NASA TECHNICAL MEMORANDUM

NASA TM X-52781

NASA TM X-52781

AN INFORMATION RETRIEVAL SYSTEM

by Charles M. Goldstein and John A. Lozan
Lewis Research Center
Cleveland, Ohio

TECHNICAL PAPER presented at
SHARE XXXIV
Denver, Colorado, March 2-6, 1970

AN INFORMATION RETRIEVAL SYSTEM

by Charles M. Goldstein and John A. Lozan

Lewis Research Center
Cleveland, Ohio

TECHNICAL PAPER presented at

SHARE XXXIV

Denver, Colorado, March 2-6, 1970

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

share XXXIV

THE NASA LEWIS INFORMATION STORAGE AND RETRIEVAL SYSTEM FOR THE IBM 360/67 - AN OVERVIEW

Introduction

The Lewis on-line information storage and retrieval system (LISR) to be described here owes its inception to the needs defined by the NASA Aerospace Safety Research and Data Institute (ASRDI) for the creation of a safety data bank. The requirement posed by ASRDI dictated the need for a generalized system capable of defining and including many different files. As a consequence, the resulting design was formulated not to serve ASRDI exclusively, but rather to provide the means to service ASRDI needs as one application among many other possible applications.

The following presentation will be given in two parts. Part 1 will provide the user overview while Part 2 will present a system overview. Part 1 will be further subdivided into initial implementation and program extension. Part 2 will be likewise subdivided into initial implementation and system extension. The outline of topics to be discussed is shown in figures 1a - 1d.

Part 1 - User Overview

A. Initial Implementation

1. Introduction

As mentioned in the Introduction, this system has been designed to provide the user with on-line information storage and retrieval capability. This includes on-line file creation, on-line search and retrieval, as well as on-line input of new files and correction to old files. It does not include on-line file maintenance, itself. The overhead of real-time file updates was not considered justified for the expected applications. The need was felt, however, for on-line interaction at all levels. Hence, this system was designed to take maximum advantage of the IBM 360/67 TSS facility.

A schematic of the data base for this system is shown in figure 2. The primary unit of this data base is called a dataplex. The information to be stored will be contained in a fileplex, as shown. A fileplex will have at least one "anchor" file and, possibly, one or more "associated" files. The associated file concept is in recognition of the different storage hierarchy requirements and the different types of information in a given file. In the present case, one fileplex will be used to store reference retrieval information and, possibly, full text. Here the anchor file has been defined as the bibliographic citation. The bibliographic citation contains the essential information about a reference, such as accession number, author, title, subject terms, classification, etc. This anchor file might well be stored on a disk. Since the abstracts are expected to be accessed much less frequently than the citations, the abstracts may be stored in an associated file in, for example, a data cell. If full text were desired on-line, then the full text might be included in still another associated file in some other mass storage medium (very likely, read-only).

In order to facilitate searching, it is desirable to create indices on such items as author's name, contract number, subject terms or key words, etc. All inverted file or indices relating to the fileplex are stored in an xplex as shown in the figure. Here, the major indices related to reference retrieval systems are indicated. At the far right of the figure we see a single file which is an adjunct to the dataplex, but yet not an associated file. In this case, this file contains the subject terms thesaurus corresponding to index number 1. This thesaurus, by definition, provides the hierarchy of the subject terms.

An overview of the file structure is shown in figure 3. A file contains numerous records. Each record may contain one or more fields; for example name, field, social security number, field, keyword field. The fields defined in this system may be either fixed or variable. They may contain repeating elements which are either fixed or variable.

2. File Creation

One of the foremost requirements of this system was the ability to easily define new files. In particular, the user should be able to sit at a terminal and, by suitable prompting, interactively define a file in a very short time. Although the present system has not yet been implemented, figures 4a through 4d simulate the creation of a new file by a user seated before an alphanumeric graphics terminal. In figure 4a, the user has responded to the usual system underscore with the command CREAT. In the next figure, the system has responded with all prompting messages needed for the definition of the first field in a new file. For the simplicity of presentation, we are assuming here that we have an alphanumeric display which allows us to present a fixed format with both horizontal and vertical tabbing of the variable input data. That is, after entering the file name and hitting return, the cursor would immediately come down to the right of the colon opposite field name. The first three prompting messages are self-explanatory. The fourth provides a means of setting bit switches for such items as male or female, married or unmarried, etc. Item 12 calls for the name of the input verification routine. This routine, if provided, would be used for testing the types of input characters (numeric, alphabetic, blank, etc.). The input validation routine (item 13), if provided, would be used for special validation and conversion of input data (e.g., the validation of a number entered which contains a check digit). Item 16, the security code, provides for field by field security protection where desired. Upon defaulting message number 17, messages 2 through 17, would reappear, prompting the user for the next field information.

There will be times when the user will want to update the field of an existing file. Certain modifications can be performed using the UPDAT command, whereupon the messages shown in figure 4c are presented to him on the display. Note that the user is not allowed to delete fields or elements or in any way change the field themselves using this command. Basic changes to the fields other than those allowed here would necessitate reformatting the entire file. Procedures for doing this are contained in the Reformatter utility to be discussed later.

3. Retrieval

In 1969, the Scientific and Technological Information Division (STID), NASA Headquarters, implemented a large scale on-line reference retrieval system. This on-line system is known by the acronym RECON for Remote Console Information System. Since users of the NASA Lewis system will also be users of the NASA RECON system, we have tried to be as compatible as possible from a user's viewpoint. RECON terminals are presently available, or will soon be available, at all NASA Centers. Since RECON is well documented (Ref. 1); only a summary of the retrieval commands will be given here. A list of the commands is given in figure 5. All the commands listed, with the exception of SAVE and CORRECT, are also to be found in RECON.

The BEGIN command is similar to that in RECON. The EXPAND command includes an additional parameter, the indexname. This is necessary if one is employing more than one index. This parameter will be defaulted for the most commonly-used index.

SELECT combines the two RECON commands, SELECT and COMBINE. With a single element as a parameter, SELECT is the same as presently implemented under RECON; utilizing one of the three Boolean operators (and/or negation), this command then becomes the same as the RECON COMBINE. An extension to the RECON command is provided by the comma when used as a delimiter between elements. This provides for the selection of a list of unique elements using one SELECT command.

Although DISPLAY and PRINT are identical to their RECON counterparts, the number of allowable formats will be expanded to include the display of abstracts and/or other associated files. KEEP, PAGE, STRATEGY and END commands remain the same as in RECON.

The SAVE command introduced here is to be completely subordinate to DISPLAY. Invoking the SAVE command will store anything presently displayed into set number 98 for future reference or output.

The CORRECT command allows the user to immediately include any corrections or additions to the record he is currently displaying. This leads us into the topic of the next session "On-line Editing".

4. On-Line Editing

Let us regress for a moment and review the philosophy of file access as incorporated in the present system. It is assumed that, in general, each file will have one owner and many users. The owner will permit the users read-only access to his file through the LISR system executive and the TSS data set sharing facility. The program executive will, furthermore, honor the field security as imposed by the owner. That is, the owner may restrict certain classes of users from certain fields of information while granting them access to other fields. Without digressing further at this point, let us note that the problem of selectively securing parts of a data set from a TSS user with read-only access should be resolved under MTT.

