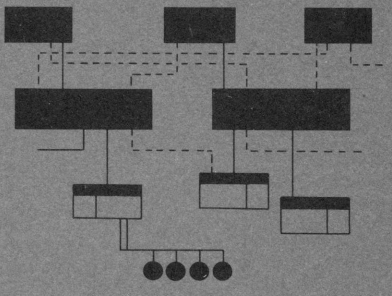
GENERAL ELECTRIC
COMPUTERS

GE-625/635 Integrated Data Store



GENERAL ELECTRIC

CPB – 1093

GE-625/635
INTEGRATED
DATA STORE

REFERENCE MANUAL

August 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

PREFACE

This manual presents a general description of the IDS system and discusses its features and capabilities. A description of the source language and detailed programming information is also presented.

IDS is a mass memory oriented system initially implemented for use with the DS-20 Disc Storage Unit. Detailed information on this equipment appears in the DS-20 Disc Storage Unit reference manual, CPB-345. Future releases of the IDS system will be applicable to the DS-15 and DS-25 mass storage units. Manuals on these units will be published shortly.

Since IDS is used to extend the functions of the COBOL language, the reader should have a working knowledge of that language before using this manual.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

CONTENTS

APPENDIX

ILLUSTRATIONS

1. INTRODUCTION

Integrated Data Store (IDS) is a new information-oriented method of integrating the operating functions of a business. Its use sharply reduces the high systems and programming costs associated with implementing an integrated business system. As a general-purpose system, it uses mass random access storage as an extension of memory and provides an efficient data organization technique. In addition, a simple but effective language is used to operate the system.

IDS provides a convenient method of describing complex information structures through the meaningful association of the data record contents. Once the data is described, the system automatically structures it to suit the hardware requirements of the mass storage device. The task of organizing data records for meaningful association is handled by the IDS system. This record association is achieved through the use of chains, which provide cross-reference linkages between records. These record chains are the integrating force of IDS.

The IDS language provides a simplified means for record processing in the environment of mass random access storage.

Present procedural languages offer programming convenience in field and sequential record processing. However, they are inadequate for processing records in the random environment of mass storage.

Language statements such as those for read/write operations produce serial rather than random actions. The burden of organizing data records and designing the logic involved in processing and maintaining these records is placed upon the programmer. These same functions are performed automatically by the IDS software system.

2. 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. Then, the results must be coordinated and correlated. IDS provides an answer to these problems: a single file organized around the data rather than the function.

The concept of Integrated Data Store significantly simplifies the design of integrated systems by permitting the establishment of meaningful associations between data records without any redundancy. A comparison between the conventional approach and the IDS approach to systems design in the following discussion illustrates the benefits of the latter.

COMPARISON OF SYSTEMS DESIGN APPROACHES

Conventional File Organization

When designing an information system, the designer must resolve many questions. 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?

If, in a conventional approach to file organization, the problem being studied is the design of a purchasing information system, it may be necessary to plan three duplicate files of purchase orders:

1. A vendor file with related open purchase orders for general processing.
2. A purchase order file for expediting purposes.
3. A file by inventory item number for production inquiries.

Figure 1 illustrates a conventional approach to file organization using disc storage.

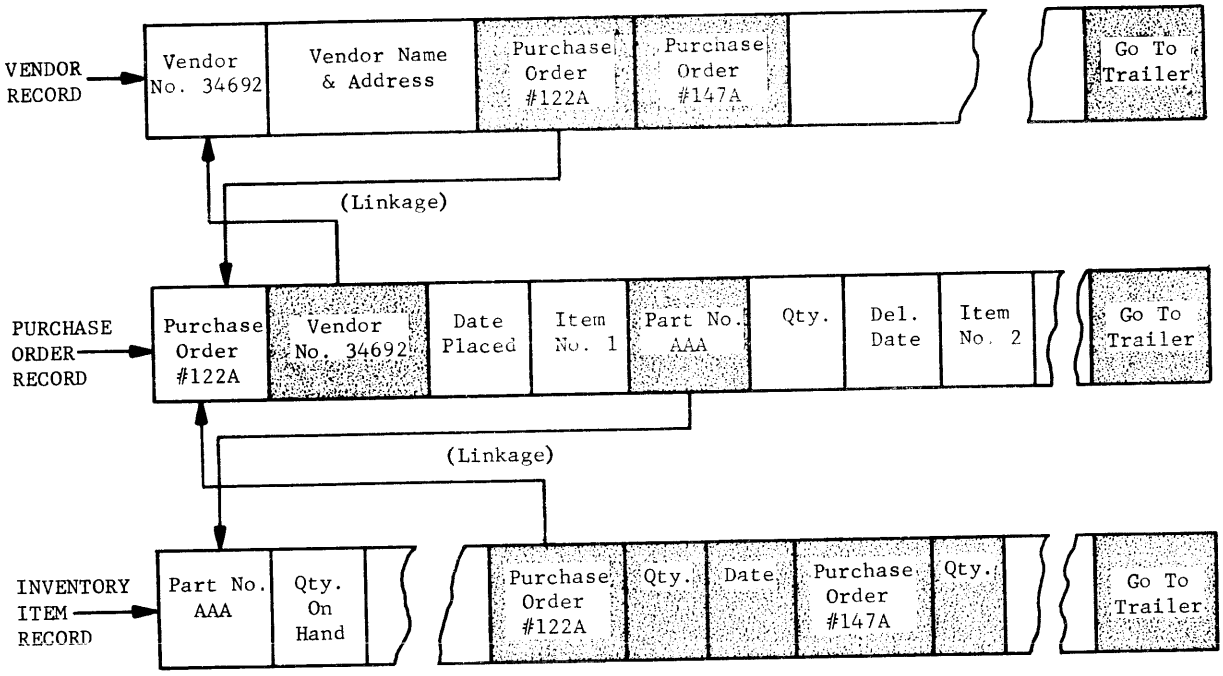


Figure 1. Conventional Record Formats

Note that the series of records shown in Figure 1 contains a considerable amount of redundant data (shaded fields). These records are typically stored in separate areas of the file unit with trailers to separate overflow areas, as shown in Figure 2. (Overflow areas are an extension of the normal file area. They store 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.

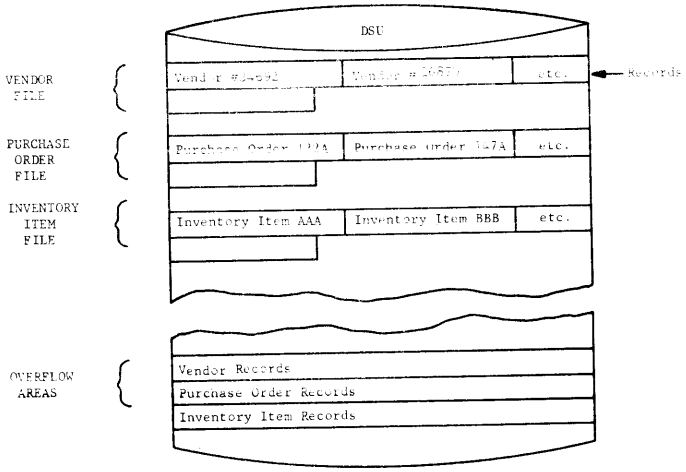


Figure 2. DSU Layout--Conventional

The processing and maintenance of records stored, as already shown, is both 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 (Figure 3).

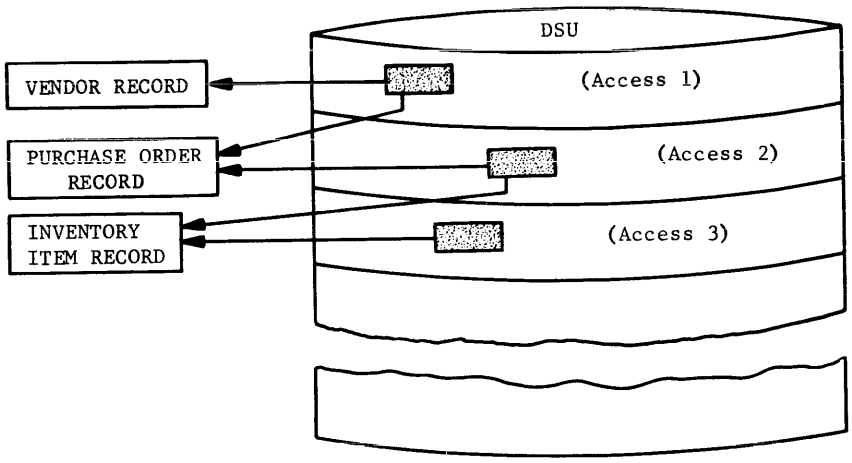


Figure 3. Record Access

If there can be 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 require considerable processing time.

IDS File Organization

IDS was designed to relieve the systems programming and storage problems inherent in the conventional approach to data organization and the subsequent processing of this data.

The IDS record construction comparable to the examples of conventional file organization appear in Figure 4.

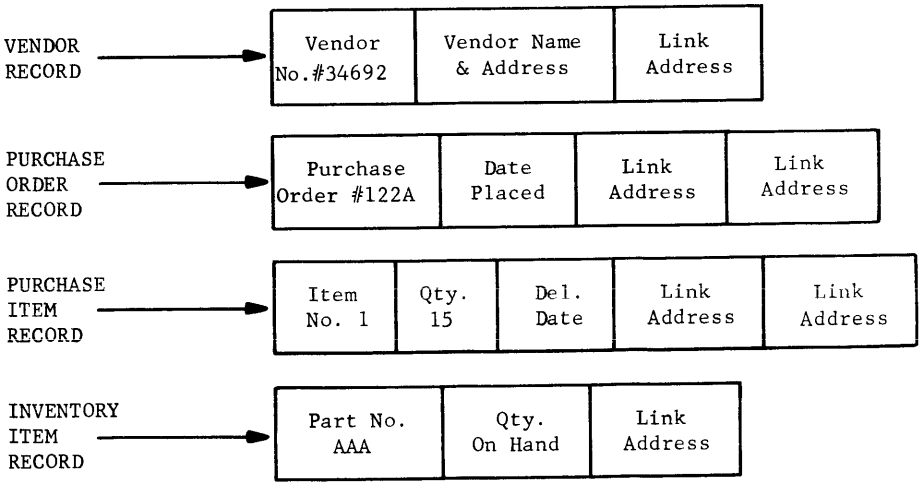


Figure 4. IDS Record Formats

Note there is no redundant information in the above 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 described the items being purchased.

ASSOCIATING RECORDS INTO CHAINS

The chaining feature is the fundamental structuring tool of IDS. Chains are made up of all the information about a particular function, as in the vendor record in Figure 5. 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. Chains are linked together automatically. 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. Figure 5 illustrates the chaining network of logical records as they might appear in an information system.

With an information system structured in this manner, file interrogation and processing is conveniently simplified.

In examining the records and chaining associations in Figure 5 note that there are four types of records:

Vendor
Purchase order
Item
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

Within each chain there can be a variable number of records. These records can be linked into any number of chains. The information system organized in this manner permits interrogation by all functions of the business with records and data stored only once in IDS.

In the IDS system, it is possible to store related records within the same block on the disc storage unit. During each disc access a block of information is transferred to memory. Therefore, 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 stored in memory.

With this feature in mind and by using Figure 5, it is easy to show 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 Figure 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.)

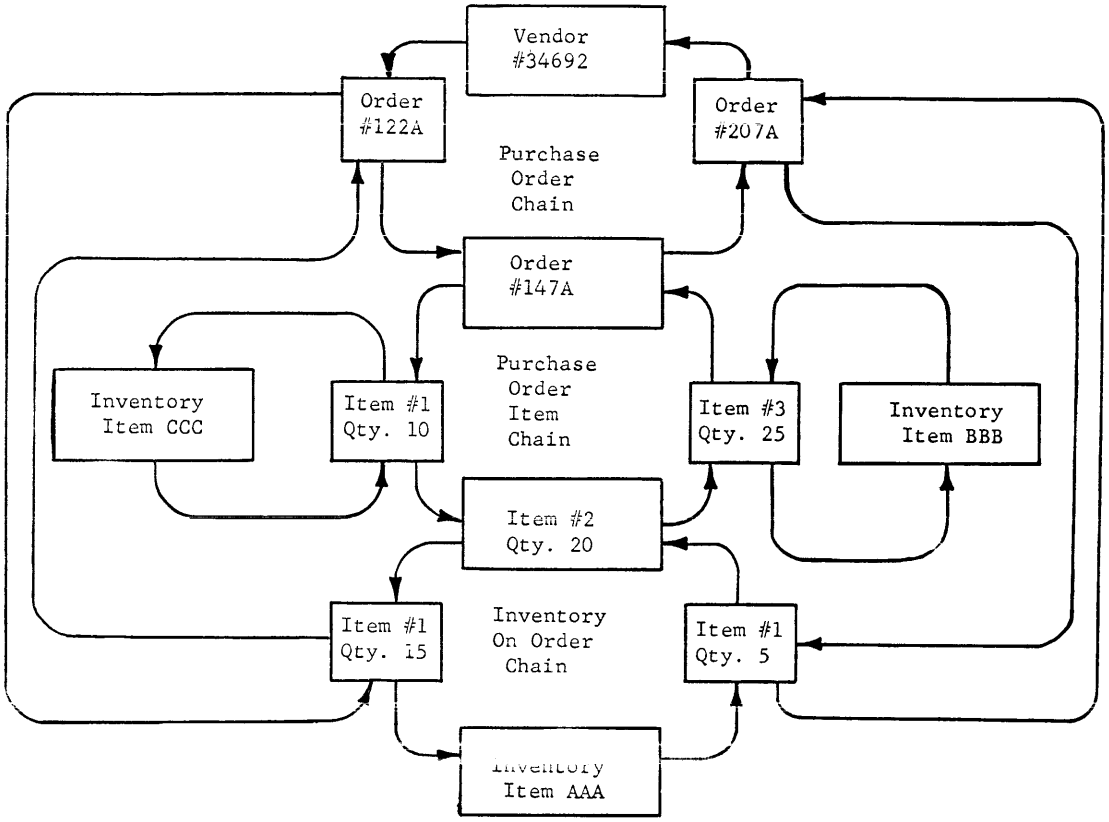


Figure 5. Chaining Example

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 order 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.

