the CURRENT record of its type and as the CURRENT record in each chain in which it is defined.

6. If the storage process would create a record with identical values of control fields in violation of the record description, the storage process is aborted and an error condition is noted.

7. When a primary record is stored, its reference code is moved to the WORKING-STORAGE field named by the RETRIEVAL VIA field-name FIELD clause.

RETRIEVE

FUNCTION:

To cause a record to be retrieved and made available for subsequent processing.

FORMAT:

```
                    { record-name RECORD    [ VIA chain-name-1 CHAIN ]
                    { DIRECT
                    { CURRENT record-name RECORD
                    {  { NEXT  }
RETRIEVE            {  { PRIOR }      RECORD OF chain-name-2 CHAIN
                    {  { MASTER}
                    { EACH AT END GO TO procedure-name-1
; IF ERROR GO TO procedure-name-2
```

NOTES:

1. The various forms of the RETRIEVE verb are hereafter referred to as the record specifiers.

2. Record-name must be one defined by a 01 level record description entry in the I-D-S Section of the Data Division.

3. Chain-name-2 must be one defined by a 98 level chain description entry in the I-D-S Section of the Data Division.

4. The reference code of the record retrieved is accessible in the communication cell named DIRECT-REFERENCE after the retrieval process is completed.

5. Of the seven record specifiers which may be used with the RETRIEVE verb, two may be classified as context independent. This means that they are independent of any prior record processing.

   - RETRIEVE record-name RECORD.

     The record retrieval action is predicated upon the RETRIEVAL VIA clause defined in the 01 level entry in the I-D-S Section. The record retrieved will depend upon the value(s) contained in the control fields of WORKING-STORAGE which uniquely identify the record. If the record is retrieved "VIA field-name FIELD," the contents of the named field (its reference code) are used.

     If the record is retrieved VIA CALC CHAIN, the contents of the RANDOMIZE field(s) are used. If the record is retrieved VIA chain-name-CHAIN, the contents of its MATCH-KEY and ASCENDING and DESCENDING sort key fields are used.

- RETRIEVE DIRECT. The record to be retrieved is the one whose reference code is stored in a communication cell named DIRECT-REFERENCE.

  The other five record specifiers may be classified as context dependent, since the actual record retrieved is dependent upon previous record processing.

- RETRIEVE { NEXT / PRIOR / MASTER } RECORD OF chain-name CHAIN

  In these cases, the record retrieval depends upon the CURRENT record within the chain specified. If NEXT or PRIOR is used, the appropriate record is retrieved regardless of the record type. When MASTER is specified, the master record of the chain named is retrieved. These record specifiers can be used only if some record has already been processed which is a member of the specified chain.

- RETRIEVE CURRENT record-name RECORD

  The record retrieved will be the last record of the record-name specified that has been processed. If no record of this record-name has been processed, or if the last record of this record-name processed was deleted, an error condition will be noted.

- RETRIEVE EACH AT END GO TO procedure-name-1.

  This record specifier facilitates a reference code ascending sequence serial search of the data base. This statement will retrieve the first record, in ascending reference code sequence, whose reference code value is equal to or greater than the reference code value stored in the communication cell named FIRST-REFERENCE. However, if that retrieved record's reference code value is equal to, or greater than the value stored in the communication cell named LAST-REFERENCE, control will be transferred to the procedure-name-1.

  When a record is retrieved, the sum of its reference code value plus one will be stored in FIRST-REFERENCE, initializing it for a subsequent execution of RETRIEVE EACH.

  The reference code for any record may be computed by multiplying the page number by 64 and adding the line number.

6. The record retrieved is recorded as the CURRENT record of its type and is the CURRENT record in each chain in which it is defined.

7. If a record cannot be retrieved according to the specifications of the retrieval statement, an error condition is noted.

8. GE-400 I-D-S only.

   The RETRIEVE VIA chain-name-1 CHAIN is implemented only on GE-400 DAPS I-D-S.

MOVE

FUNCTION:

1. To cause all field values or selected field values of the record last processed to be moved to WORKING-STORAGE.

2. To move the contents of the named CHAIN TABLE to the WORKING-STORAGE field specified by field-name-3.

FORMAT:

Option 1

```
; MOVE { field-name-1 [field-name-2] ... }
       { TO WORKING STORAGE               }

; IF ERROR GO TO procedure-name
```

Option 2 (implemented on GE-400 DAPS/I-D-S only)

```
; MOVE    chain-name CHAIN TABLE TO field-name-3
```

NOTES:

1. This statement must be used before any reference can be made to the data in WORKING-STORAGE.

2. MOVE to WORKING-STORAGE will unpack all of the 02 level fields of a record.

3. MOVE field-name-1, field-name-2,... will MOVE the named 02 level fields to WORKING-STORAGE.

4. Each field is unpacked into WORKING-STORAGE in accordance with the 02 level Field Description entries as interpreted by COBOL as 01 level entries.