While read-only access will preserve file integrity and prevent users from changing or deleting elements of the original file, it is nevertheless highly desirable that the user should have the facility for immediately noting corrections and/or additions to the records he is perusing. The CORRECT retrieval command has been incorporated for this purpose. Figure 6 schematically depicts the method that has been implemented. Upon displaying a record, the user keys in a correction. These corrections are then stored in a user data set as transactions until the user has finished that particular search. At the end of his search, all the CORRECT transactions stored in the ~~user library are automatically transferred to a transaction~~ queue belonging to the owner. The owner's transaction queue accepts such transactions from all users of his file. The owner then periodically obtains a listing of the transactions in this queue or displays them at a terminal in order to decide which transactions he wishes to incorporate in the file. Incorporation of these transactions will be solely at the discretion of the owner. In this way, we hope to obtain complete file integrity as well as the advantage of on-line additions and corrections to any records in the file.

5. File Maintenance

File maintenance includes additions of new records, the correction of records already present, the validation of field entries, the updating of all pertinent indices and the final storage assignment of all information. As indicated in figure 7, input to the maintenance routine originates from three sources. The primary source is input associated with new records or data. New records will pass through a syntax checker before being stored in an intermediate transaction queue. Input to the syntax checker may be in the form of prestored data sets or direct terminal input. The syntax checker module allows for a review of all transactions in the intermediate transaction queue as indicated in the figure. The input transactions may, at this time, be further modified or deleted before being entered into the file.

Two other sources of input are generated by the on-line text editing capability via the retrieval CORRECT command and the transaction generator. The former has already been discussed in section 4, while the latter will be discussed under Utilities in a later section. After transactions from all three sources have been merged, they enter a final transaction queue. Maintenance is then performed on this transaction queue as shown in the figure.

It must be emphasized that maintenance is to be performed in a batch mode. The frequency of maintenance will be totally under the discretion of the file owner.

6. Terminal Support

The initial alphanumeric display device to be supported is the Computer Communications Inc. (CCI) CC-30 display terminal. This terminal can display twenty-four lines of 40 characters each. The terminal support will initially be imbedded in the LISR. It will not allow a user to log on under TSS. From a user's point of view, the display will look like a RECON terminal.

7. Utilities

The following utility programs will be part of the initial implementation:

a. Backup

This utility allows the user to write onto tape any part or all of a given dataplex. The frequency with which backup will be utilized is also under the discretion of the owner.

b. Restore

In the event of the loss of information from any direct access device, restore allows the restoration of that information from the latest backup tape.

c. Reorganize

In the event of many modifications or deletions to a given file, the efficiencies of access to that file may be seriously degraded. Reorganize is used with Restore and Backup to effect a file clean up.

d. Purge

Purge is utilized to delete all or some parts of a dataplex.

e. Transaction Generator

One of the major design criteria of this system was to provide for the inclusion of files of information from other information centers. This capability is provided by the Transaction Generator. In order to incorporate the file from another center, the user would, in practice, first call CREAT to define the characteristics of that file in the NASA Lewis system. He would then access the Transaction Generator and describe to it, interactively, the characteristics of the file to be input. He would also identify to the Transaction Generator the name of the newly-created file. The Transaction Generator would then operate on the input file to create the corresponding transactions for the file maintenance program discussed in the previous section.

f. Reformat

The deletion or modification of any field of a record will in general affect the whole structure of the file and the corresponding descriptor tables. In order to protect the integrity of the file, but at the same time, to allow for such changes, the Reformat utility has been provided. This program defines an entirely new file, incorporating the desired changes or deletions to fields in the old file. The old file is then used as input to the Transaction Generator utility which, in turn, provides the transactions to create the new file.

B. Program Extensions

The following extensions are considered an integral part of information storage and retrieval system. They have not been included in the initial implementation due to reasons of priority scheduling only.

1. Text Editing

   Although the CORRECT command in the retrieval subsystem provides a means of editing records, this method of text editing is only considered an expediency. What is truly desired is a means of editing at a graphic display utilizing function keys and/or a light-pen to effect insertions, deletions, paragraph changes, etc. The corrections, as they appear on the display, should then be massaged by the system to provide the proper transactions for file maintenance.

2. Light-Pen Support

   Since the general class of user is not expected to be an expert typist, it is highly desirable to provide him with a means of conducting on-line searches as easily as possible. Implementation of the light-pen with light keys would greatly assist in this purpose. Selection of terms from the authority word list, combinations, displays, prints, etc. can all be controlled directly utilizing light pen and light key.

3. Natural Language Retrieval Strategies

   It is intended to give each user of the retrieval system access to many different files. Since all reference retrieval files have the same basic logical structure, initial implementation will simply offer the user a list of files from which he will choose the file of interest. It would be more desirable, however, to have the user present his request in easily-stated natural language and let the system search pertinent files to return the desired information to the user. Where the user's request is ambiguous, the system should be capable of prompting the user for more definitive information regarding his search.

4. Extended Character Set Input

   Standard computer input is restricted to character sets not greater than 96 characters (ASCII). The inclusion of technical information in abstracts, for example, requires a much extended character set which includes mathematical symbols, Greek letters, subscripts, superscripts, etc. The AEC, for example, is presently utilizing approximately 1250 unique characters in the publication of abstracts for the Nuclear Science Abstracts (NSA). The AEC-DTIE (Division of Technical Information Extension) is currently implementing an on-line system for purposes of on-line input of NSA abstracts. This system, or one similar, will be required by the ASRDI safety data bank.

5. Terminal Support

Subsequent to initial implementation, additional alphanumeric terminals will be supported as the need arises. In addition, other terminals such as storage tube displays will be implemented to give us the requirements for extended character set input and output as well as the storage and retrieval of graphic information.

6. Publications Formatting with Composition

As is true with most other information centers, ASRDI will be concerned with recurring and demand publications of information. Methods of formatting textual material with dynamic figure composition for various photocomposition devices will be developed. Rapid dissemination of important safety information will be a prime criteria.

7. Input-Output Tape Utility

Output tapes generated by the IBM 360/67 computer under TSS are not compatible with other 360 OS systems nor with other manufacturers computer hardware. As a result, an additional tape utility is being programmed in order to provide tapes in various formats for transmission to other centers with different types of computers. This utility will provide for various formats and character translation. This utility shall further be extended in order to read tapes of any format obtained from other installations.

8. New Storage Device Support

The inclusion of large amounts of archival records in an on-line information system makes the exclusive use of disk storage uneconomical. For this reason, other on-line storage devices will be supported. The first device to be supported will be the Data Cell. Future considerations will include varieties of read-only storage.

9. Report Generator

In order to provide the nonprogrammer-user of the system means to obtain reasonable output of the information he desires, some form of report generator will be required. The report generator will no doubt utilize or be a part of the Publications Formatting Subsystem.

Part 2 - System Overview

A. Initial Implementation

1. Design Criteria

The design of this system was to be general, in nature, but limited primarily by a number of real or imposed constraints. The primary constraint was that of time; it was desired that the system be at least partially operational by mid-1970. This fact, coupled with the presence of a large scale interactive computer system at NASA Lewis, determined that the retrieval system should be designed around the facilities provided by the IBM 360/67 and the Time Sharing System (TSS). One benefit derived from this decision was the use of the extensive facilities for data file handling and interactive computing available under TSS.