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, which is then associated with its purchase orders through chains.

Flexibility of the IDS system organization can best be illustrated by viewing a second method of processing the records in Figure 5. The production control functions of the business might, for example, inquire as to the inventory status, both on hand and scheduled on order, of inventory item AAA. 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 on order chain.)

2. Retrieve next item 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 purchase order 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 accomplished simultaneously with the above processing. This is possible because the items in the inventory on order chain are also chained into the order item chain and in turn into the purchase order chain. By traversing these chains, all necessary purchase order and vendor data can be obtained.

3. IDS ENVIRONMENT

The reader must have a general knowledge of a disc storage unit (DSU) to understand the over-all organization and concept of IDS. IDS implementation for the Compatibles/600 computers utilizes the DS-20 Disc Storage Unit. The following discussion presents some basic information on the DSU and on IDS record organization.

DS-20

Each DS-20 contains a maximum of 16 circular discs. Both sides of each disc are used, giving 32 recording surfaces. Information is recorded as magnetized spots on concentric tracks.

Each disc surface is divided into an inner zone and an outer zone. There are 128 circular tracks in each zone, making a total of 256 tracks on each side of each disc. The 128 outer tracks are divided into 16 sectors, and the 128 inner tracks are divided into 8 sectors. Figure 6 shows the format and the manner in which sectors are numbered on disc surfaces.

The sector (sometimes called a "frame" or "record") is the basic, physically addressable unit on a disc and consists of 240 characters for the GE-600 Series.

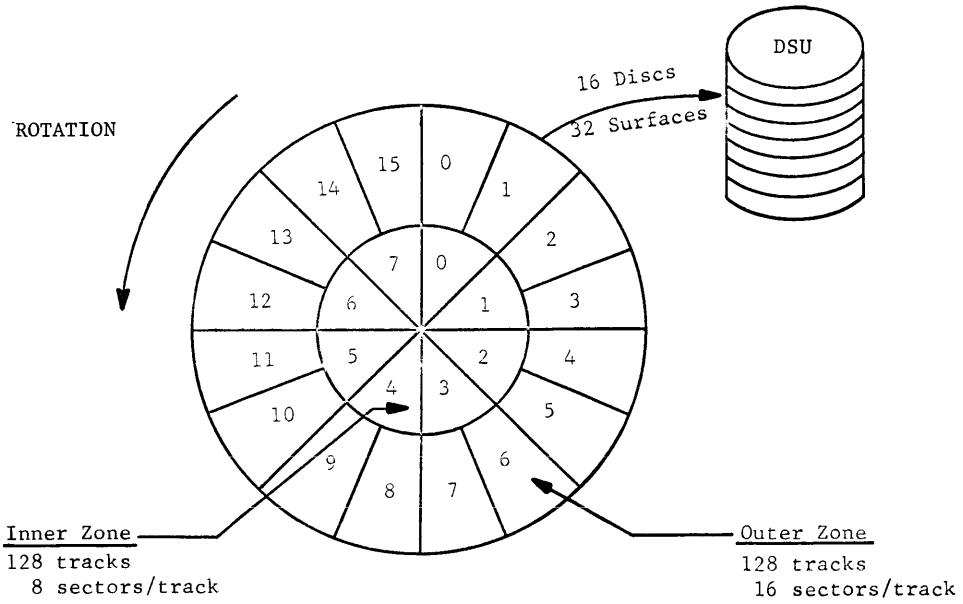


Figure 6. Disc Surface Configuration

Read/Write System

Each disc is served by a separate movable positioning arm. Each arm contains eight read/write heads, four for the upper surface of the disc and four for the lower.

Sectors are continuously addressable within each individual file unit. Up to 32 contiguous sectors can be read or written with one instruction from the central processor. With the arm in one position, it is possible to transfer a total of 96 sectors.

For each sector on the disc for information storage, there is permanently recorded a sector (record) address. When a search is made to locate the correct sector for a read or write operation, the physical address and the address contained in the instruction are compared. When the two addresses are confirmed electronically as identical, the addressed sector is read or written. The disc address tells where a record is physically located in the file.

Logical records are those records designated by the systems designer/programmer--for example, inventory record or payroll record. They contain data fields pertaining to a specific segment of the application system. Logical records are composed of data and chain fields. The total combined length of a logical record may be any length up to the capacity of a data block (IDS page) which is typically 8-16 sectors.

Conventional disc organizations (not IDS) specify one or more logical records to be contained in a sector, or one or more full or partial sectors to be equal to a logical record. Examples of conventional disc records are shown in Figure 7.

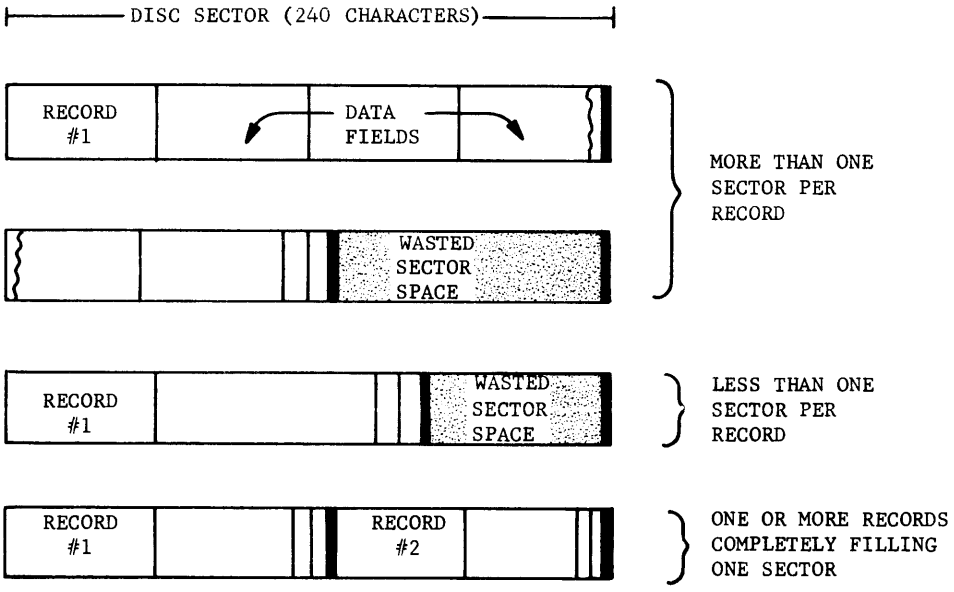


Figure 7. Conventional Disc Records

Note that where equality, or multiples thereof, does not exist (normally the case), inefficiencies in disc storage utilization result.

IDS Pages

In the IDS system a page (or block) of records consists of a fixed number of disc sectors as assigned by the program implementor. A page may contain any combination of logical record types linked into their respective chains. Each type of record has its own specific length. Related record types are associated and linked according to their data content and may be stored within the same page. Space is fully utilized by the automatic packing of these records within the page. During processing, whole pages are read into memory. Therefore, with proper data organization, the probability is high that much of the related information will be available in a single file access. Figure 8 is an example of an IDS page.

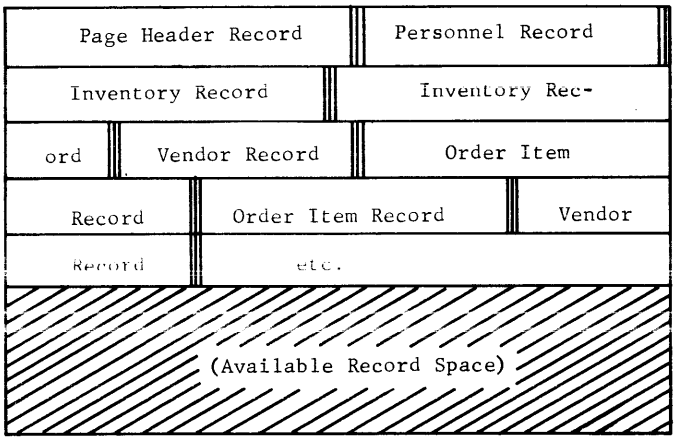


Figure 8. IDS Page

The organization of the "page" facilitates file maintenance and processing, as well as making it possible to accommodate a variable number of record types without duplication of information.

Every page begins with a unique Page "Header" record. This record 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

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

1. Line number
2. Record type
3. Record length

Reference Code

The reference code is a relative addressing code permanently assigned to each record. It can be considered to be a logical address in contrast to an address that defines where the record is physically located in the file. Once a record is assigned a reference code, it maintains that reference code regardless of expansion, contraction, segmentation, or other basic reorganization of the record or 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 specific line number of a data record contained within the page as explained below:

1. The page number portion, a sequential number permanently assigned to each page, which defines what proportion of the way through the IDS page 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 IDS mapping routine at execution time.
2. The line number portion, is a numbering system of records within each 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.

INPUT/OUTPUT CONTROLLER

The Input/Output Controller of the IDS controls the mass storage device. Its major function is to control the flow of pages of records in and out of memory in response to commands to store, retrieve, modify and delete specific data records. It also controls the page processing function within memory. Figure 9 illustrates the functions of the Input/Output Controller.

Data Buffers

To minimize the mass storage seek and transfer time, an inventory of data pages is maintained in memory. These pages are stored in numerous buffers in memory. The number of buffers depends on the amount of space available after the IDS 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 desired 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.

Priority Control

Each time a new page is brought into memory, its page number buffer location is placed at the head of a page list. If a page already in memory is used again, 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).

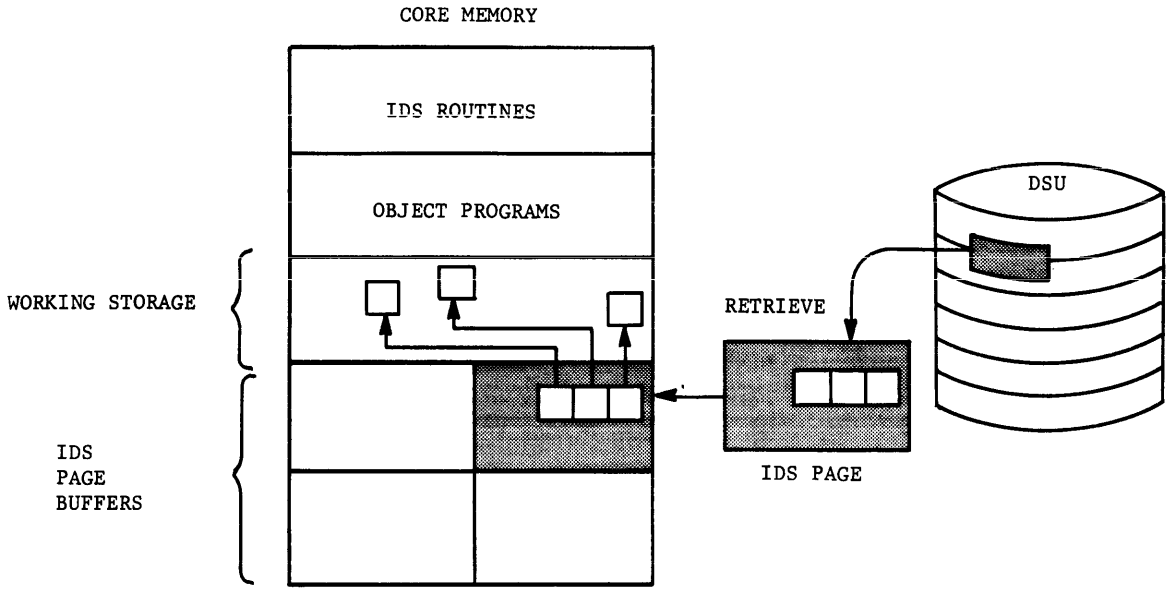


Figure 9. Input/Output Controller

Record Unpacking

Once the Input/Output Controller finds a place for the page in memory, it locates the record called for in the page. The fields from the record will be unpacked into Working-Storage if a MOVE TO WORKING-STORAGE command is specified.

File Protection

The automatic control provided by the Input/Output Controller for bringing records into memory, writing from memory to disc storages and making the record available for the programmer in the Working-Storage area, eliminates to a large extent the possibility of erroneous updating of record fields. File integrity is therefore maintained through the elimination of record maintenance by the programmer and by restricting the programmer to the updating of record fields in the Working-Storage area only. The modify function, which can address data fields, provides the only way of changing an actual record.

4. DATA ORGANIZATION

Data organization refers to the interrecord relationships established within the IDS. 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 IDS record contains a set of "data" fields which collectively describe the event, activity, status, or plan that the record represents. IDS augments these records with additional fields called "identification" and "chain" fields, as shown in Figure 10.

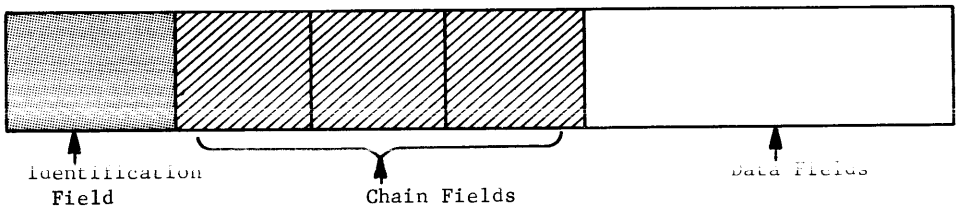


Figure 10. IDS Record

The chain fields contain the reference codes of other IDS records. They point from one record to the next, creating a circular association of records (Figure 11).

These associations are automatically processed according to the data descriptions and the procedural commands executed. The arrows in Figure 11 indicate the linking actually carried out through the storing of the reference code of one record in the body of the prior record.

DATA RECORDS AND FIELDS

The records of the IDS are fixed-format, fixed-length records 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. In order that control may be maintained, 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.