5. GE-400 I-D-S only.

   A. Selective field moves on the GE-400 line I-D-S must repeat the statement, MOVE before each field name. The IF ERROR statement must appear only after the last one.

   B. When using option 2, MASTER, PRIOR, CURRENT, and NEXT entries of the named CHAIN TABLE will occupy successive eight-character sub-field of the 32-character field specified by field-name-3.

6. GE-600 I-D-S only.

   The TO WORKING STORAGE noise words may not be used on the GE-600 I-D-S.

FUNCTION:

To retrieve and move to WORKING-STORAGE the data fields of the master record of the chain specified.

FORMAT:

```
; HEAD chain-name CHAIN ; IF ERROR GO TO procedure-name
```

NOTES:

1. Chain-name must be the name of a chain as defined by a level 98 entry. Some record within the chain-name chain must have been previously processed.

2. This is the only I-D-S statement which includes an implied move of the record to WORKING-STORAGE.

3. After execution of this statement, the master record retrieved is the CURRENT record of its respective type. It becomes the CURRENT record in each chain in which it is defined as detail. However, it does not become the CURRENT record in any chain in which it is defined as a master record. In those chains, the CURRENT record remains unchanged. Note that the function of the statement is very similar to that of the RETRIEVE MASTER RECORD; MOVE TO WORKING-STORAGE, except for the manner in which the chain tables are updated.

MODIFY

FUNCTION:

To modify the contents of selected fields of the current record and/or to relink any chain which may be affected by the modification of a control field.

FORMAT:

```
; MODIFY field-name-1    [; field-name-2...]
  IF ERROR GO TO procedure-name
```

NOTES:

1. The field modified must be a 02 level entry. The contents of WORKING-STORAGE are moved to the equivalent field of the record.

2. When the field modified is a Sort-key field, the record is relinked in accordance with the new value of its Sort key field.

3. When the field modified is a MATCH-KEY field, the association of the current record with its master is changed. The record is delinked from its current master, its new master is retrieved, and the record is linked to its new master according to the CHAIN-ORDER clause for the chain. The record then is a detail of a different chain.

4. If the successful execution of the MODIFY statement would create DUPLICATE records in chains where they are not allowed, the modification will not be executed and an error will be noted.

5. GE-400 I-D-S only.

   Multiple field modification on the GE-400 I-D-S must repeat the statement MODIFY before each field-name. The IF ERROR statement need appear only after the last one.

FUNCTION:

To delete the current record and remove it from the chains in which it is a detail, delete all of its detail records, and to optionally perform certain functions when specified detail record types are accessed during the deletion process.

FORMAT:

```
; DELETE  [ON record-name-1 DETAIL]

 a)      [; MOVE    TO WORKING-STORAGE ]

 b)      [; HEAD    chain-name CHAIN ]

 c)      [; {PERFORM procedure-name-1 }  ]
            {GO TO procedure-name-1   }

         [; {OTHERWISE } ON  record-name-2 DETAIL.... ]]
            {ELSE      }

         ; IF ERROR GO TO procedure-name-2
```

NOTES:

1. The execution of a DELETE statement makes the record retrieved unavailable for any further processing, and subsequent attempt to reference such a record results in an error condition.

2. If a record to be deleted is a master with attached detail records, the system deletes all such details beginning at the lowest level in the structure before deleting the master under consideration.

   The conditional statement "ON record-name-1 DETAIL" is used to interrupt the deletion process when a detail named by record-name is accessed. When this statement is used, the statements of the type a), b), and c), if present, are executed prior to the actual deletion of the detail record. After the execution of these statements, the deletion process is continued unless one of the statements was a GO TO statement. In that case, control is transferred to the procedure named. When the record accessed is not a record as specified by record-name-1, it is compared with the name specified by record-name-2. The reserved words OTHERWISE or ELSE separate the statements for different record types.

3. The MOVE TO WORKING-STORAGE statement under deletion control does not permit the selective field moves.

DEBUG

FUNCTION:

To permit selective dumping of I-D-S records, chain tables, or pages.