The choice of a programming language in which to write the system was also affected by the time constraint. It was felt that an assembler language based system could not be implemented within the time available without significantly increasing the size and cost of the project. Of the high level languages available to TSS, only PL/I seemed practical for this type of system. In addition to making the coding less complex and significantly reducing the number of instructions that must be written, it was felt that the compile time facilities available in PL/I would be extremely valuable during the entire implementation phase of the project.

As mentioned, the primary motivation behind the development of this system was the requirement of the Safety Institute. However, even from the earliest times, it was required that the design of the system be such that it could either be directly used in other areas, or at least be modified for such purposes.

2. File Support

The first problem to be approached was the design of a file structure to hold the information required and a means of accessing the data on file. Our solution to this problem was the development of the "dataplex". A dataplex was designed to contain all of the information in a particular data base, as well as the control and descriptive information required to maintain and access it. The primary file for data content was designated as the "anchor" file. Because it might be desirable to physically separate the data into a number of sub-sets, the "associated" file was developed. Since ready access to the data was required for a number of different criteria, the ability to create indices on various fields of either the anchor or associated files was provided. To allow for a generalized method of accessing the various files of the various dataplexes, it was decided that a data set descriptor should be defined for each file. These descriptors would describe each of the fields of a data set in such a way that field-level access could be provided for all the data sets in the system.

In Figure 8 we show a sample dataplex. This dataplex consists of the anchor file, two associated files, and two inverted indices. The descriptors for these files are all contained in a regionalized ISAM file.

In Figure 9, we see the actual data that is recorded in each of these files. The data, in reality, consists of six distinct fields. The most important field (KEY) would be the field that is used to identify the individual records in the file. Of the remaining five fields, two are to have inverted indices made of their contents and three are to be placed on associated files. As shown, the anchor file would consist of three fields, the key field and two data fields. One of these fields (FLD2) has an inverted index (INDEX1). The first associated file (ASSOC1) also has three fields, one of which (FLD5) has an inverted index (INDEX2). The second associated file has two fields, the key field and one data field. Note that the field names are consistently maintained when one field is present in more than one file. The second part of this figure shows the significant fields of the descriptor records for each of the files shown. Note here that each descriptor has a header record that contains information concerning the file as a whole. For our purposes, the only significant field is the length of the fixed portion of each record in the file (25 for the anchor file).

As shown, the anchor file descriptor has seven entries, a header record and six field descriptors. For the fields actually a part of the anchor file, each entry is complete.

However, for the fields contained in the associated files, the only data entered is a switch which indicates the presence of the associated file and the name of that file. The associated file descriptors are similar to the anchor, except that they contain only entries for the fields present in the file and that they cannot have other files associated to them. Note that the existence of inverted indices is handled in much the same way as associated files; a switch indicates the presence of the index and a separate field contains its name. Inverted index descriptors are all similar. Each has three entries, a header record, the key field for the index (i.e., the data field being indexed) and the list of keys that contain that same data item.

In figure 10, we have shown the various fields that comprise a descriptor header record and a descriptor field record. The header record specifies the type of file and its current status, indicates the length of the fixed portion of the records in the file and indicates the number of index and associated files present. The field descriptor specifies the type of field and its recording mode, its location and length, the names of any validation or formatting routines to be used for the field, the presence of and names of any index or associated file, and the security restrictions on the field.

These two record types are very similar, by design. In fact, every record in the system has, basically, the same format, appropriately named the Universal Record Format. This format is a derivation of the normal S/360 variable record format. It has a system-maintained length field, a base length portion and a variable portion. The base length portion contains all fixed length fields, of which the first is designated as the key of the record. The variable length portion contains all of the variable length fields in the record. These fields can be of three types, a simple variable-length field, a field composed of a variable number of fixed-length elements, or a field composed of a variable number of variable-length elements.

The question is probably arising - why have we designed such an elaborate structure? The answer is so that we might write one generalized executive routine to handle the interface between all programs and the various types of files in the system. This executive we have called DBPAC, for Data Base access PACkage. As shown in figure 12, DBPAC provides a PL/I interface between the users and the TSS data files. It does this by giving the user facilities for

1) file accessibility

2) record retrieval and posting

3) field retrieval and posting

4) list processing.

DBPAC is essentially an interpretive routine which uses the descriptors to satisfy user requests for adding, deleting, modifying, or extracting information from the files. It should be noted that DBPAC is, for the most part, written in PL/I.

Because this is the first implementation of both DBPAC and the system as a whole, it was felt that the internal structure of DBPAC might change, but that it would be desirable if those changes could be made transparent to any of the programs in the system. For this reason, we made use of the preprocessor feature of PL/I to develop a file support preprocessor, called DBPL/I. This preprocessor also enabled us to make the use of DBPAC facilities similar in nature to the use of normal PL/I facilities. A list of the DBPL/I facilities is provided and their syntax requirements is given in figures 13 and 14.

3. Terminal Support

It was obviously desirable to have a system, such as this, make use of a CRT type terminal device as the primary interface between the user and the system. Unfortunately, TSS did not, and does not, support such a device. Therefore, we have also had to develop the support for a CRT terminal. In figure 15, we show both the basic structure of the normal TSS terminal support and the basic structure of our terminal support. Basically, the GATE routines provide a logical IOCS level of terminal support for TSS, as does our data base remote terminal support (DBRTS). The TAM routines provide physical level IOCS support for terminals, as does our data base IOREQ support (DBIORQ). IOREQ is a TSS facility comparable to EXCP under OS which allows the programmer to construct his own channel command word lists for the direct execution of input/output functions.

The terminal support routines provide the user with the ability to access a terminal, primarily a CRT device, on a line or screen basis. In addition, they provide some basic buffer maintenance routines which allow the CRT user to control the amount and content of information presented at any point in time.

Just as with DBPAC, in DBRTS we want to protect the user programs from changes in the internals of the terminal support, as well as simplify the use of the support. For this reason, we have developed a terminal support pre-processor (TSPL/I). Here again, the programmer is presented with facilities that are similar in use and concept to those normally provided in PL/I. Figure 17 contains a list of the commands available to the TSPL/I user and their format.

4. Overall Structure

The modules which comprise the remainder of the Lewis Information Storage and Retrieval System are divided into five functional groups,

1) Descriptor Editor

2) Transaction Generator

3) Maintenance

4) Utilities

5) Retrieval

The Descriptor Editor is a conversational routine, available to the "owner" of a dataplex. It uses TSPL/I in a prompting mode, together with the TSS system message file, to interact with the user to create or update descriptors. It automatically constructs a header record for each descriptor and prompts the user for the characteristics of each field in the fileplex. It will automatically separate the indicated fields into associated files and construct their descriptors. It will also automatically construct the index file descriptors, where indicated. The program makes provisions for reasonable default values, whenever possible, and bypasses prompting for characteristics not applicable to the field being processed. The program can be run non-conversationally, even though it was primarily designed for interaction with the user. Processing of the descriptor can be suspended at the end of any given field entry and can later be continued from that point. Once a descriptor has been successfully created and data loaded to the file, a second entry point to the Descriptor Editor can be used to update those fields of the descriptor which do not affect the physical characteristics of the existing file. Except for its initial linkage from command language, the program is written entirely in PL/I, making extensive use of the facilities of both DBPL/I and TSPL/I.