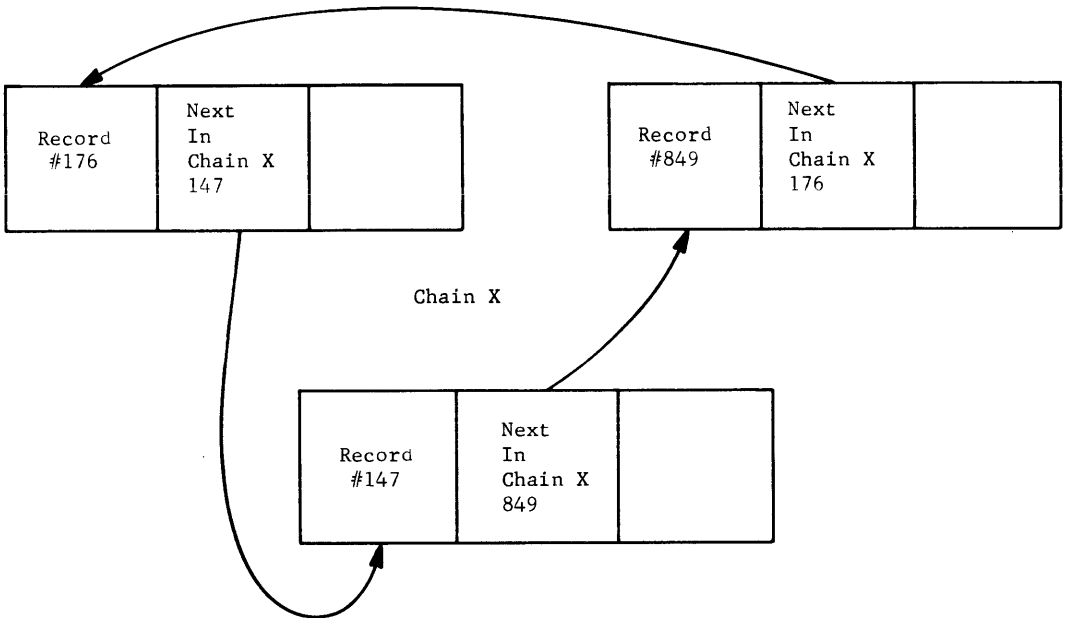


Figure 11. Chain Association

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 part number or an employee's name. These fields will be specified by the systems designer.

Chain fields contain the reference code for addressing and are defined for each chain in which a record participates. Experience in IDS systems indicates that the average record is in only two chains. An occasional pivotal record in the information integration 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.

IDS records are stored only once in IDS. 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).

Record Classes

The Integrated Data Store recognizes three distinct classes of records that it must store and retrieve. The designation of the data records as to class is at the option of the systems designer and is based on the storage and retrieval requirements of these data records.

IDS record processing requires that there be some aspect of every record which makes it unique, or different from any other record. All records are unique by virtue of their reference code. Some records are also unique because they contain one or more data fields--such as a drawing number, order number, and pay number--where no duplicate values are allowed.

CALCULATED RECORDS. Any record within the system can be classified as a "calculated" record. Its storage and retrieval are based upon the contents of one or more data fields. The contents of these fields are externally specified numbers--such as employee numbers, part numbers, or order numbers. The contents of these fields are processed through a randomizing procedure which determines a page number for an initial storage location. The record is stored at this page or one very close to it. A purchase order or an inventory item record is typically defined as this type of record. The subsequent retrieval of this record follows this same basic procedure.

SECONDARY RECORDS. "Secondary" records are a class made unique by virtue of their chain relationship to a specified type of master record and a sort control field used to sequence that chain. The item records associated with a purchase order (master) record are a good example of secondary order records (refer to Figure 5). These records are stored and retrieved by first locating the purchase order record and then stepping through the order item chain to locate the item record by comparison of its item number field.

PRIMARY RECORDS. Records designated in the data description as "primary" are unique only as a result of their reference codes. Generally primary records are used in place of calculated records where the external assignment of identification fields, such as part number or order number, is not required. In these cases, an internally generated number (the reference code) is assigned and used as the identification field.

IDS CHAINS

An IDS chain is illustrated in Figure 12. Its characteristics are:

- One master record and any number of detail records
- Links records together in an endless loop
- Associates related records in meaningful sequences

In addition to records being designated as to class for storage and retrieval purposes, they must also be defined as to their relationship within a **chain--master or detail records.** These record relationships are specified, when the chain is defined, in the data description.

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

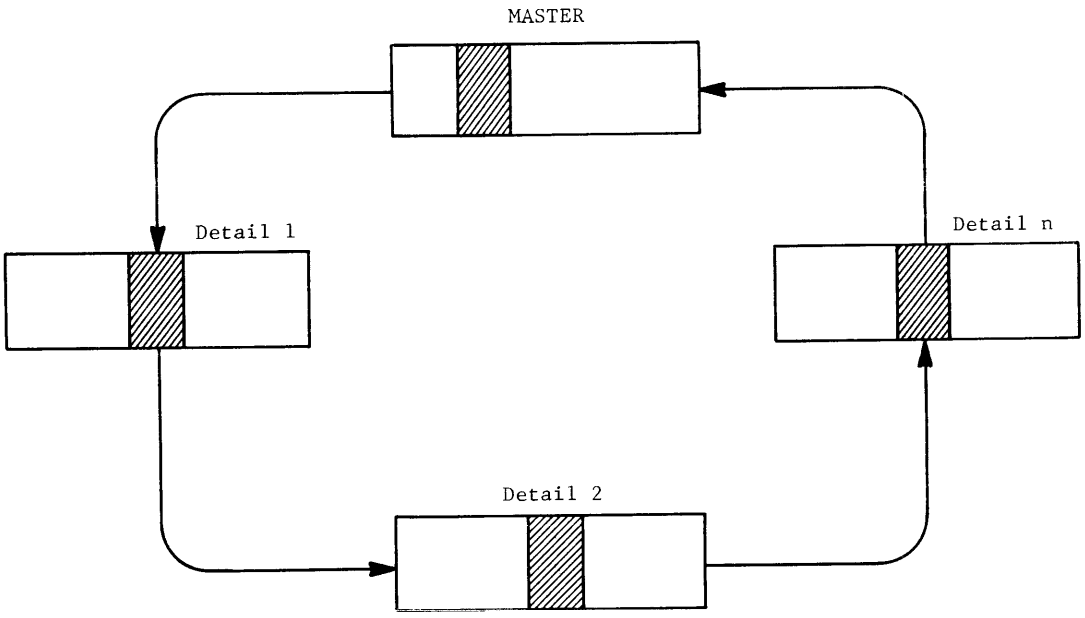


Figure 12. IDS Chain

MULTIPLE CHAINS

Chains exist for two separate but closely related reasons. First is the case where the source documentation shows that 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 Figure 13.

The second case for chain structuring of information involves the association of several related source documents. Relating all the purchase orders for a given vendor is an example. Figure 5 (page 7) illustrates this example. The purchase order chain associates all of the purchase order records with their vendor record.

IDS 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 IDS. The chain is destroyed 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.

Detail records cannot be stored unless a master record exists which qualifies as the particular master according to the specified master selection rule for that chain.

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

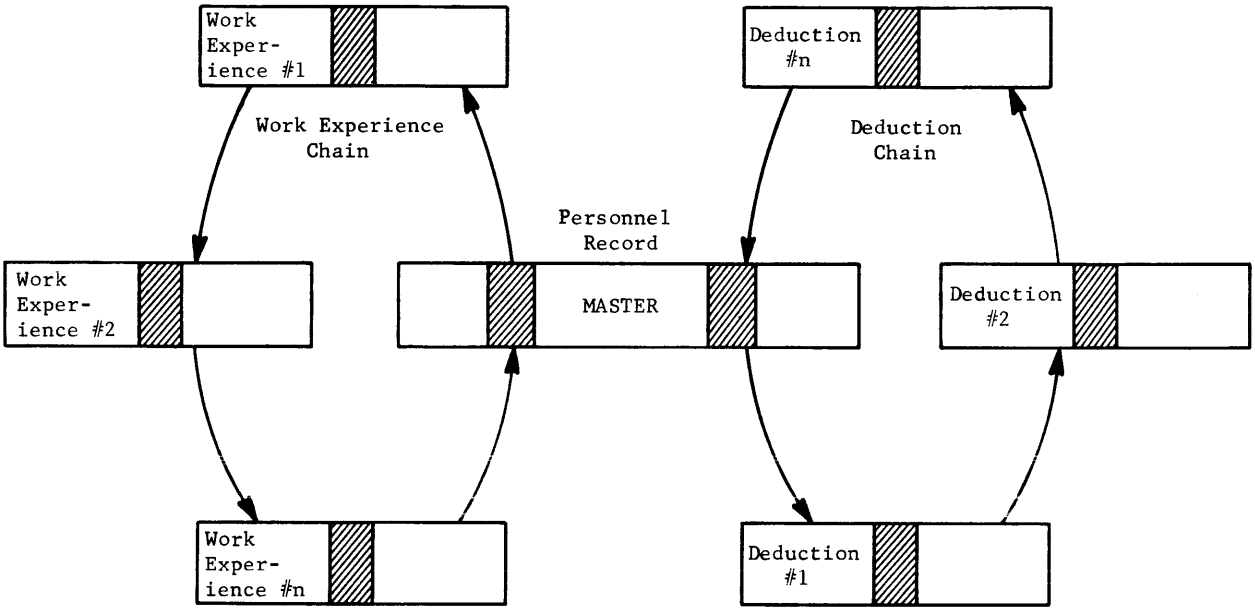


Figure 13. Master Record of Two Chains

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 (detail or master) 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.

A record cannot be defined as a detail to itself directly or indirectly.

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 patched to relink around the deleted record.

CHAIN PROCESSING

IDS offers complete flexibility in the retrieval of records within a chain by providing three methods of chain processing. These methods are illustrated in Figure 14.

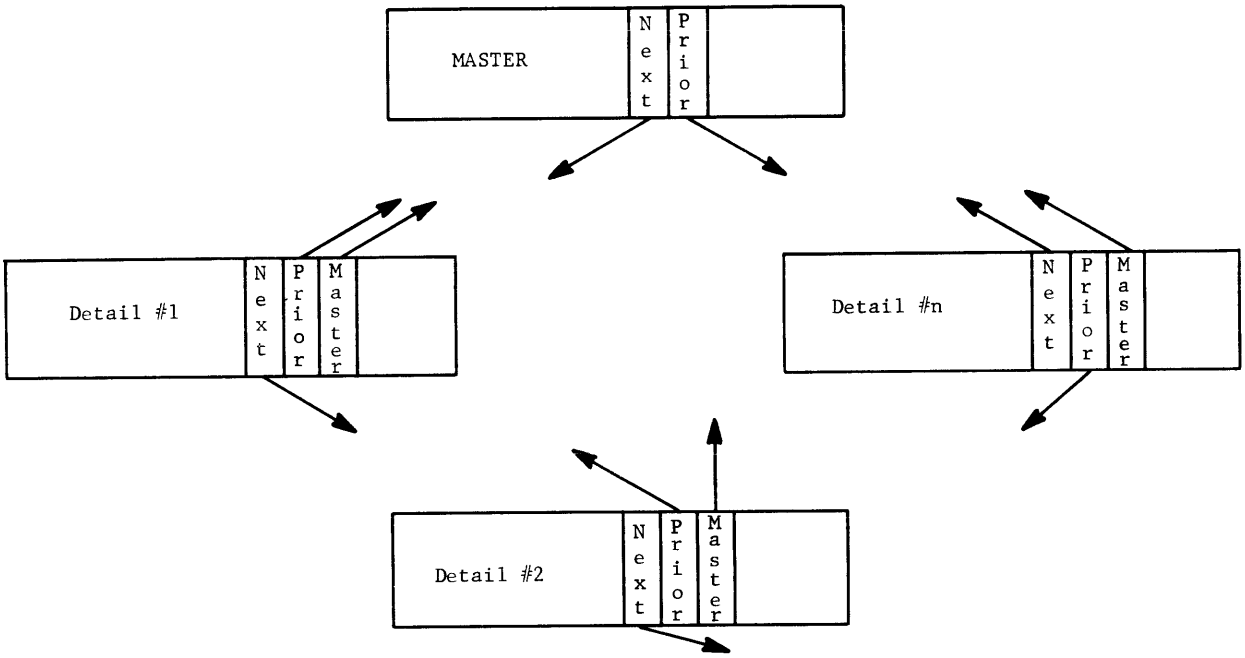


Figure 14. Chain Processing

1. Chain NEXT--The definition of a record in a chain automatically provides the record with a chain-next field. This is the manner in which all chains are constructed. Each record contains a chain-next field which contains the reference code of the next record in the chain.

2. Chain PRIOR (optional)--IDS provides a chain-prior field for all records in a chain when the chain is specified by the systems designer as prior processable. This field contains the reference code of the prior record in the chain. This permits the chain to be processed efficiently in a backward direction.

3. Chain MASTER (optional)--IDS provides a chain-master field for all detail records in a chain when specified in the data description. This field contains the reference code of the master record of the chain. Retrieval of the master record is much faster with this ability to address the master record directly from any detail in the chain. Processing need not access all the detail records in the process of seeking the master.

LINKING A NEW DETAIL RECORD

In order to insert a new detail record in a chain, two steps are required:

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

MASTER RECORD SELECTION

There are two rules under which the master record is selected for a new detail record. These are:

- Select Unique Master--This rule uses the record retrieval criteria, established in the data definition for the master record, to retrieve the particular master record indicated by the data values currently stored in the match control field of Working-Storage.

- Select Current Master--This rule uses the last record processed, of the master record type, as the master record of the new detail.

CHAIN ORDERING

All chains in the Integrated Data Store system are ordered in one of six methods specified by the systems designer with the chain-order clause in the IDS language.

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

The six options of the chain-order clause are:

1. Sorted Within Type--With this option the records of the chain are maintained in sequence within record type, independent of other types.