FORMAT:

```
DEBUG [CURRENT BUFFER] [; CURRENT RECORD] [; CURRENT CCBLOC]
      [; chain-name-2 CHAIN] [; AND TRACE chain-name-3 CHAIN]
```

NOTES:

1. BUFFER — Octal/BCD printout of the current page
   RECORD — Octal/BCD printout of the current record
   CCBLOC — BCD printout of Direct Reference, First Reference, Last Reference, Record type, and Error Reference; octal printout of Direct Reference, First Reference, Last Reference, and Record type.

2. Chain-name-2 and chain-name-3 must be the names of chains as defined by a level 98 entry of the I-D-S section of the Data Division.

   Chain-name-2 Octal/BCD printout of the reference codes contained in the master definition of the chain requested.

   Chain-name-3 Octal/BCD printout of all records contained within the chain requested.

3. The above procedural statement may appear anywhere in the context of the COBOL Procedure Division and must always be preceded by ENTER IDS.

4. The printouts produced by the DEBUG verb will be written on the system output file for printing on the execution report.

5. Implemented on GE-600 I-D-S only.

FUNCTION:

To execute a specific procedure and then return to the normal sequence.

FORMAT:

```
; PERFORM procedure-name
```

NOTES:

1. Procedure-name may be any COBOL procedural paragraph in the Procedure Division. It may not contain any I-D-S statement.

GO TO

FUNCTION:

To transfer control to the procedure named.

FORMAT:

```
; GO TO procedure-name.
```

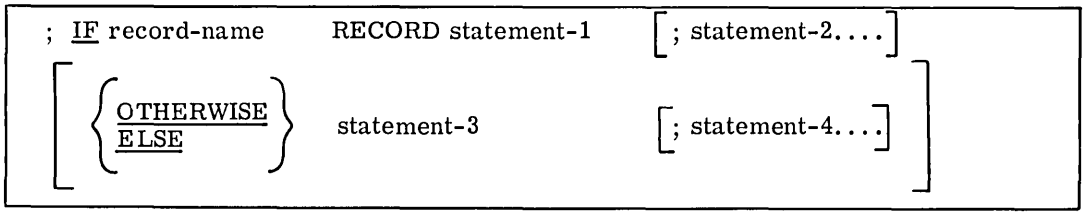
NOTES:

1. Procedure-name may be any COBOL or I-D-S procedural paragraph.

FUNCTION:

To conditionally transfer control to an alternate procedure.

FORMAT:



NOTES:

1. The IF record-name clause is specifically designed to support the use of those retrieval statements where the type of record to be retrieved may not be known until after the retrieval is complete. Specifically, the IF record-name clause is often, but not necessarily always, used with RETRIEVE DIRECT, RETRIEVE EACH, and RETRIEVE {NEXT / PRIOR} RECORD OF chain-name CHAIN statements.

2. If the record retrieved is of the type specified by the record name, statement-1 and subsequent statement-2's will be executed in sequence and then control will be transferred to the next sentence in the program. A GO TO procedure-name statement may be used as either statement-1, or statement-2 to cause a programmer specified transfer to some alternate sentence in the program. If the record retrieved is not of the type specified, then control will be transferred around statement-1 and subsequent statement-2's to statement-3, or to the next sentence in the absence of an OTHERWISE or ELSE phrase.

   Statement-1, 2, 3, 4 may be any one of the following statements: MOVE TO WORKING STORAGE, MODIFY, DELETE, HEAD, PERFORM, or GO TO. In addition, statement-3 may be another IF record-name clause. This allows multiple test branch logic based on record type.

CLOSE

FUNCTION:

To transfer all modified pages currently residing in the core buffers to the mass storage unit at the end of a program.

FORMAT:

CLOSE	file-name	; IF ERROR GO TO procedure-name

NOTES:

1. This statement must be executed before any COBOL STOP RUN statement. No automatic closing takes place.

2. GE-400 I-D-S only.

   The file-name clause of the CLOSE statement applies only to the GE-400 I-D-S. The file-name must agree with the file-name used in the SELECT IDS clause in the ENVIRONMENT DIVISION and in the MD entry in the DATA DIVISION (IDS SECTION).

APPENDIX B

I-D-S RESERVED WORDS

I-D-S uses all the reserved words specified for COBOL. In addition, it employs the reserved words listed below. The user must avoid using these words for I-D-S data-names.