The Transaction Generator provides a means of introducing data files of information from other systems into the Lewis system. This sub-system is divided into two basic phases. The first phase uses conversational prompting to build a control table identifying the characteristics of the data to be processed. This function is analogous to the creation of descriptors by the Descriptor Editor, except that these control tables are transient and not really a part of the file structure of the system. The second phase of the Transaction Generator uses these control tables to extract the information from the external file (normally a logical record tape dump), manipulate it as indicated, and produce valid card-image input transactions that can then be loaded using the normal file loading facilities of the maintenance routines. If the conversion of the data cannot be completely handled by the facilities available and the parameters indicated in the control table, a number of "user exits" are provided to complete this function. These routines are also written almost completely in PL/I. Because these routines do not directly interface with any of the data base files, they do not make use of the facilities of DBPL/I or DBPAC.

The maintenance sub-system is divided into three basic sections; syntax checking, queue dumping, and maintenance, itself. Syntax checking enables someone to enter maintenance transactions into the system either conversationally or non-conversationally.

The owner of a dataplex controls which USERIDS can perform this function for each of his dataplexes. Once entered, transactions can be examined for correctness and completeness by the person entering them at any time before they are applied to the data base itself. The queue dumper is a function restricted to the owner of a dataplex, whose purpose is to combine all of the maintenance transactions pertaining to a particular dataplex into one file restricted to his own use. This routine can also read the input from the Transaction Generator and the CORRECTION features of the retrieval sub-system. The output of this program is a single file of transactions ready for application to the dataplex. The maintenance function, itself, then reads this transaction file, analyes the action specified and, if the data passes validation, performs the indicated function. It should be noted that the final responsibility for the validity of the transactions lies with the owner. The system can ensure the fact that the transactions are syntactically correct and that the data to be entered is in a valid format for the field to which it applies, but although some validation of the content of the data is performed, there is a limit to the inherent validating capabilities of the system. Except for some system dependent input/output functions, these programs are, again, written in PL/I utilizing both DBPL/I and TSPL/I. As with the Descriptor Editor, the user need only be concerned with the anchor file and its fields, all other posting is performed by DBPAC automatically.

The utilities are a set of routines which provide the owner of a dataplex with functions useful in maintaining large data files. These functions are,

1) file backup

2) file restoration

3) file reorganization

4) file reformatting

5) file purging.

File backup allows the owner of a dataplex to get a logical record dump, on tape, of any or all files in a given dataplex. The program builds a list of the files to be dumped and then DDEF's a tape and write the file to tape on a record basis. This is repeated for each file in the list.

File restoration allows the owner of a dataplex to restore a previously dumped file, or files, to the dataplex. It also operates on a logical record basis and simply reverses the backup function. It builds a list of the files to be restored, DDEF's the appropriate tape, verifies the applicability of the tape, and writes the tape to the file on a record basis. This is repeated for each file in the list.

With the original design of TSS ISAM support, it was possible for a file to become so disorganized that access to the file was seriously affected. Because of this, a file reorganization function was included which would perform a backup immediately followed by a restore. Because of changes made to the access method during the interim, it is not clear that this is still true.

File reformatting allows the owner of a dataplex to add or delete fields from an existing dataplex. It first updates the appropriate descriptors and then reads, updates, and rewrites each record on file in the new format.

File purging provides the owner of a dataplex with a convenient facility for erasing any or all files in a dataplex and their corresponding descriptors. An option on the program is to produce a historical backup tape.

Each of the utilities uses DBPL/I to access both the dataplex files and their descriptors and TSPL/I to prompt for input parameters and to write diagnostic messages to the user. They are all primarily written in PL/I.

The retrieval sub-system provides the user with a list of commands that he can use to access the various indices, compiles sets of record keys which can then be displayed in a number of different formats. Retrieval has a director which controls the execution and sequencing of the various commands. It makes use of the list processing functions of

DBPL/I to retrieve and manipulate the lists of record keys indicated by the user during retrieval. Retrieval is primarily a conversational program and uses TSPL/I to control all communication between the system and the user. Except for certain system dependent functions, the retrieval modules are written in PL/I.

Figure 18 is a graphic representation of the components of the Lewis system and their relation to each other and TSS.

B. System Extentions

1. Tuning

Because of the extent to which the system is being written in PL/I, it is felt that, once the system has been in operation, analysis will show areas of coding that could be redone in assembler language to improve performance.

2. New Facilities

At this point in time, there are no facilities for a straight linear search of the files and the accumulation of statistics or averages. This will probably be desired when other applications are added and should not prove to be a difficult implementation.

In conjunction with linear searching and also with the current search capabilities, it would be highly desirable to have a generalized report generator available. This might involve a modification to the descriptors, as well as the implementation of some special processing routines. If this is the case, the Descriptor Editor would have to be modified, as would those sections of DBPAC that handle descriptor processing, but none of the other routines should be affected.

KWIC indexing is an existing method of information presentation that is useful in many situations. With a small amount of design, it should be possible to develop a set of routines that will use the inverted indices and the data on file to produce KWIC formatted output from the existing data files.

3. Multi-terminal Task (MT/T)

One of the new features being added to TSS is the ability of one task to interface between a number of terminals. By using this feature to handle the situation where multiple users are using the same routines on different files, it should be possible to reduce the overhead and improve performance.

4. Terminal Input/Output

Our present terminal support is simply a special purpose user program. If an equivalent facility for CRT type terminals could be developed as standard system support, it should improve performance.

5. RTAM

As shown before, both TSS terminal support and our own is divided into a physical and logical level. Current plans call for TSS to remove their physical level of support from virtual memory, where it is paged and place it in real core, to improve efficiency. If this is the case, and support for a CRT type terminal is provided, it should also benefit performance.

NASA-LEWIS INFORMATION STORAGE AND RETRIEVAL SYSTEM FOR IBM 360/67 TSS

PRESENTATION OUTLINE

PART I. - USER OVERVIEW

- A. INITIAL IMPLEMENTATION
- B. PROGRAM EXTENSIONS

PART II. - SYSTEM OVERVIEW

- A. INITIAL IMPLEMENTATION
- B. SYSTEM EXTENSIONS

CS-53219

Figure 1 (a)

PART I - USER OVERVIEW

A. INITIAL IMPLEMENTATION

1. INTRODUCTION
   - A. DATA BASE
   - B. RECORD FORMAT
2. FILE CREATION
3. RETRIEVAL
4. ON-LINE EDITING
5. FILE MAINTENANCE
6. TERMINAL SUPPORT
7. UTILITIES
   - A. BACKUP
   - B. RESTORE
   - C. REORGANIZE
   - D. PURGE
   - E. TRANSACTION GENERATOR
   - F. REFORMAT

CS-53220

Figure 1 (b)

PART I - USER OVERVIEW

B. PROGRAM EXTENSIONS

1. TEXT EDITING
2. LIGHT-PEN SUPPORT
3. NATURAL LANGUAGE RETRIEVAL STRATEGIES
4. EXTENDED CHARACTER SET INPUT
5. TERMINAL SUPPORT
6. PUBLICATIONS FORMATTING WITH COMPOSITION
7. I/O TAPE UTILITY
8. REPORT GENERATOR
9. NEW STORAGE DEVICE SUPPORT