2. Sorted--With this option the various records of the chain are maintained in a single sequence regardless of the number of record types in the chain. With this option the control fields of the various records must be of identical size.

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

3. First--This option causes the detail to be added as the first detail record in the chain relative to the master record.

4. Last--This option causes the detail to be added as the last detail record in the chain relative to the master record.

5. Before--This option causes the insertion of the detail record just before the current record in the chain. This option may be used only in conjunction with the "Current Master" selection rule.

6. After--This option causes the insertion of the detail record just after the current record of the chain. This option may be used only in conjunction with the "Current Master" selection rule.

PRIME CHAIN

Access time in present disc-type random access memories varies greatly, since it depends on the position of the desired record relative to the record last accessed. The IDS organization of records acknowledges this factor of hardware design and stores new detail records as close as possible to the master record of the chain. When a detail record is specified as a detail in several

chains, a prime chain may be chosen and defined by the systems designer preparing the data description. Selection of a prime chain should be based upon an estimate of the most active chain. Thereafter, when an IDS page is retrieved which contains the master record of a prime chain, it is highly probable that the detail records of that chain will also be contained in that page or a page closely associated with it. The prime chain is the chain used to retrieve a secondary record by the RETRIEVE command, unless specified otherwise by the data description.

DATA STRUCTURE

A special graphic technique called "IDS shorthand" has been developed to display records and their master-detail (chain) relationships.

Its use is particularly important in developing an over-all view when planning an information system. This technique (see Figure 15) 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 Figure 15.

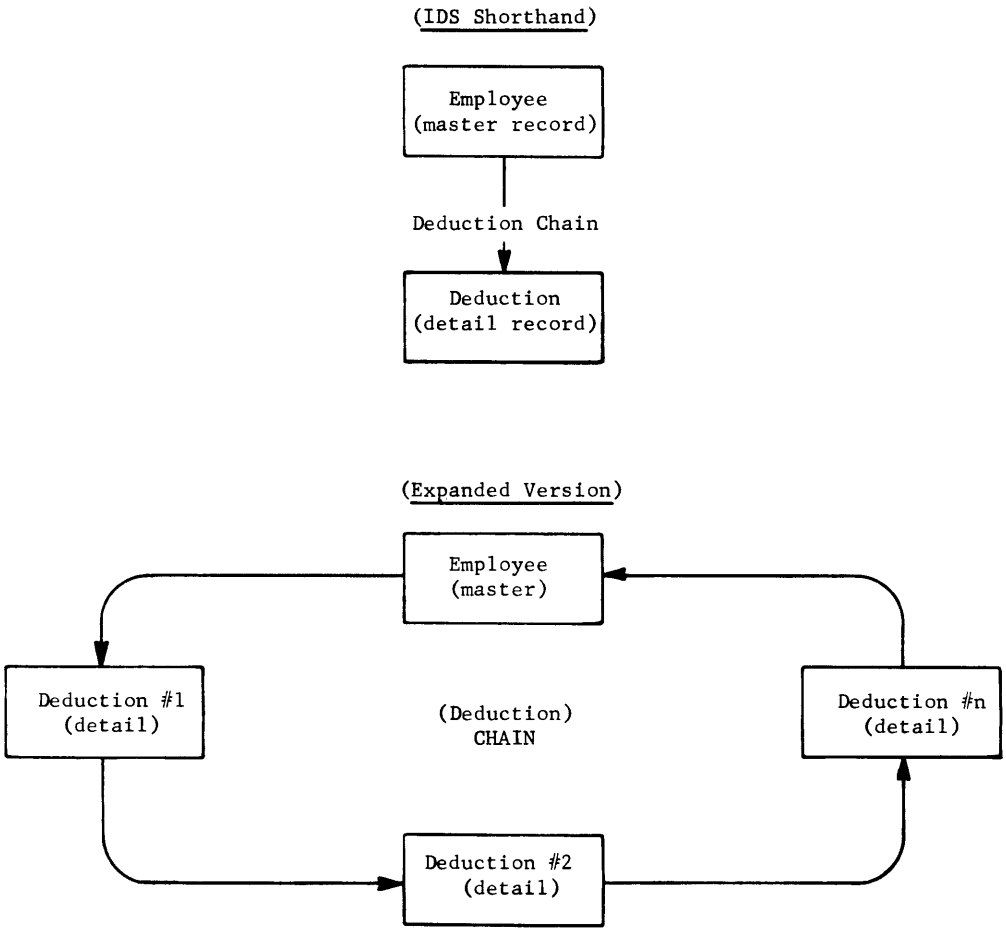


Figure 15. IDS Shorthand

In the foregoing example of IDS shorthand, 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 (Figure 16) shows how a vendor record and a particular order record from the example shown in Figure 17 is normally structured in the IDS system.

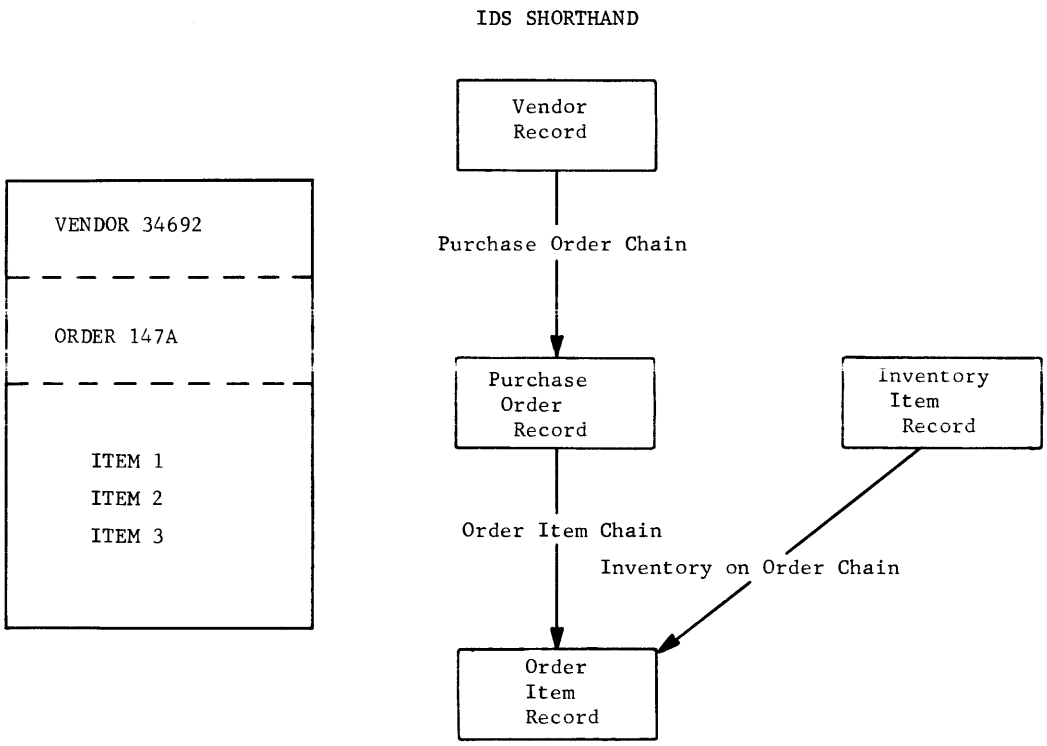


Figure 16. Purchase Order Data Structure

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.

The data structure in Figure 16 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 purchase order chain
2. Thus, the vendor record is only a master.

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

1. It would be a detail in the purchase order chain
2. Each order, in turn, would be the master of an order item chain
3. Thus, the purchase order record is both a master and a detail.

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

1. It would be a detail in the inventory on order chain.
2. It would be a detail in the order item chain
3. Thus, the order item record is a detail in two chains.

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

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

The expanded data structure for the above records is shown in Figure 17.

Figure 16 showed the same data structure in IDS shorthand.

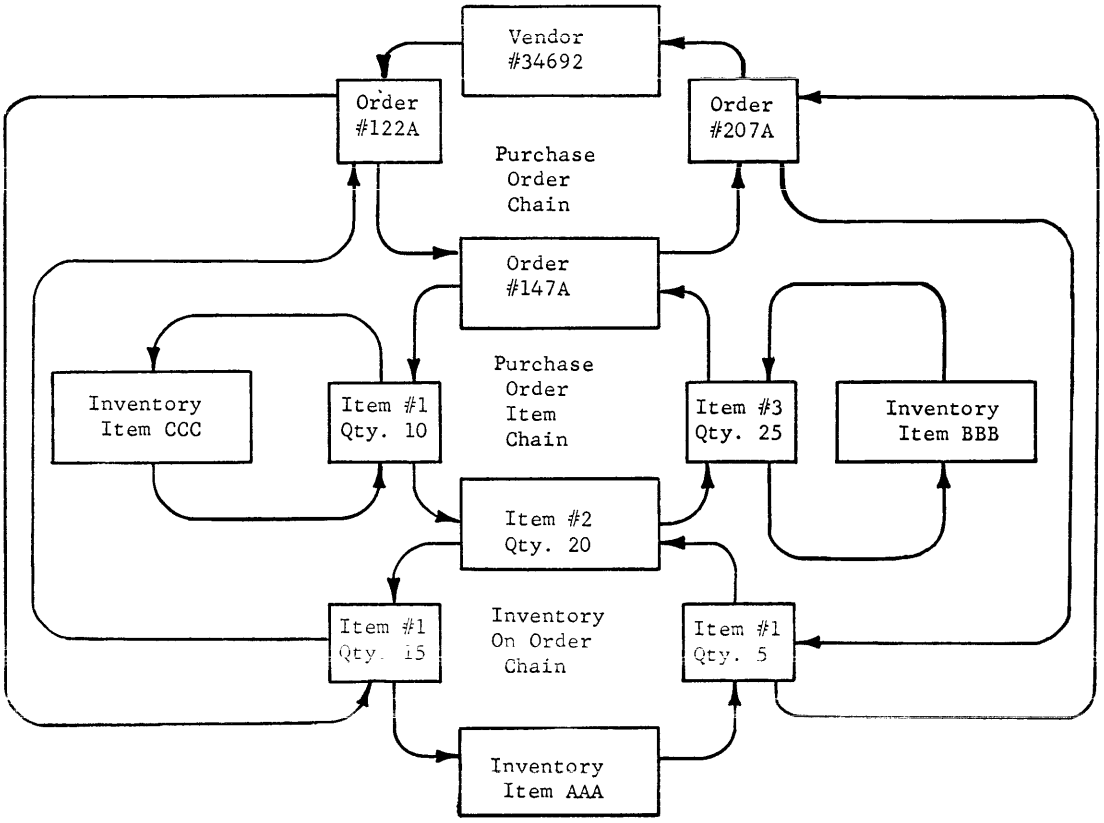


Figure 17. Chain Network

SUMMARY OF DATA STRUCTURES

By using IDS shorthand, very complex data structures may be presented in a condensed and understandable form. Thus, the following examples (Figures 18 and 19) show a quick summary of data structures which are legal and illegal within IDS.

NOTE: A circular definition, as shown in Figure 19, is not allowed.

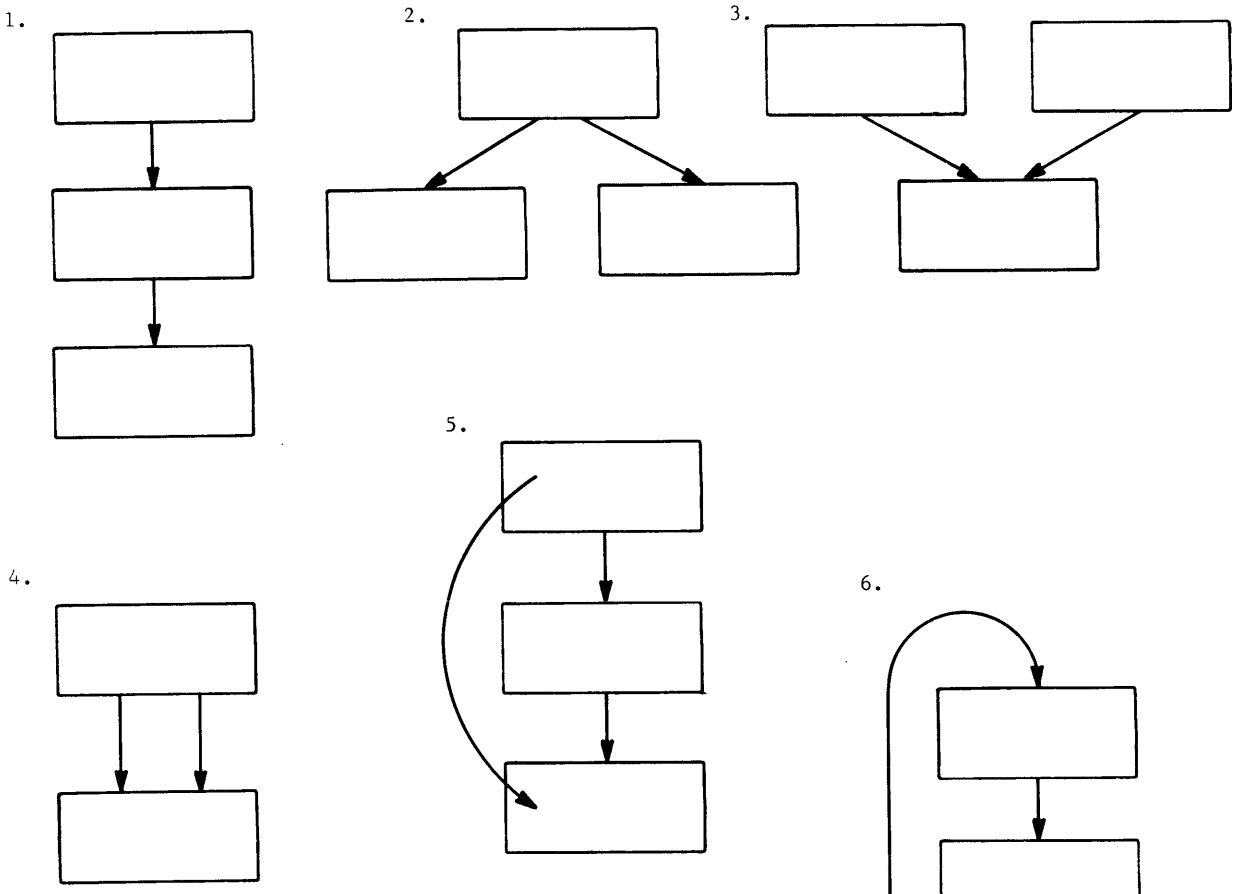


Figure 18. Legal IDS Structures

Figure 19. Illegal IDS Structure

5. IDS PROGRAMMING LANGUAGE

SOURCE LANGUAGE

Introduction

The source language of IDS is an extension of COBOL as implemented for the GE-600 Series product line. 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 IDS usage. These exceptions are mentioned in the following sections on the IDS language.