AFTER
ALLOWED
ARE
ASCENDING
AUTHORITY
AUTHORITY KEY
BEFORE
CALC
CHAIN
CHAIN-ORDER
CURRENT
DELETE
DESCENDING
DIRECT
DIRECT-REFERENCE
DUPLICATES
EACH
ERROR
ERROR-REFERENCE
FIELD
FIRST
FIRST-REFERENCE
HEAD
IDS
INTERVAL
KEY
LAST
LAST-REFERENCE
LINKED
MASTER
MATCH-KEY
MODIFY
NEAR
NEXT
PAGES
PAGE-RANGE
PLACE
PRIOR
RANDOMIZE
RANGE
RETRIEVAL
RETRIEVE
SELECT
SORTED
STORE
SYNONYM
TYPE
UNIQUE
UPDATE
VIA

APPENDIX C

CHAIN TABLE UPDATING

The RETRIEVE NEXT, PRIOR, and MASTER statements, and SELECT CURRENT MASTER data description, are dependent upon the current status of the Chain Tables in carrying out their functions. Therefore, the I-D-S user need have some knowledge of the existence, function and update logic of these tables in order to plan his program logic.

These tables are updated by the Chain Update Subroutine. The logic of this routine follows:

STORE
RETRIEVE
(MODIFY)

1) For the record which is the object of the statement, the chain tables are updated by the Chain Update Subroutine.

2) For master records necessarily accessed to execute the statement (including MODIFY where control fields are changed) the chain tables are updated to reflect their master record's processing.

DELETE

For record specified and all its details, if record is:

1) Detail, then only the current entry in chain table is affected and it is set equal to the next entry.

2) Master, then all entries are set to zero.

If a DELETE ON data-name DETAIL GO TO statement-name clause is executed, then the chain tables are updated by the Chain Update subroutine.

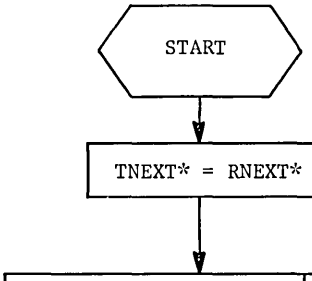
HEAD

For the record which is the object of the HEAD chain-name statement, its chain tables are updated as follows:

1) If Detail, then tables are updated using Chain Update subroutine.

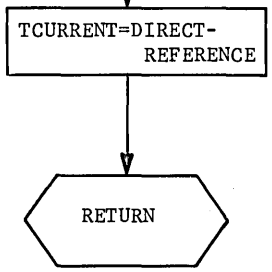
2) If Master, then tables are not updated.

Update Chain Subroutine



LINKED TO PRIOR?	RECORD ACCESSED BY THIS CHAIN?	DIRECTION OF CHAIN ACCESS	TPRIOR =
YES	--	--	RPRIOR
NO	NO	--	0
NO	YES	NEXT	TCURRENT
NO	YES	PRIOR	0
NO	YES	MASTER	0

RECORD ROLE IN THIS CHAIN	DETAIL LINKED TO MASTER?	RECORD ACCESSED BY THIS CHAIN?	THEAD =
MASTER	--	--	DIRECT-REFERENCE
DETAIL	YES	--	RHEAD
DETAIL	NO	YES	--
DETAIL	NO	NO	0



* T---- Chain table entries
R---- Record chain field

APPENDIX D

Data Storage Unit Hardware Considerations

The Integrated Data Store will map the user-designated pages into the assigned storage device. It converts page numbers into the appropriate hardware address at execution time. This allows changes or additions of hardware without changes in the user's program or reorganization of the file.

The determination of actual hardware address is based upon the origin of the logical file within the total information environment and the relative location of the page within the logical file.

Efficient utilization of the mass storage device(s) is dependent upon selecting a page size consistent with the hardware characteristics. A page will be mapped into a space which is defined as a number of data storage unit sectors. The minimum number of sectors is one and the maximum number is defined by the number of sectors required to store the maximum size page (4096 characters).

If the page size is not an exact multiple of the sector size, expressed as characters, the excess characters will be stored using another full sector, but using only that required to meet the specified page size. The balance of the sector is not used.

The term "area" is defined as the maximum number of sectors which may be read or written with a single I/O command. An integral number of pages will be stored within an area. When the page size is divided into the area capacity (both expressed as a number of sectors) the dividend is the number of pages which will be established per area. The balance of the sectors of the area are not used.

Best utilization of the data storage units is accomplished when: 1) the page size specified is an integral number times the character capacity per sector and that same integral number is an exact divisor of the area size expressed in sectors.

APPENDIX E

GLOSSARY AND INDEX OF TERMS USED IN I-D-S REFERENCES

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO U.S.A.