CS-53218

Figure 1 (c)

DATA BASE OVERVIEW

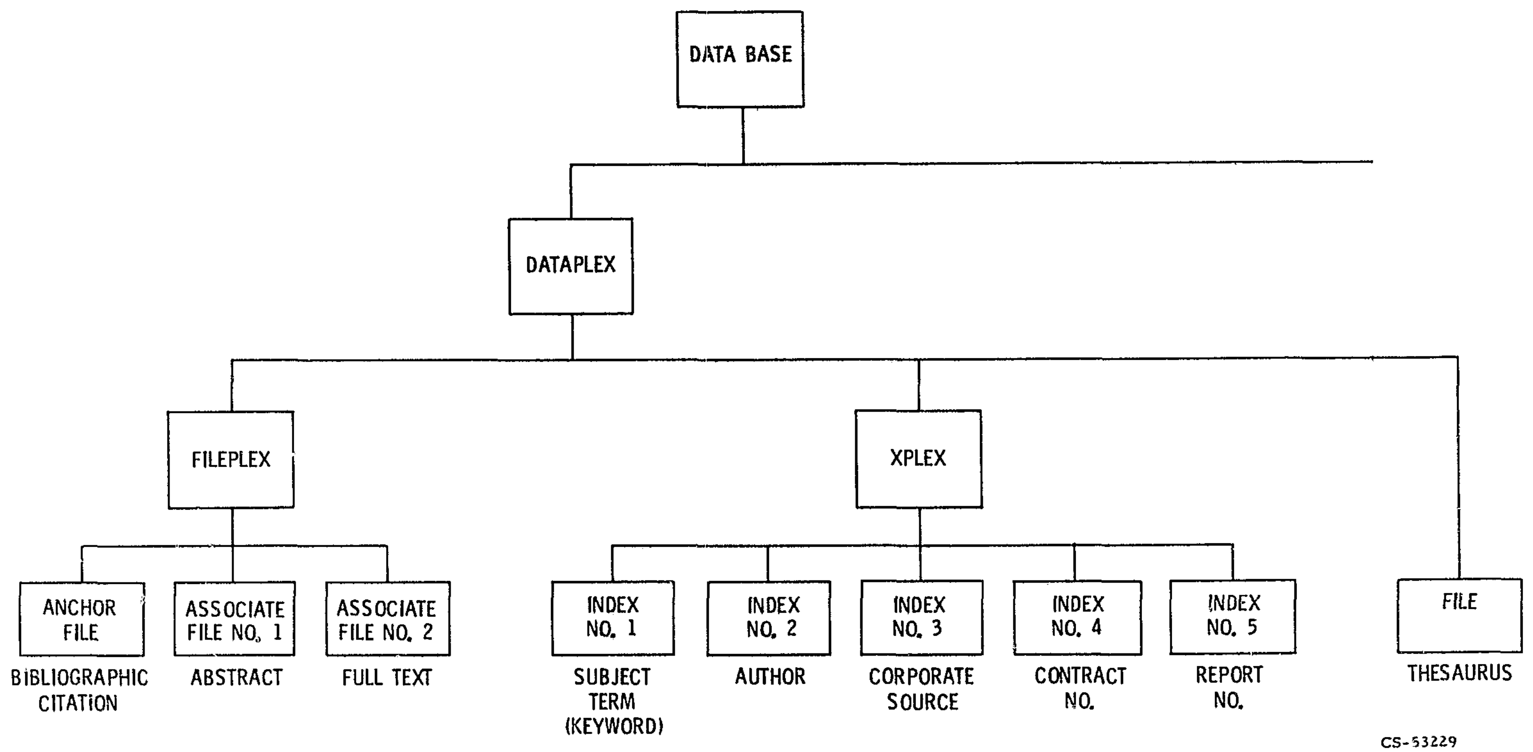


Figure 2

FILE OVERVIEW

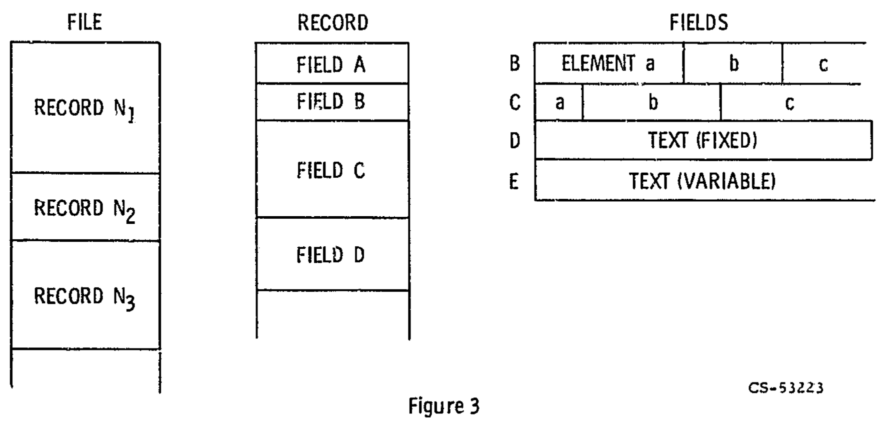


Figure 3

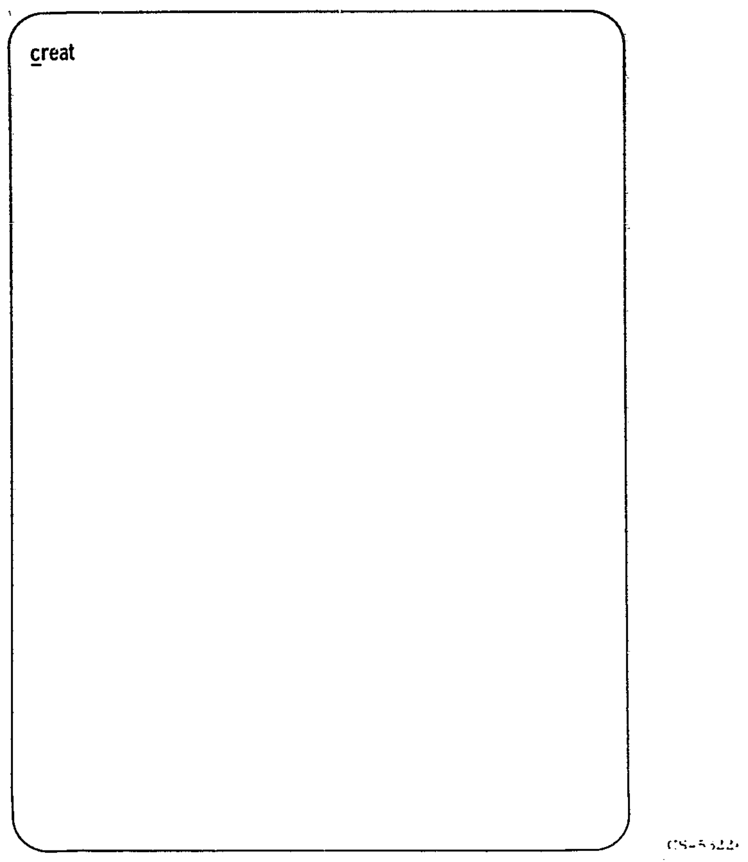
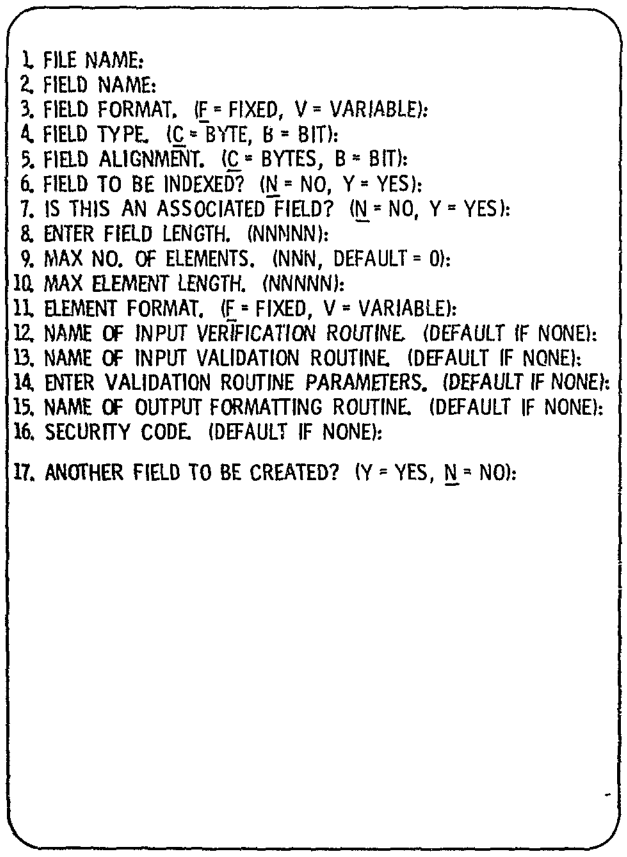