Identification Division

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

Environment Division

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

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

FILE-CONTROL. SELECT IDS file-name

ASSIGN TO file-code-1.

NOTES:

1. The SELECT IDS sentence must be used only once to identify the IDS data file.
2. Other optional phrases of the SELECT sentence as specified for COBOL should not be used with the SELECT IDS sentence.
3. 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 object program's files with its peripheral device by matching the file codes. (See the GE-625/635 Comprehensive Operating Supervisor Reference Manual, CPB-1002.)

4. When multiple file units are used, an additional restriction is placed on the file code, if an IDS 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 IDS 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 referencedby his program do not use a file code which may conflict with this procedure.

Data Division

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

The IDS Section contains a File Description, Record Description, and Chain Definition as required to describe the complete data file.

FILE DESCRIPTION. The File Description entry provides information regarding the physical characteristics of the IDS data file. The entry is used only for documentation purposes, since the Input/Output Controller module of IDS expects to find these same characteristics stored as a special Environment record within the disc storage unit. (See Appendix A.) As this implies, the disc storage unit must be preconditioned with the Page Header records and the Environment record prior to the execution of any IDS program.

The entry consists of a level indicator, a file name, and a series of clauses which define the physical characteristics of the IDS 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. The format for the complete IDS File Description entry follows.

IDS File Description Entry

FUNCTION: To document information concerning the physical structure of the IDS file.

FORMAT: MD file-name [; PAGE CONTAINS integer-1 CHARACTERS]
[; FILE CONTAINS integer-2 PAGES].

NOTES:

1. The PAGE size (integer-1) specified may be any value up to a maximum of 4096 characters as determined by the particular IDS application. However, the most efficient use of the storage capacity of Mass Storage Device involved should be considered when establishing the page size. When the DS-20 is the device involved, any size other than 240, 480, 960, 1920, or 3840 characters will result in wasted space on the device due to the addressing characteristics of the DS-20.

2. The FILE clause expresses the total physical storage requirements of the IDS 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. The maximum number of pages possible within the IDS page numbering system is 262,144 (2^{18}).

RECORD DESCRIPTION. In general, the format and usage of Record Descriptions required for IDS are the same as those for COBOL Record Descriptions. The exceptions to the normal usage are described below.

The Record Description entries perform three functions:

1. Provide information to IDS regarding the format of each logical record type as it will exist within a page on the external storage device.

2. Define the internal Working-Storage areas which are used to pass data between the user's program and the mass storage device.

3. Provide parameters which will affect the procedures used to store and retrieve data records.

The external format of an IDS record consists of control fields and data fields, as shown in Appendix A. Only the data fields are defined by the Record Descriptions supplied by the IDS Section. The first character of the set of data fields of a record is always the first character following the last chain field. This means that computer word orientation is never applied to external data formats. SYNCHRONIZED, as used in the normal COBOL sense, does not apply to the IDS record formats; if used, it will not affect external formats. In defining the external format of the record, only the level 02 Record Description entries are considered by the IDS Translator. The SIZE and CLASS clauses are the significant elements of the description; however, any of the standard COBOL clauses may be used with the following exceptions: (1) The OCCURS, RENAMES and editing clauses are not applicable to IDS and should not be used at the 02 level within the IDS Section. (2) The COPY clause as specified by COBOL should not be used. A special version of the COPY clause for use with IDS is defined in this manual in the discussion of the unique IDS Record Description clauses.

In addition to defining the external format of the IDS record, the Record Description entries define the Working-Storage areas which will serve as the communication interface between the user's routine and the IDS data file. As a record is retrieved from the storage device, the user can make various data fields of the record available only by causing the record to be moved to Working-Storage. The same process is used when a record is stored. The user must first have initialized Working-Storage with the data fields of the record to be stored.

The IDS Translator causes unique Working-Storage areas to be established for each level 02 entry processed in the IDS Section. These areas are always created so that the first character of the area will begin in character position zero of a computer word. The area created for a given 02 entry may contain subfields which are defined by any number of lower level entries; these entries may be separately referenced by the user's COBOL procedure. The IDS operations, however, can operate only on units of data represented by level 02 Record Description entries. Therefore, any field that is to serve as a control field or any field that may be modified by IDS must be defined as a level 02 entry.

Lower level entries (03-49) may be used to define subfields of the 02 entry with no restrictions. Any legal COBOL clause may be used, as long as it does not contradict the description provided for the 02 entry. For a further clarification of the concept of levels of data description, see the GE-625/635 COBOL Reference Manual, CPB-1007.

Two of the standard COBOL record description clauses allowable at the 02 level do not cause the generation of Working Storage areas. These clauses are the REDEFINES and FILLER clauses.

The REDEFINES clause may be used for its normal purpose of redefinition of an area previously defined. This enables the COBOL procedural statements to reference the Working Storage area by either of its definitions. The field-oriented functions of IDS, (MOVE, MODIFY), however, will respond only to the original definition of the field.

The use of FILLER as a data-name at the 02 level causes the IDS Translator to ignore the entry after taking note of the size of the area involved. No Working Storage is created and that portion of the external format is not available to the user's program. This technique can be used to advantage when the use of core memory is critical.

An additional restriction imposed by the IDS Translator prohibits the use of qualification of those data names at the 02 level of Record Description. Qualification of lower level entries up to the 02 level is permissible. If the same data name occurs as a 02 entry for different record types, the same Working Storage area will be shared by the various records involved.

The third function of the Record Descriptions is to define certain special IDS characteristics of each record type of the data. These special characteristics are defined at the 01 level and are described in detail on the following pages.

IDS Record Description Entry

FUNCTION: To specify the additional characteristics unique to IDS, which may be used to define data records.

FORMAT Option 1.

01 data-name-1 COPY FROM LIBRARY

FORMAT Option 2.

01 data-name-1 TYPE IS integer-1

;RETRIEVAL VIA { data-name-2 FIELD | { data-name-3 | CALC } CHAIN }

[;PLACE NEAR data-name-4 CHAIN]

[;PAGE-RANGE IS integer-2 TO integer-3]

[;INTERVAL IS integer-4 PAGES]

[;AUTHORITY IS integer-5]

NOTES:

1. Each of the above clauses is applicable only at the record (01) level.
2. Data-name-1 must be unique since qualification by file name is not meaningful.
3. All format considerations are as specified for COBOL.
4. The option 1 entry is used 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 data-name-1.

   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 option enables the user to include only those portions of the total data description 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.

TYPE

Type

FUNCTION: To define the Record Type code to be used for reference purposes for each record type within IDS.

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.

Retrieval

FUNCTION: To specify the procedure to be used when this record is to be retrieved by the "RETRIEVE data-name" form of the verb. (See RETRIEVE Verb, Procedure Division)

FORMAT:

```
                                  { data-name-2                  FIELD }
;RETRIEVAL VIA                    { { data-name-3 }              CHAIN }
                                  { { CALC        }                    }
```

NOTES:

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

2. The first of the three variations of this clause (RETRIEVAL VIA data-name-2) provides for records whose reference codes enable them to be retrieved directly. Data-name-2 refers to a field which is defined for this record. The user must supply the IDS generated reference code to retrieve a record defined in this manner. This is done by initializing the contents of the data-name-2 field in Working-Storage. In this case, data-name-2 is the field that is equivalent to the reference code, because it is not contained in the data record as defined but instead is contained as the record's 24-bit reference code. (See Appendix A for the reference code format.) During the retrieval process, the supplied contents of data-name-2 are converted to a 24-bit binary integer and the record corresponding to that value is retrieved. These special considerations associated with the field names require that the field be defined as an 8-digit numeric value (PICTURE IS 9(8) SYNCHRONIZED LEFT.)

3. The second of the three variations of this clause (RETRIEVAL VIA data-name-3 CHAIN) provides for the retrieval of a record which is a detail in the chain named. The master of the chain must first be retrieved and then, by searching the chain, the specific record may be found. Data-name-3 must refer to a defined chain name.

4. The third of the three variations of this clause (RETRIEVAL VIA CALC CHAIN) provides for the retrieval of the record through the use of its defined control field to produce a random number. This random number is equated to a specific page of the data file. The Page Header record is retrieved and the program finds the specific record by searching the CALC chain whose master is the Page Header record.

5. These three RETRIEVAL procedures provide a basis for classification of each record as one of the following:

   Primary - Retrieved directly via reference code
   Secondary - Retrieved via its chain association
   Calculated - Randomized to the page containing the chain which leads to the record.

Subsequent discussions of IDS will refer to records using these terms.

PLACE

Place

FUNCTION: To specify a "special case" procedure to be used when this record is to be stored in the mass storage device.

FORMAT: [;PLACE NEAR data-name-4 CHAIN]

NOTES:

1. Data-name-4 must be a defined chain name and this record must be a detail in that chain.
2. When the record has been defined as a primary or secondary record by the RETRIEVAL clause, the record will be stored physically near its logical position in the chain named.
3. This clause is not meaningful when the record has been defined as a calculated record by the RETRIEVAL clause.
4. When this clause is omitted, primary records will simply be stored in a convenient location. A secondary record will be stored near its logical insert point in the chain defined by the RETRIEVAL clause.

Page-Range

FUNCTION: To provide a method of partitioning the various records of IDS within the total environment.

FORMAT: [;PAGE-RANGE IS integer-2 TO integer-3]

NOTES:

1. Integer-2 and integer-3 represent the first and last page numbers of a series of pages within which records of this type are to be stored. Integer-2 may be of greater magnitude than integer-3. In this case, the range is defined as beginning with the integer-2 page and extending to the last page of the total environment; continuing with page 1 and extending through the integer-3 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. This clause supersedes the storage procedures specified by the PLACE clause.

Interval

FUNCTION: To provide a method to ensure a uniform distribution of a given type of records across the total IDS environment.

FORMAT: [:INTERVAL IS integer-4 PAGES]

NOTES:

1. Integer-4 represents the number of pages which will be skipped when a record is stored relative to the previous record processed of the same type.
2. This clause supersedes the storage procedures specified by the PLACE clause.
3. This clause is only applicable to primary or secondary records as defined by the RETRIEVAL clause.
4. This clause will normally be used only for the initial loading of the file unit.

Authority

FUNCTION: To provide a method to safeguard the data contained in a record from unauthorized reference or modification.

FORMAT: [;AUTHORITY IS integer-5]

NOTES:

1. 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.

COMPLETE CHAIN DEFINITION ENTRY SKELETON

CHAIN DEFINITION. As has been mentioned before, a record normally belongs to at least one and possibly many chains. A Chain Definition entry must exist for each such chain. The Chain Definition entries for a given record must immediately follow the Record Description entries for that record.

The Chain Definition entry consists of a level indicator (98), a chain name, and a series of clauses which define the characteristics of the chain. The complete Chain Definition entry skeleton is shown below followed by a detailed description of the clauses.

Complete Chain Definition Entry Skeleton

FUNCTION: To specify the status of the record as either a master or a detail of a chain, and to specify the characteristics of the chain.

FORMAT Option 1.

```
98 data-name-1 CHAIN MASTER
                                  { SORTED WITHIN TYPE }
                                  { SORTED             }
                                  { FIRST              }
;CHAIN-ORDER IS                   { LAST               }
                                  { BEFORE             }
                                  { AFTER              }

[;LINKED TO PRIOR ] .
```

FORMAT Option 2.

```
                { data-name-1 }
98              {             }   CHAIN DETAIL
                { CALC        }

    [;RANDOMIZE ON date-name-2  [;RANDOMIZE...]]

    [                                 { ARE FIRST     } ]
    [ ;DUPLICATES                     { ARE LAST      } ]
    [                                 { NOT ALLOWED   } ]
```

```
[;{ASCENDING  } KEY IS data-name-3 [: { ASCENDING  } ...]]
  {DESCENDING }                    [  { DESCENDING }    ]

[;ASCENDING RANGE KEY IS data-name-3]

[;SELECT { UNIQUE  } MASTER]
         { CURRENT }

[;MATCH-KEY IS data-name-4 [;MATCH-KEY....]]

[;SYNONYM data-name-5 EQUALS data-name-4]

[;LINKED TO MASTER] .
```

NOTES:

1. Level-number must be 98.

2. Data-name-1 must be unique, since as a chain name it may not be qualified.

3. Either option 1 or option 2 must be used for each chain in which the currently defined record is a part. The clauses shown for option 1 may be used only when the record is the master of a chain. Only one master record may be defined per chain. The clauses shown for option 2 may only be used when the record is a detail of a chain. Any number of record types may be defined as detail records of a chain.

4. An option 2 entry must be used whenever the record has been defined as a calculated record by the RETRIEVAL clause. In this case, the record must be defined as a detail of the CALC chain. (See Note 4 of the RETRIEVAL clause in the description of the Record Description entry.)

Linked-Prior

FUNCTION: To provide an extra chain field for each record of the chain which points to the prior record in the chain.

FORMAT: [;LINKED TO PRIOR]

NOTES:

1. This clause gives the file designer the option of specifying an extra chain field which points to the prior record in the chain. This is desirable for several reasons and also has several disadvantages.