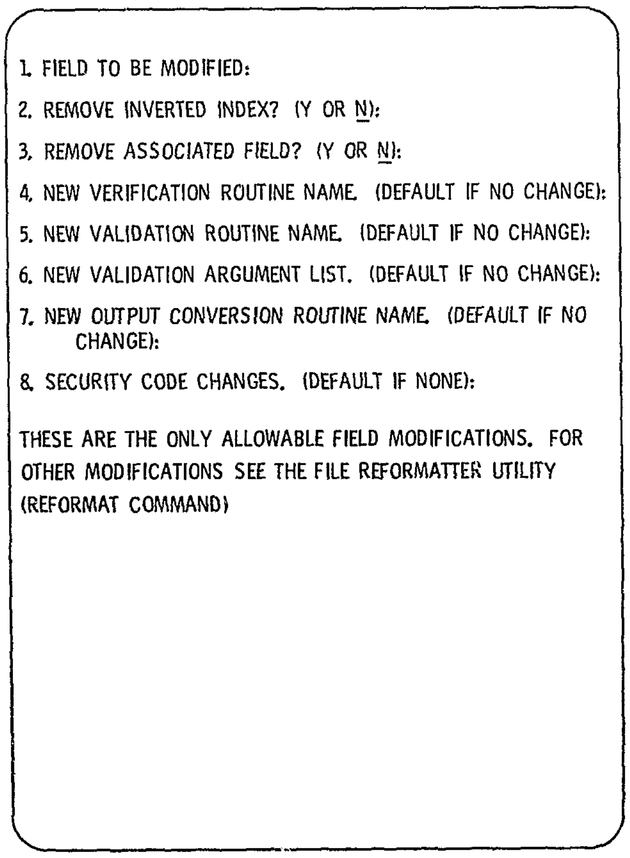
Figure 4



```
1. FILE NAME:
2. FIELD NAME:
3. FIELD FORMAT. (F = FIXED, V = VARIABLE):
4. FIELD TYPE. (C = BYTE, B = BIT):
5. FIELD ALIGNMENT. (C = BYTES, B = BIT):
6. FIELD TO BE INDEXED? (N = NO, Y = YES):
7. IS THIS AN ASSOCIATED FIELD? (N = NO, Y = YES):
8. ENTER FIELD LENGTH. (NNNNN):
9. MAX NO. OF ELEMENTS. (NNN, DEFAULT = 0):
10. MAX ELEMENT LENGTH. (NNNNN):
11. ELEMENT FORMAT. (F = FIXED, V = VARIABLE):
12. NAME OF INPUT VERIFICATION ROUTINE. (DEFAULT IF NONE):
13. NAME OF INPUT VALIDATION ROUTINE. (DEFAULT IF NONE):
14. ENTER VALIDATION ROUTINE PARAMETERS. (DEFAULT IF NONE):
15. NAME OF OUTPUT FORMATTING ROUTINE. (DEFAULT IF NONE):
16. SECURITY CODE. (DEFAULT IF NONE):

17. ANOTHER FIELD TO BE CREATED? (Y = YES, N = NO):
```

CS-53227

Figure 5



```
1. FIELD TO BE MODIFIED:

2. REMOVE INVERTED INDEX? (Y OR N):

3. REMOVE ASSOCIATED FIELD? (Y OR N):

4. NEW VERIFICATION ROUTINE NAME. (DEFAULT IF NO CHANGE):

5. NEW VALIDATION ROUTINE NAME. (DEFAULT IF NO CHANGE):

6. NEW VALIDATION ARGUMENT LIST. (DEFAULT IF NO CHANGE):

7. NEW OUTPUT CONVERSION ROUTINE NAME. (DEFAULT IF NO
      CHANGE):

8. SECURITY CODE CHANGES. (DEFAULT IF NONE):

THESE ARE THE ONLY ALLOWABLE FIELD MODIFICATIONS. FOR
OTHER MODIFICATIONS SEE THE FILE REFORMATTER UTILITY
(REFORMAT COMMAND)
```

CS-53228

Figure 6

RETRIEVAL COMMANDS

```
BEGIN
EXPAND term, indexname
SELECT {set|elem}{+|*|-|,}
LIMIT set/yr
DISPLAY set/format/item
SAVE
KEEP set/item
PAGE
PRINT set/format/item
STRATEGY
END
CORRECT
```

CS-53224

Figure 7

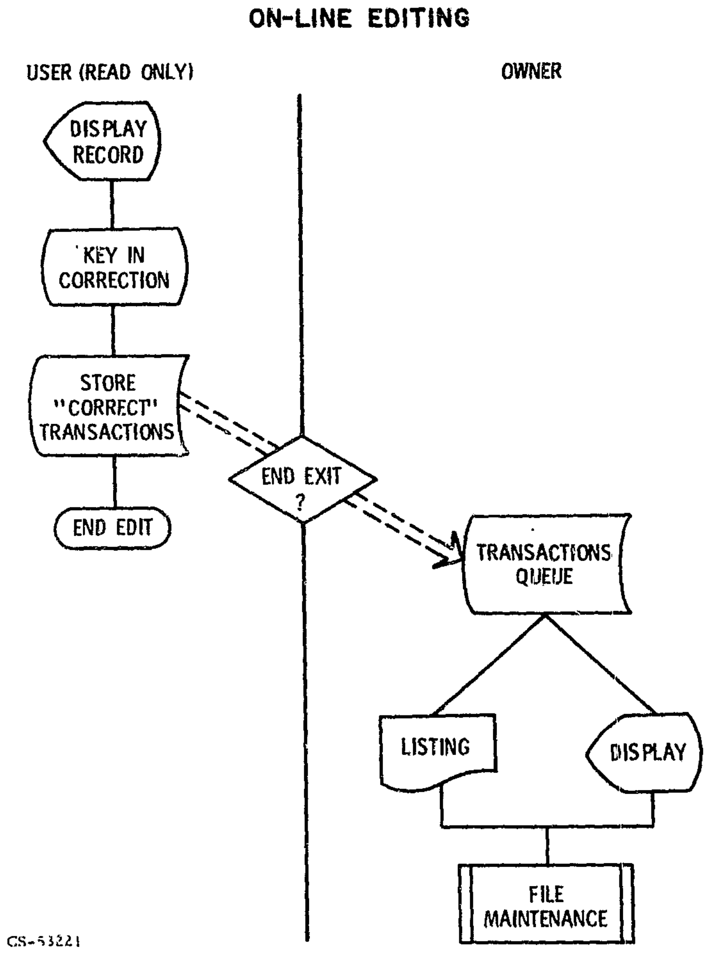


Figure 8

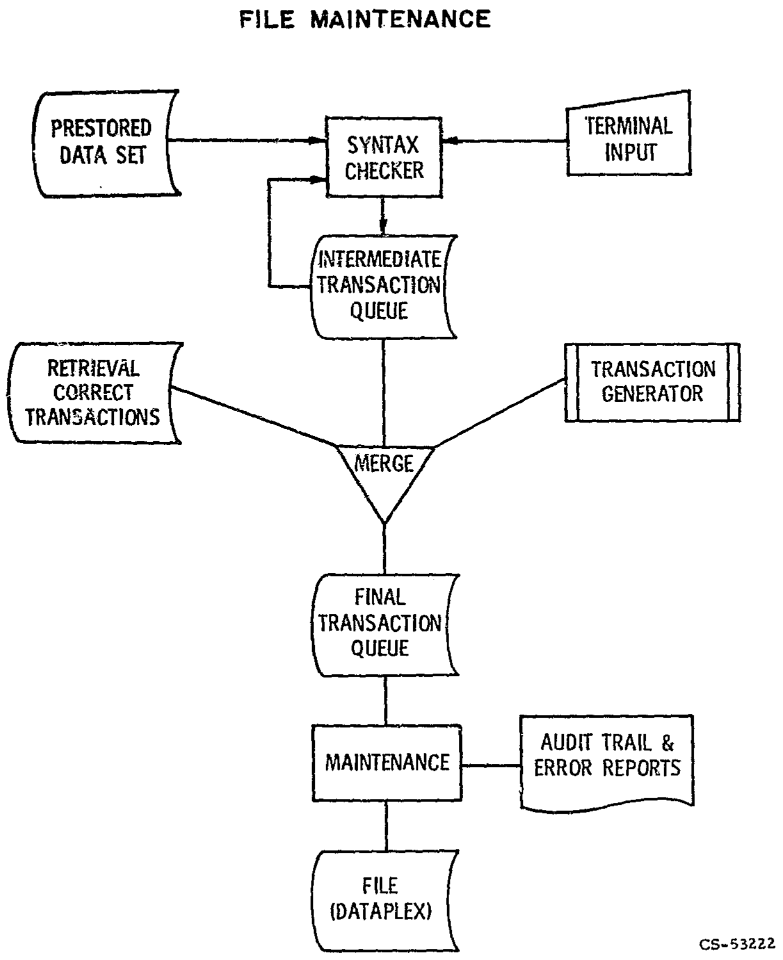


Figure 9

II. SYSTEMS ASPECTS OF LISR

- A. INITIAL IMPLEMENTATION
  - 1. DESIGN CRITERIA
    - a. TSS
    - b. PL/I
    - c. General Purpose
  - 2. FILE SUPPORT
    - a. Dataplex
      - 1) Descriptors
      - 2) Inverted Indices
      - 3) Associated Files
    - b. Universal Record Format
    - c. DBPAC
    - d. DBPL/I
  - 3. TERMINAL SUPPORT
    - a. IOREQ Support
    - b. Terminal Support
    - c. TSPL/I
  - 4. OVERALL STRUCTURE
    - a. Descriptor Editor
    - b. Transaction Generator
    - c. Maintenance
    - d. Utilities
    - e. Retrieval

CS-53200

Figure 10

II. SYSTEMS ASPECTS OF LISR

B. EXTENSIONS

1. Tuning
2. New Facilities
   a. Linear Search Capability
   b. Report Generator
   c. KWIC Output
3. MTT
4. TIO
5. RTAM