   The most obvious advantage is associated with the use of the "PRIOR OF CHAIN" record specifier. (See Procedure Division, RETRIEVE Verb, Note 5.) Ability to move in both directions is often extremely useful in problem solving. For example, a prior chain field may be used in delinking a record from a chain. Using both the prior and next chain fields, the prior and next records are known and the chain can be easily patched to delink a record. This is extremely helpful if the chain fields of a record are modified and must, therefore, be delinked and relinked into a new chain position. The modify command requires that this relinking be completed immediately. If the prior record's address is not known, the entire chain will be searched until the prior record is discovered, so that the record may be delinked. The deletion process has a delayed delinking scheme which makes the designation of the chain prior field less important.

   The use of chain prior fields has two disadvantages. First, the record size must be increased to include the chain prior field. Second, the linking process is slower, because the chain prior field of the next record must be modified when a new record is inserted.

   Under two conditions, chain prior fields are automatically provided. If the "before current record" ordering rule is selected, all record types in the chain will be assigned chain prior fields for that chain. If the "last detail in chain" ordering rule is specified, the master record (only) will be assigned a chain prior field.

Randomize

FUNCTION: To specify those fields of a calculated record which should be used to locate the record in the data file.

FORMAT: [;RANDOMIZE ON data-name-2 [;RANDOMIZE....]]

NOTES:

1. This clause must be used when the Chain Definition entry is for a CALC chain.
2. Data-name-2 must be a field contained in the record being defined.
3. The randomizing routine of IDS uses as many fields as are specified in determining the page in which the record is to be located.

CHAIN-ORDER

Chain-Order

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

FORMAT:

```
                                  { SORTED WITHIN TYPE }
                                  { SORTED             }
                                  { FIRST              }
;CHAIN-ORDER IS                   { LAST               }
                                  { BEFORE             }
                                  { AFTER              }
```

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 content of the defined sort control fields of the detail records. When the (SORTED WITHIN TYPE) option is used, records of the chain are maintained in sequence within record type, independent of other types. When the (SORTED) option is used, the various records of the chain are maintained in a single sequence regardless of the number of record types in the chain. In this case, the sizes of the control fields of the various records must be identical.

3. The last four options 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 first detail in chain relative to the master record.

   LAST - Insert last detail in chain relative to the master record.

   BEFORE - Insert record just before or prior to the current record of chain.

   AFTER - Insert record just after the current record of chain.

Duplicates

FUNCTION: To specify whether or not duplicate records may exist in the chain.

FORMAT:

```
[ ;DUPLICATES { ARE FIRST  }  ]
              { ARE LAST   }
              { NOT ALLOWED}
```

NOTES:

1. This clause must be used whenever 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 duplicates.
3. When the NOT ALLOWED option is specified, an error condition will exist whenever insertion of a duplicate record is attempted.
4. This clause need not be used when the chain is the CALC chain. In this case duplicates are never allowed by IDS.

Sort Key

FUNCTION: To specify those data fields which will control the sequence of the detail record of the chain named.

FORMAT Option 1.

```
[ ;{ ASCENDING  }   KEY IS data-name-3 [ : { ASCENDING  }... ] ]
  { DESCENDING }                         { DESCENDING }
```

FORMAT Option 2.

```
[;ASCENDING RANGE KEY IS data-name-3]
```

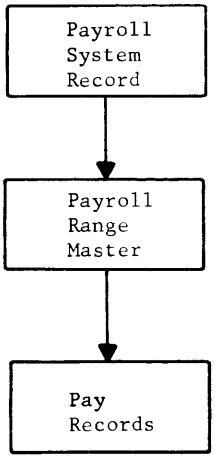
NOTES:

1. Data-name-3 must be a field contained in the detail record being defined.

2. This clause must be used in one of its two forms whenever the chain has been defined as a SORTED or SORTED WITHIN TYPE chain.

3. When multiple sort control keys are required to define a chain sequence, the various clauses required must be presented in sequence from major control field to minor, thus establishing the sort level of each field.

4. The option 2 variation of this clause implies an additional function for the control field named by data-name-3. In addition to controlling the sequence of the detail records within the chain, the value of the field also delimits the upper maximum of the range of values the field may assume. When a record defined in this manner is retrieved using the "RETRIEVE data-name-1" record specifier, the value supplied in Working Storage for data-name-3 is compared with the value of the field in each detail of the chain. A match occurs whenever the value of the field in Working Storage is equal to or less than the value actually stored in the record.



This type of record is used principally for segmenting a long sorted chain to improve the access time to specific records of the chain. An example of this data structure and its application is shown here.

In this example, the Pay Records are each identified by a control field called PAYNO which may contain a value of 1 to 10,000. These records are to be segmented by establishing 100 Payroll Range Masters. Each such record controls a range of 100 payroll numbers.

The Range Masters contain a control field whose initial value is 100, 200, ... or 10,000. The Range Master records are sequenced in the Range Chain in accordance with their control fields. When a Pay Record is stored into the system, a master record of the Payroll Chain must first be selected. The selected record would be the first Payroll Range Master in the Range Chain that has a control field equal to or higher than that of the new Pay Record. A Pay Record with a PAYNO of 2126 would select as its master the Payroll Range Master with the control field value of 2200.

The option 2 clause above identifies the fact that an exact match is not required to select the master of the Payroll Chain, in addition to serving its normal function as a sort control key specification.

SELECT

Select

FUNCTION; To specify the criteria for selecting the specific master record from many master records of a given type.

FORMAT:

```
:SELECT { UNIQUE  } MASTER
        { CURRENT }
```

NOTES:

1. One of the two forms of this clause must be used, unless the record is defined as a detail of the CALC chain. This definition implies that the Page Header record is the unique master.

2. When using the UNIQUE option, the master is selected by matching the data field values in the master record with those supplied in Working Storage for the new detail. In the purchase order example (see Chapter 4), the order chain exemplifies this form of master record selection. As a new order record is entered into the system, the vendor code (supplied with the data fields) of the order record causes the appropriate vendor record to be retrieved. This vendor record is the master of the chain into which the new detail (order record) will be inserted.

3. When CURRENT is specified, the master selected is the current master and the detail is inserted in its chain. The item chain in the purchase order example demonstrates this type of master selection. Since the item records represent all of the items of an order and since they are stored as a batch at the same time as the order record is stored, the current order record is the master to which the items should be associated.

Match-Key

FUNCTION: To specify those data fields which must be initialized in Working-Storage to enable unique identification of the detail record defined by this level 98 entry.

FORMAT: ;MATCH-KEY IS data-name-4 [;MATCH KEY ....]

NOTES:

1. This clause must be used only when the SELECT UNIQUE clause is used.
2. The fields named by this clause depend upon the RETRIEVAL clauses specified for each of the higher level master records defined for the hierarchical structure which includes this detail record.
   - When a master record is defined as a primary record, the data field which is equivalent to the reference code of the master must be named as a MATCH-KEY for this detail record.
   - When a master record is defined as a secondary record, each of the data fields which control the retrieval of that record must be named as a MATCH-KEY field for this detail record.
   - 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.
3. The use of this clause effectively defines the fields named as though they were contained in the current detail, although they are actually contained only in the master records of chain hierarchy.
4. This clause is not required when the record is defined as a detail of the CALC chain.

SYNONYM

Synonym

FUNCTION: To specify an alternate name for a field defined as a MATCH-KEY field.

FORMAT: [;SYNONYM data-name-5 EQUALS data-name-4]

NOTES:

1. Data-name-4 must be a field that is defined as a MATCH-KEY field for that record. Data-name-5 is a synonym or alternate name of that field.

2. The following example illustrates the use of this clause. The clause may be used when a given record is defined as a detail in two chains each having the same record type as its master. In this case, the detail must be linked into each of the chains based upon the contents of its defined MATCH-KEY fields. Since a common record type is the master for both chains, the match control fields are actually the same for each chain. A separate field is set up to hold the alternate match field value for each match control field. The SYNONYM clause equates the alternate field and its equivalent.

   When this clause is used, a separate Working-Storage area is established for the field identified by data-name-5. In the example above, the user must initialize this alternate Working-Storage area with the MATCH-KEY field which controls the chain for which the SYNONYM clause is used.

Linked-Master

FUNCTION: To provide an extra chain field for each detail record of the chain which points to 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 to the master of the chain and, thus, eliminating the need for processing all of the intervening detail records serially.

SAMPLE CODING. The following example describes the records shown in Figure 16 in IDS 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 ALPHANUMECIC.
            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.
        02  ITEMNO  SIZE IS 2 NUMERIC
        02  MATLIDENT  SIZE IS 18 AN.
        02  ORDER-QTY  PICTURE IS  9999V9.
            98  ITEM CHAIN DETAIL  SELECT CURRENT MASTER
                ASCENDING KEY IS ITEMNO  DUPLICATES NOT ALLOWED.
```

Procedure Division

The IDS procedural statements store and retrieve data with respect to the mass storage device. 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 IDS controller accomplishes these functions.

The communication interfaces between the IDS controller and the balance of the COBOL Procedure Division are the Working-Storage areas which are established for each record type defined by the record description entries of the IDS Section. All COBOL references to data from the IDS file are to these Working Storage areas.

The procedural statements of IDS may appear anywhere in the context of the COBOL Procedure Division. An IDS sentence is always preceded by ENTER IDS. The sentence may contain any number of IDS statements and must be terminated by a period. The IDS sentence may be further delimited by the statement ENTER COBOL. However, this is not required, since the period following the sentence is a sufficient delimiter. All other format rules for the COBOL Procedure Division apply to the IDS language statements. A paragraph name or section name may be assigned to an IDS sentence in a manner consistent with normal COBOL format.

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

STORE

Store

FUNCTION: To place a record into the IDS data file and to establish any chain fields which may be required.

FORMAT: STORE data-name-1 RECORD

NOTES:

1. Data-name-1 must be one defined for a record level (01) entry of the IDS Section of the Data Division.

2. When this verb is used, the following is assumed: (1) The Working-Storage area for this record has been initialized with an exact image of the record. (2) Any other control fields required to provide unique identification of the master records of the defined chains which include data-name-1 have been initialized in their respective Working-Storage areas.

3. The record is placed into the file as defined by the PLACE or RETRIEVAL clauses of the Record Description entry.

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

5. The record stored is recorded as the CURRENT record of its type and the CURRENT record in each chain in which it is a master or detail.

6. If the storage process creates a duplicate record in violation to any DUPLICATES NOT ALLOWED clause, or if the unique or range master selected, cannot be retrieved, the storage process is terminated with all linkages restored as before and an error condition is noted.

7. When a primary record is stored, the reference code assigned to the record is moved to the Working-Storage field which is equivalent to the reference code for the record.

Retrieve

FUNCTION: To cause a record to be made available in the memory buffers.

FORMAT:

```
           ┌ ┌ data-name-1            ┐                                   ┐
           │ ┤                        ├ RECORD                            │
           │ └ CURRENT data-name-1    ┘                                   │
           │ ┌ NEXT   ┐                                                   │
RETRIEVE   ┤ ┤ PRIOR  ├   RECORD OF data-name-2 CHAIN                     ├
           │ └ MASTER ┘                                                   │
           │   EACH AT END GO TO procedure-name-1                         │
           │   DIRECT                                                     │
           └                                                              ┘
```

NOTES:

1. The various options of the RETRIEVE verb are hereafter referred to as the record specifiers.
2. Data-name-1 must be the name of the record (01) level entry defined in the IDS Section of the Data Division.
3. Data-name-2 must be the name of a chain as defined by a level 98 entry of the IDS Section of the Data Division.
4. Regardless of the record specifier used, this verb causes the record referenced to be retrieved and made available in the memory buffer. This action may or may not require that a page be transmitted from the mass storage device, since the record may already be in memory. No other action, such as moving the record to Working Storage, is implied by this verb.
5. Of the seven record specifiers which may be used with the RETRIEVE verb, two may be classified as absolute. This means that only one record will satisfy the retrieval specification, whenever one of these two specifiers is executed.

   RETRIEVE data-name-1 RECORD. The record retrieved depends on the RETRIEVAL clause defined as a part of the level 01 entry in the IDS Section and upon the value contained in the control fields of Working-Storage which uniquely identify the record.

   RETRIEVE DIRECT. The record to be retrieved is identified by the reference code stored in a communication cell named DIRECT REFERENCE. The user is responsible for initializing the communication cell prior to the execution of this command.

The other five record specifiers may be classified as relative, since the actual record retrieved is a function of what has transpired previously.

RETRIEVE { NEXT / PRIOR / MASTER } RECORD OF data-name-2 CHAIN.

In this case, 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, all intervening records are ignored and 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 type of chain.

RETRIEVE CURRENT data-name-1 RECORD. This record specifier instructs the system to retrieve the last record of the type specified that was processed by any STORE or RETRIEVE verb. If the last record processed had been deleted, it, of course, would not be available and an error condition would exist.

RETRIEVE EACH. This record specifier provides for a serial search of the mass storage device beginning with the reference code contained in a communication cell called FIRST-REFERENCE. This command retrieves the first record found, unless the reference code of the record retrieved is greater than the contents of a communication cell called LAST-REFERENCE. At this point the system exits according to the AT END statement. The user must initialize the communication cells with the appropriate reference codes. The user may retrieve all of the records in the file through the repeated use of this record specifier. As a record is retrieved, the reference code value contained in the FIRST-REFERENCE cell is incremented by one, so that subsequent use of this verb is initialized.

6. The reference code of the record retrieved is placed in the communication cell named DIRECT-REFERENCE after the retrieval process is complete.

7. The record retrieved is recorded as the CURRENT record of its type and the CURRENT record in each chain in which it is a master or detail.

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

IDS IMPERATIVE STATEMENTS. The imperative statements included in this section are provided as a part of the IDS language to extend the function of the basic STORE and RETRIEVE verbs. Four of these statements apply only to the RETRIEVE verb: the GO and PERFORM statements may be used with either verb.

When these statements are used, they must occur in the order in which they are to be executed. They may be contained within the sentence beginning with the basic verb and ending with a period, or they may be used as separate sentences to accomplish their intended result. In the latter case, the statements must be preceded by ENTER IDS.

The specific formats of these statements and detailed discussions of the restrictions and limitations associated with each appear on the following pages.

Move

FUNCTION: To cause the record last processed to be moved from the buffer to Working Storage or to cause selected fields of the record to be moved.

FORMAT: ;MOVE [data-name-1 [, data-name-2 ....]]

NOTES:

1. The implied source of the MOVE is the record last processed and available within the buffer.
2. This statement must be used before any reference can be made to the data in the record.
3. When the statement includes the list of fields identified by data-name-1, data-name-2, etc., only those fields are moved to Working Storage. Otherwise, all fields are moved.
4. When the record is moved to Working Storage, each data field as defined by the 02 entries of the Record Descriptions, is unpacked into an integral number of words as if the fields had been defined as SYNCHRONIZED. These fields are SYNCHRONIZED LEFT unless SYNCHRONIZED RIGHT was specified in the 02 entry.

Head

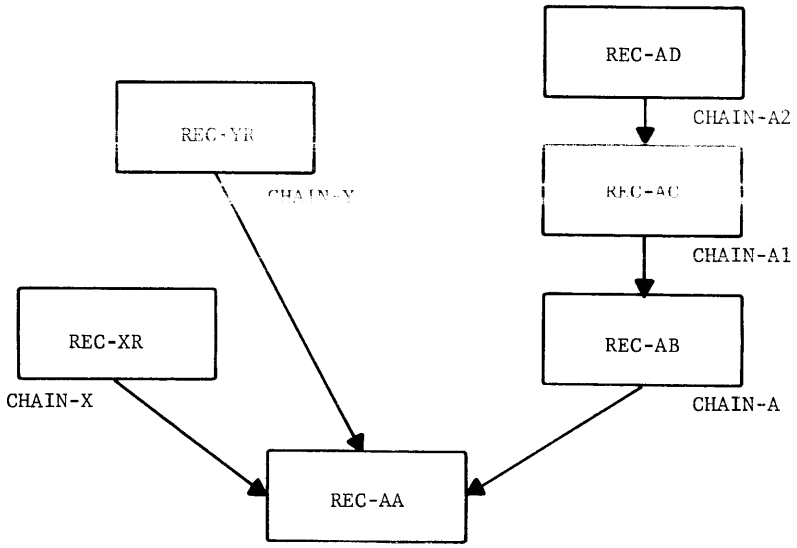
FUNCTION: To enable the automatic retrieval and move to Working-Storage of the master record of the chain specified.

FORMAT: ;HEAD data-name-1 CHAIN [;HEAD...]

NOTES:

1. Data-name-1 must be the name of the chain as defined by a level 98 entry of the Data Division. Further, some record within the data-name-1 chain must have been previously referenced by an IDS statement.

2. This statement can best be illustrated with the example of a data structure shown below in IDS shorthand.



In this case, assume that REC-AA was the record initially retrieved by the RETRIEVE verb. At this point, three chains include REC-AA and, therefore, three possible master records may be referenced by the HEAD statement. Notice, however, that once HEAD has been used to reference CHAIN-A, the next higher level CHAIN-A1 can be referenced.

3. This statement includes an implied move of the record retrieved to Working-Storage.

4. After execution of this statement, the master records retrieved are the CURRENT records of their respective types. They become the CURRENT records in each chain in which they are defined as details. However, they are not the CURRENT records in chains in which they are defined as master records. In those chains, the detail record which leads to the master is the CURRENT record.

5. Note that the function of the statement is very similar to that of the RETRIEVE MASTER RECORD statement, except for the manner in which CURRENT of chain is maintained. (Note 4.)

Modify

FUNCTION: To modify the contents of a field of the record last processed and to maintain any chains which may be controlled by the modified field.

FORMAT: ;MODIFY data-name-1 [, data-name-2 ....]

NOTES:

1. This statement causes the contents of Working Storage to replace the contents of the fields specified in the record in the buffer.

2. Data-name-1 may be any field contained in the record specified by the RETRIEVE verb. In addition, data-name-1 may be a field that has been defined as a MATCH-KEY field by the detail Chain Definition (level 98) entry associated with the record retrieved.

3. When the field to be modified is a MATCH-KEY field, the association of the detail record with its master is changed. The record is delinked from its old master, its new master is retrieved, and the record is linked to its new master according to the CHAIN-ORDER clause for the chain. Note that, in this case, the field specified does not change, but, instead, the record becomes a part of a different chain.

4. A field in the record retrieved that is defined as a sequence key field may also be changed. In this case, the field is changed, the record is delinked from its master, and the record is relinked to the master in accordance with the new value of its sequence key field.

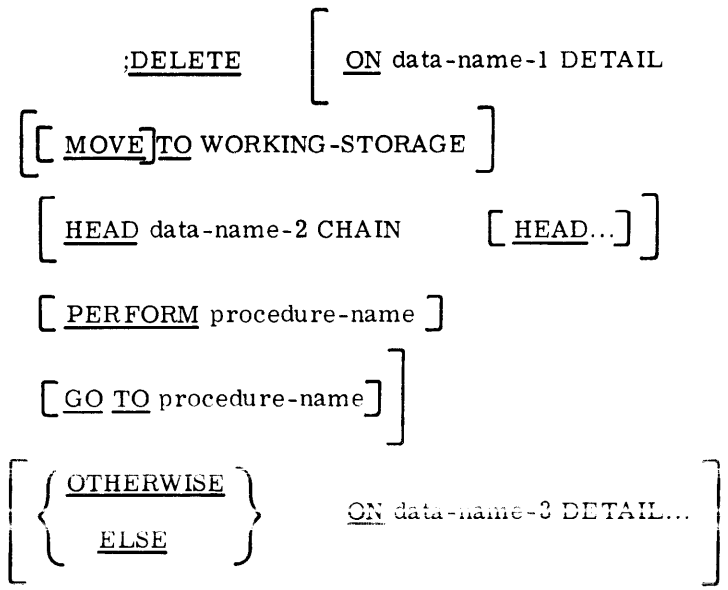
5. Note that when the field to be modified is a control field such as those mentioned in Notes 3 and 4, a conflict in the use of Working-Storage can occur. When the original retrieval action was accomplished by the "RETRIEVE data-name" record specifier, the appropriate Working-Storage fields for the various control fields had to be initialized with the values currently contained in the data structure. Since the modify function assumes the change values are in Working-Storage, a conflict is apparent. To resolve this problem the user should perform the following functions:

   1) RETRIEVE data-name RECORD
   2) Initialize Working-Storage with change values
   3) MODIFY file name

Delete

FUNCTION: To delete the record retrieved with all of its dependent detail records and to optionally perform certain functions when specified detail record types occur during the deletion process.

FORMAT:



NOTES:

1. The implicit object of the DELETE statement is the record last retrieved by the RETRIEVE verb.

2. The deletion process deletes a record only when there are no dependent details in its chains. When details are present, the system first attempts to delete the dependent detail records. Since the hierarchical data structure of IDS may involve many levels of detail records, this statement assumes powerful capabilities and should be used with due consideration .

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

4. The conditional statement "ON data-name-1 DETAIL" is used only when it is necessary to interrupt the deletion process when a dependent detail of the type named by data-name-1 is encountered. When the statement is used, various imperative statements immediately following 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 this case, control is not returned to the deletion process. When the record encountered is not the type named by data-name-1, it is compared with the type named by data-name-3. The reserved words OTHERWISE or ELSE separate the tests for different record types that may be encountered. A record encountered which does not match any of the specified record types is deleted in the normal manner.

5. As a record is deleted it is not implicitly moved to Working-Storage.

GO

Go

FUNCTION: To depart from the normal in-line sequence of procedures.

FORMAT: :GO TO procedure-name-1

NOTES:

1. Procedure-name-1 may be any COBOL or IDS procedural paragraph in the Procedure Division.
2. When this statement is encountered within the IDS sentence, all subsequent statement are bypassed and control is transferred to the procedure named.

Perform

FUNCTION: To depart from the normal in-line sequence of procedures in order to execute a specific procedure and then return to the normal sequence.

FORMAT: ;PERFORM procedure-name-1

NOTES:

1. Procedure-name-1 may be any COBOL procedural paragraph in the Procedure Division.
2. For other details concerning the PERFORM statement see the GE-625/635 COBOL Reference Manual, CPB-1007. Only the simple PERFORM (option 1) is recognized within an IDS sentence.
3. The procedure to be executed may not contain any IDS statement.

IDS CONDITIONAL STATEMENTS. The conditional statements of IDS are a logical extension of the basic STORE and RETRIEVE verbs. Generally, they involve the key word IF, followed by the condition to be tested, followed by the imperative statements to be performed.

IDS conditional statements are of two general forms· either form may appear in the string of statements following a basic verb.

The specific formats of these statements and a discussion of their restrictions and limitations are outlined below.

FORMAT Form 1.

```
;IF data-name-1 RECORD { statement-1 [;statement-2...] }
[ { OTHERWISE } { statement-3 [;statement-4...] } ]
  { ELSE      }
```

NOTES:

1. This form may only follow a RETRIEVE DIRECT, RETRIEVE EACH, or RETRIEVE {NEXT / PRIOR} RECORD OF data-name CHAIN.
2. Statement-1, statement-2, etc., may be any imperative statements. Statement-3, statement-4, etc., may be any imperative statements, or a conditional of this form (form 1).
3. This form of conditional is an implicit comparison of the record type just retrieved with the record type named by data-name-1. If the record type of the record last processed equals the record type named, the condition is true and statement-1, statement-2, etc., will be executed in order. Then control is transferred to the next sentence. If the condition is false statement-3, statement-4, etc., will be executed. If the clause beginning with the key words OTHERWISE or ELSE is omitted and the condition is false, control will be transferred to the next sentence.

FORMAT Form 2.

```
;IF ERROR statement-1 [ { OTHERWISE } statement-2 [;statement-3...] ]
                       { ELSE      }
```

NOTES:

1. This form may only follow a STORE or RETRIEVE verb or a MODIFY, DELETE, HEAD or MOVE imperative statement.
2. Statement-1 may only be a GO TO or a PERFORM imperative. Statement-2, statement-3, etc., may be any imperative statement appropriate to the basic verb, or a conditional of form 1, if appropriate.
3. This form signals the occurrence of any logical or physical error as a result of some IDS command. The specific errors which may occur are a function of the command executed. A detailed coding scheme for identifying errors is defined in Appendix C. The user program may determine the type of error by referring to a communication cell named ERROR-REFERENCE.

Close IDS

FUNCTION: To force the writing to the file unit of any pages in memory which have been modified.

FORMAT: CLOSE

NOTES:

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

Open IDS

FUNCTION: To initialize the processing of the IDS data file.

FORMAT:

```
OPEN [ FOR { RETRIEVAL } ]
           { UPDATE    }
[WITH AUTHORITY-KEY integer-1]
```

NOTES:

1. This statement must be executed before any other IDS verb is executed.

2. When the IDS file is opened for RETRIEVAL, the STORE, DELETE, and MODIFY statements of IDS are not operative. An attempt to use these statements under these conditions results in an error condition during program execution. When the FOR { RETRIEVAL / UPDATE } clause is not used, UPDATE is assumed by IDS.

3. 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 $4095_{(10)}$.

   When this clause is used, each reference to a record of the IDS file involves a match of the AUTHORITY value defined for the record with the AUTHORITY-KEY supplied. When a valid match occurs, the IDS 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.

   The exact details of the matching process may be modified with each installation, to suit individual requirements.

6. OPERATIONAL CHARACTERISTICS

GECOS SYSTEM CONTROL

The IDS Translator is a "system program" which may be called by GECOS. The $ IDS control card is used to call the IDS Translator from system storage. This control card is identical to the COBOL control card except for the contents of columns 8-12 which contain "IDS" instead of "COBOL." The various processing options which may be specified are identical with those specified for COBOL. (See GE-625/635 Comprehensive Operating Supervisor, CPB-1002A for details of the COBOL control card.)

At the time of allocation for the IDS Translator, sufficient resources (memory and peripheral devices) are allocated to provide for COBOL. When the translator has completed its function, it passes control to COBOL using the GECOS entry point GECALL. If the translator encounters a serious error, COBOL is not called and the job is terminated after appropriate explanation has been placed on the "execution report" associated with the job. Figure 20 is a flow diagram of the compilation process of an IDS program.

OBJECT PROGRAM EXECUTION

The object program created from the IDS source language consists of a modular set of subroutines which interpretively execute the store and retrieve commands. GELOAD loads these subroutines as a result of the calls generated by the compilation process.

Because of the interpretive mode of execution, the complete data description of the IDS data file, also generated by the translator, must be available to these routines.

The Input/Output Controller module of IDS controls the mass storage device. It transfers pages in and out of the core in response to commands to retrieve a specific record, or to store a specific record. In order to minimize the seek and transfer time, an inventory of pages is maintained in memory. Increasing the number of pages stored in core increases the possibility that the one needed next will already be there. To further improve the possibility of finding the page desired in core, the Input/Output Controller keeps track of the sequence of page utilization and holds the most recently active data pages in core. Pages which are infrequently accessed are retired from core as others are called in. The Input/Output Controller notes which pages have been modified and writes only the modified pages back to the mass memory device. Sufficient core memory must have been allocated to the object program to ensure a minimum of two pages retained in memory, but usually a more practical number is four or more.

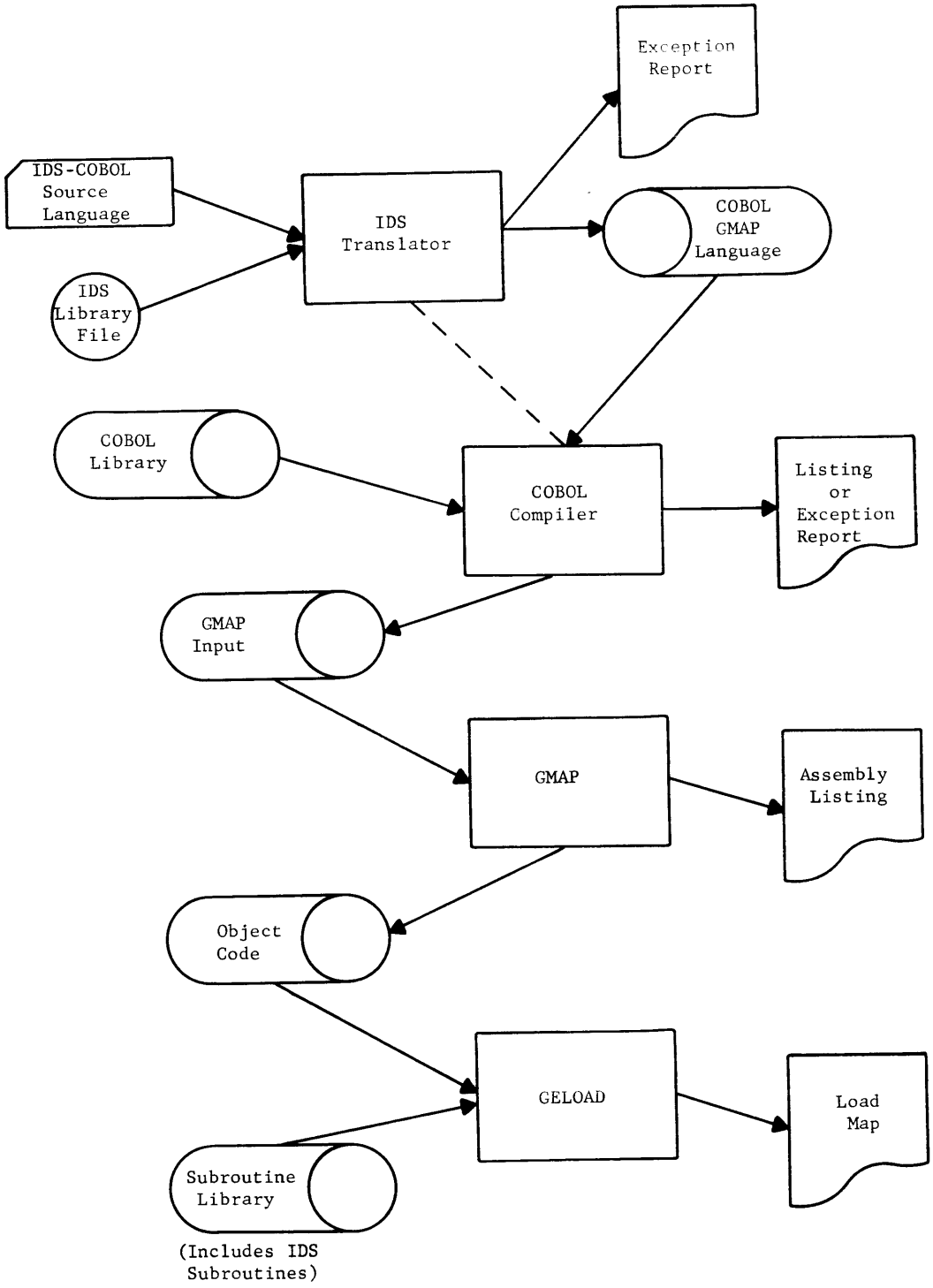


Figure 20. IDS Compilation Process

ASSIGNMENT OF IDS BUFFERS

The buffer area for the Input/Output Controller module of IDS is established when the IDS file is opened. The user defines the size of this area (that is, the number of buffers established) by employing one of the two following procedures.

1. A labeled common area (.QAREA) may be used. Its size is specified by a GELOAD control card as shown below:

1	8	16
$	USE	.QMAX/1/,.QAREA/n/,.QMIN/1/

The control card must be inserted in the object deck so that GELOAD will encounter it prior to its retrieval of the IDS routine from the library. (See CPB-1002.)

The value supplied for "n" must be equal to the number of words to be used for buffers. When the file is opened, the size of .QAREA will be determined and divided into "m" buffers, where "m" is the integral portion of the area size divided by page size. The resulting value of "m" must be at least 2 and is limited to 50 buffers.

2. If the above control card procedure is not used, the OPEN routine of IDS will attempt to use the area of memory not assigned to other program segments. This "available area" is defined by the contents of word $31_{(10)}$ of the program as initialized by GELOAD during the loading process. As in the procedure above, the "available area" is then divided into some number of buffers depending on the size of the area. Again a minimum of 2 and a maximum of 50 buffers is established.

When the IDS buffers are established in the latter manner, the user must insure that the area to be used is not concurrently used by any other segment of the program. For example, a program which calls upon IDS and SORT/MERGE would cause a conflict in the use of memory, since both routines would attempt to use the available area. To guard against such a possibility, the OPEN routine of IDS determines that the available area is actually available and sets a bit in the GECOS "switch word" indicating the assignment of the area to IDS. If some other routine has previously made use of the area, the program is aborted. The use of the labeled common area, as outlined for procedure 1 above, would, of course, eliminate any potential problem of this type.

FILE UNIT INITIALIZATION

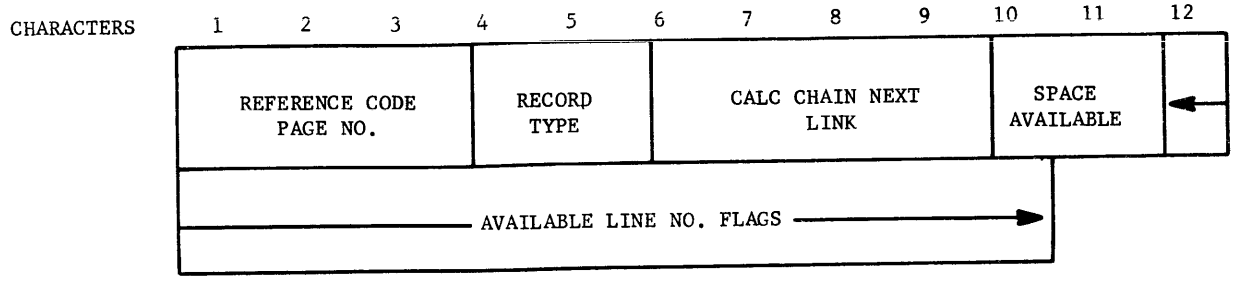
Prior to the operation of any IDS program, the disc storage unit must have been initialized with a Page Header record as the first record of each page in the IDS data environment. See Appendix A for the detail format of the Page Header record.

In addition to the Page Header record, a single Environment record must be stored as the first record of page 1 of the data environment. The Environment record is used each time the IDS file is opened to accomplish necessary initialization of the Input/Output Controller module of IDS.

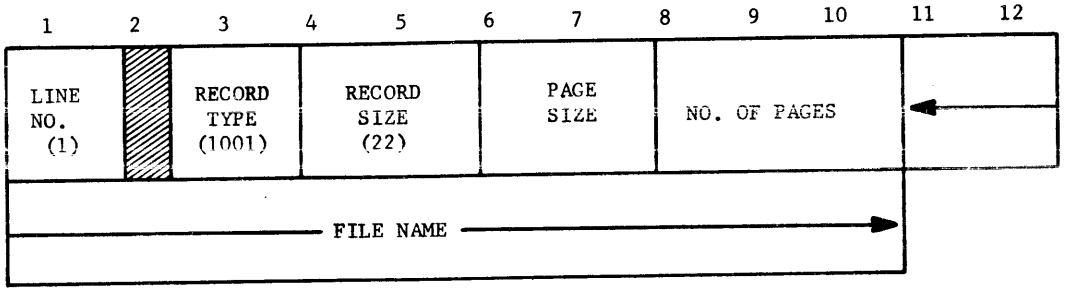
A generalized utility program will be available to accomplish this disc storage unit initialization.

APPENDIX A
IDS RECORD FORMATS

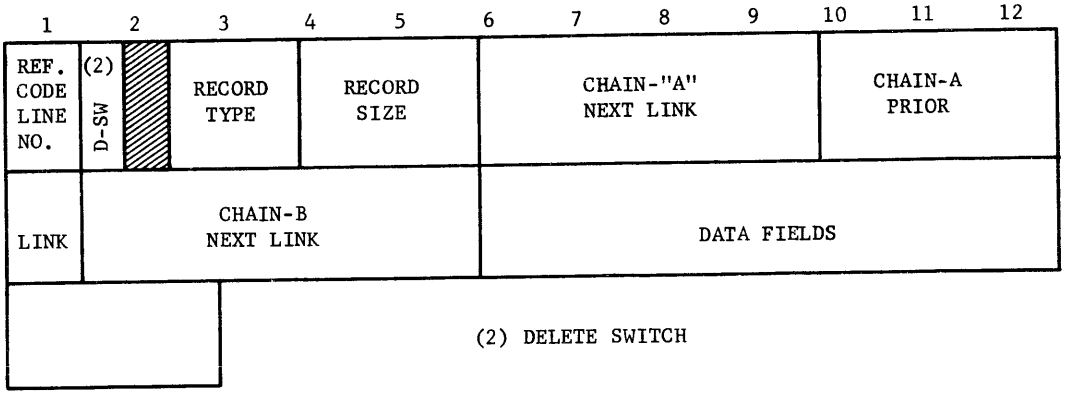
PAGE HEADER RECORD FORMAT



ENVIRONMENT RECORD



TYPICAL DATA RECORD FORMAT



NOTE: Reference Code is a composite of the Page Numbers from the Page Header record and the unique Line Numbers from the data record.

APPENDIX B
IDS RESERVED WORDS

IDS uses all the reserved words specified for COBOL. In addition, it employs the reserved words listed below. The user must avoid using words on both these lists for data-names.

ALLOWED
AUTHORITY
CALC
CHAIN
CHAIN-ORDER
CURRENT
DELETE
DIRECT
DIRECT-REFERENCE
DUPLICATES
EACH
ERROR-REFERENCE
FIELD
FIRST-REFERENCE
HEAD
IDS
INTERVAL
LAST-REFERENCE
LINKED
MASTER
MATCH-KEY
MODIFY
NEAR
PAGE-RANGE
PRIOR
UNIQUE
RANDOMIZE
RECORD-TYPE
RETRIEVAL
RETRIEVE
SORTED
STORE
SYNONYM
VIA
WITHIN

APPENDIX C
IDS ERROR CONDITIONS

Two types of error conditions may occur during the execution of an IDS program. The first of these, involves those situations which are data dependent and must be anticipated by the procedural logic of the program. These errors are returned to the user program and may be tested by reference to a cell called ERROR-REFERENCE. The following codes are in this category.

R01	The retrieval of a record depends upon the selection of the current master of a given type. That record has not been retrieved or has been deleted.
R02	Record cannot be retrieved because the record or one of its masters has been deleted.
R03	The primary record retrieved is not the same record type as that specified in the Record Definition.
R04	Attempt to retrieve a record which does not exist within the data structure identified by supplied control fields.
R05	Attempt to retrieve the current record of a given type when that record type has not been previously retrieved or has been deleted.
R06	Attempt to retrieve DIRECT when DIRECT-REFERENCE is zero.
R07	Attempt to retrieve a record which has been deleted and its Delete switch is set.
R08	Reference code of the record to be retrieved is not found within the specified page.
R09	Reference code of the record to be retrieved is outside the total range of pages of the file.
D01	Attempt to store an unallowed duplicate record.
S01	Attempt to store a record when there is no space available within the range of pages specified for the record.

The second type of error involves those situations which are the result of improper use of the IDS functions, invalid definition of the data file, or hardware malfunctions which cannot be recovered by the software. These conditions, which are listed on the following page, result in an abort of the program with the following message printed on the execution report:

*** ABORTED BY IDS QUIT ROUTINE REASON CODE XX ***

In addition to the above message, a memory dump of the program is produced. Any time an abort occurs for any reason within an IDS program, the IDS file is first CLOSED with the appropriate IDS pages restored to the mass storage device.

Reason Code	Description
01	Attempt to store or retrieve a record with the IDS file not opened.
02	Capacity of the disc file allocated via GECOS is not large enough for the number of pages specified by the environment record.
03	Insufficient memory allocated to provide at least 2 buffers.
04	Authority-key does not match authority lock.
05	Invalid device-type allocation.
10	Definition Error-No detail definition for secondary or calculated record.
11	IDS subroutine table overflow. QASC, QDLTE and QSTOR must be reassembled.
12	Definition Error-Unique field for primary record has not been defined.
13	Definition Error-No record definition has been established for this record type.
14	Usage Error-Attempt to head chain for which no current record exists.
15	Attempt to Modify, Delete or Store a record with processing mode not equal to "Update."
16	Field to be modified or moved not defined for the current record.
17	Attempt to delete a record not previously retrieved.
18	Attempt to retrieve next in a chain for which no current record exists.
19	A control definition has been encountered which has an invalid control code.
20	A sort control or a randomize control field is specified with a record increment of zero.
21	Record definition of a record to be stored indicates that the record is less than 6 characters long.
22	Record definition of a record to be stored indicates that the record size is greater than page size 22.
23	Attempt to store a page header record.
24	Attempt to store a primary record with no unique field defined.

Reason Code	Description
25	Attempt to store a record which is a detail in one or more chains when no storage chain is specified.
26	Record type of record retrieved is not defined for the chain specified.
27	Invalid conditional action specified for a conditional delete statement.
52-55	Linking of a record cannot be completed. Either the record cannot be retrieved from the disc or, for some other reason, the chain-pointer has been lost or is invalid.
56	Page read from mass storage device is not the page requested.
57	Attempt to delink a record when the chain table "NEXT" is zero. (Subroutine error)
58	Attempt to link a record into a chain with the chain table "NEXT" equal to zero. (Subroutine error)

LITHO U.S.A.