CS-53201

Figure 11

SAMPLE DATAPLEX

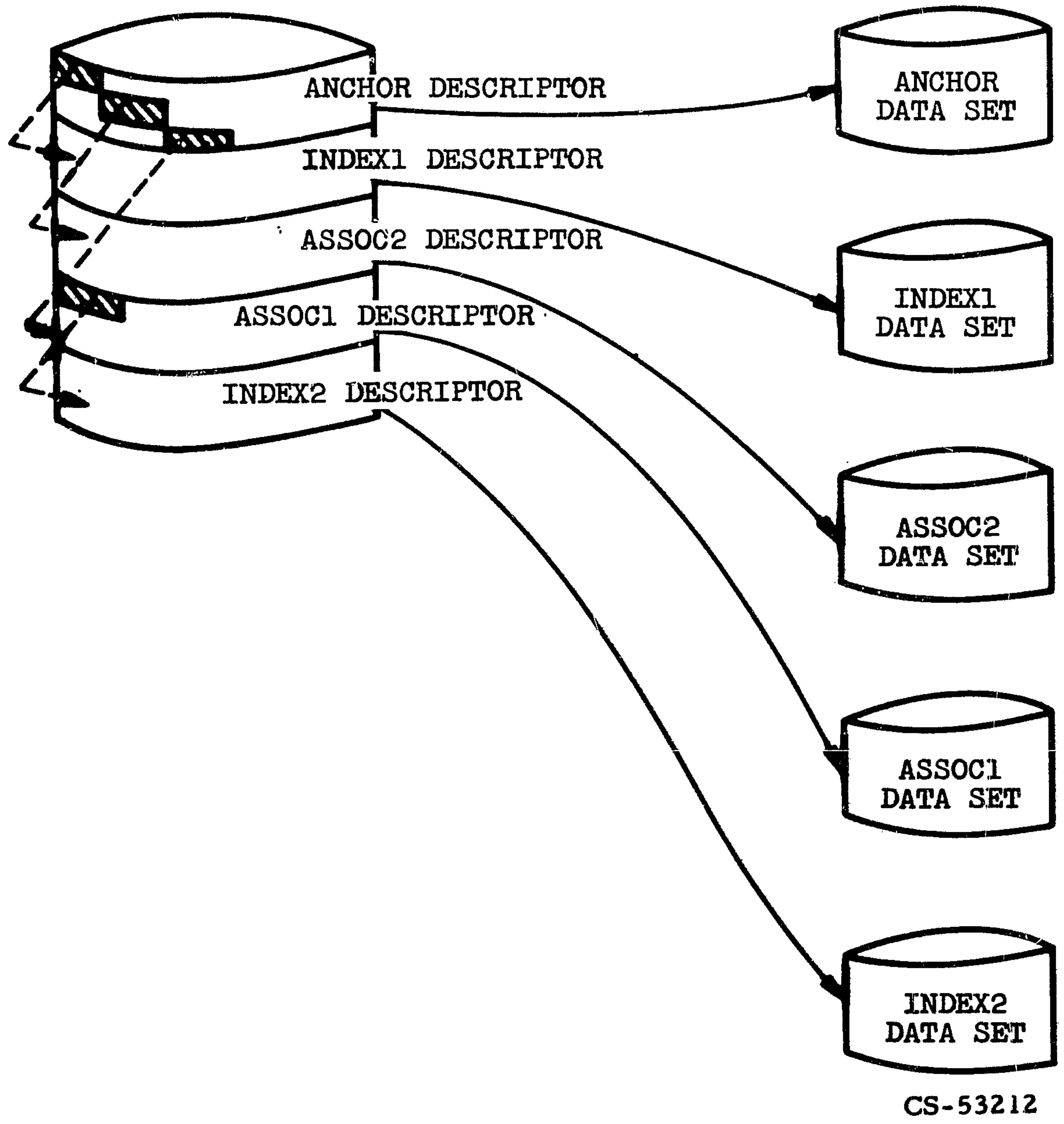


Figure 12

DATA FORMAT

KEY	FLD1	FLD2	FLD3	FLD4	FLD5
		X	A	A	A X
7	6	8	10	12	4

ANCHOR

KEY	FLD1	FLD2
K1	1A	2A
K2	1B	2B
K3	1C	2A
K4	1D	2C

ASSOC1

KEY	FLD3	FLD5
K1	3A	5C
K2	3A	5B
K3	3A	5A
K4	3A	5A
K5	3A	5A

ASSOC2

KEY	FLD4
K1	4A
K2	4B
K3	4C
K4	4D
K5	4E

INDEX1

FLD2		KEY	
2A	K1	K3	
2B	K2	K5	
2C	K4		

INDEX2

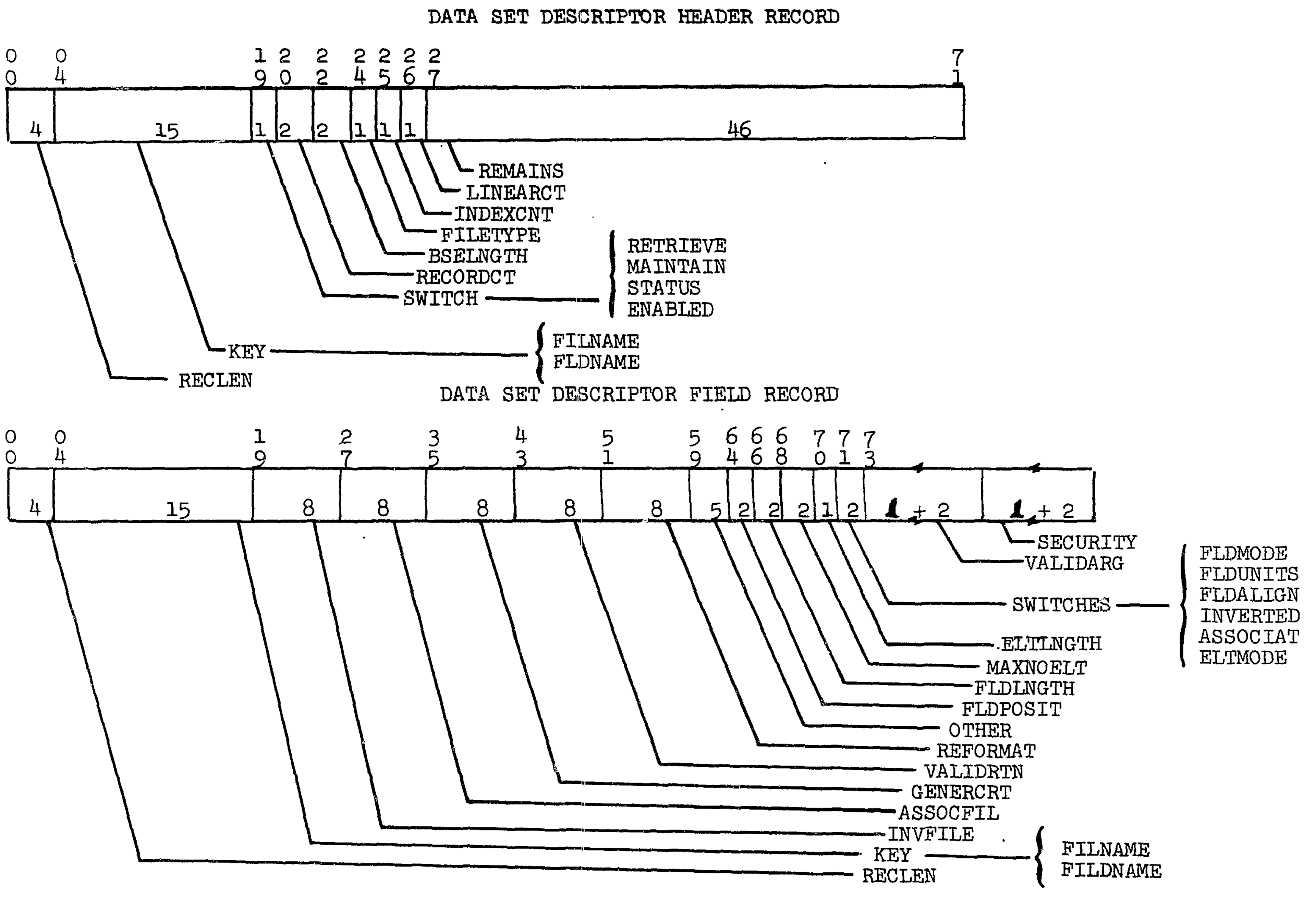
FLD5		KEY	
5A	K3	K4	K5
5B	K2		
5C	K1		

DESCRIPTORS

FILE NAME	FIELD NAME	FIELD LOC.	FIELD LTH.	INV FILE	ASC FILE	INDEX NAME	ASSOCIATED NAME
ANCHOR			25				
ANCHOR	FLD1	11	6	0	0	--	--
ANCHOR	FLD2	17	8	1	0	INDEX1	--
ANCHOR	FLD3			0	1		ASSOC1
ANCHOR	FLD4			0	1		ASSOC2
ANCHOR	FLD5			0	1		ASSOC1
ANCHOR	KEY	4	7	0	0	--	--
ASSOC1			25				
ASSOC1	FLD3	11	10	0		--	
ASSOC1	FLD5	23	4	1		INDEX2	
ASSOC1	KEY	4	7	0			
ASSOC2			21				
ASSOC2	FLD4	11	12	0			
ASSOC2	KEY	4	7	0			
INDEX1			12				
INDEX1	FLD2	4	8				
INDEX1	KEY	12	7N				
INDEX2			8				
INDEX2	FLD5	4	4				
INDEX2	KEY	8	7N				

CS-53202

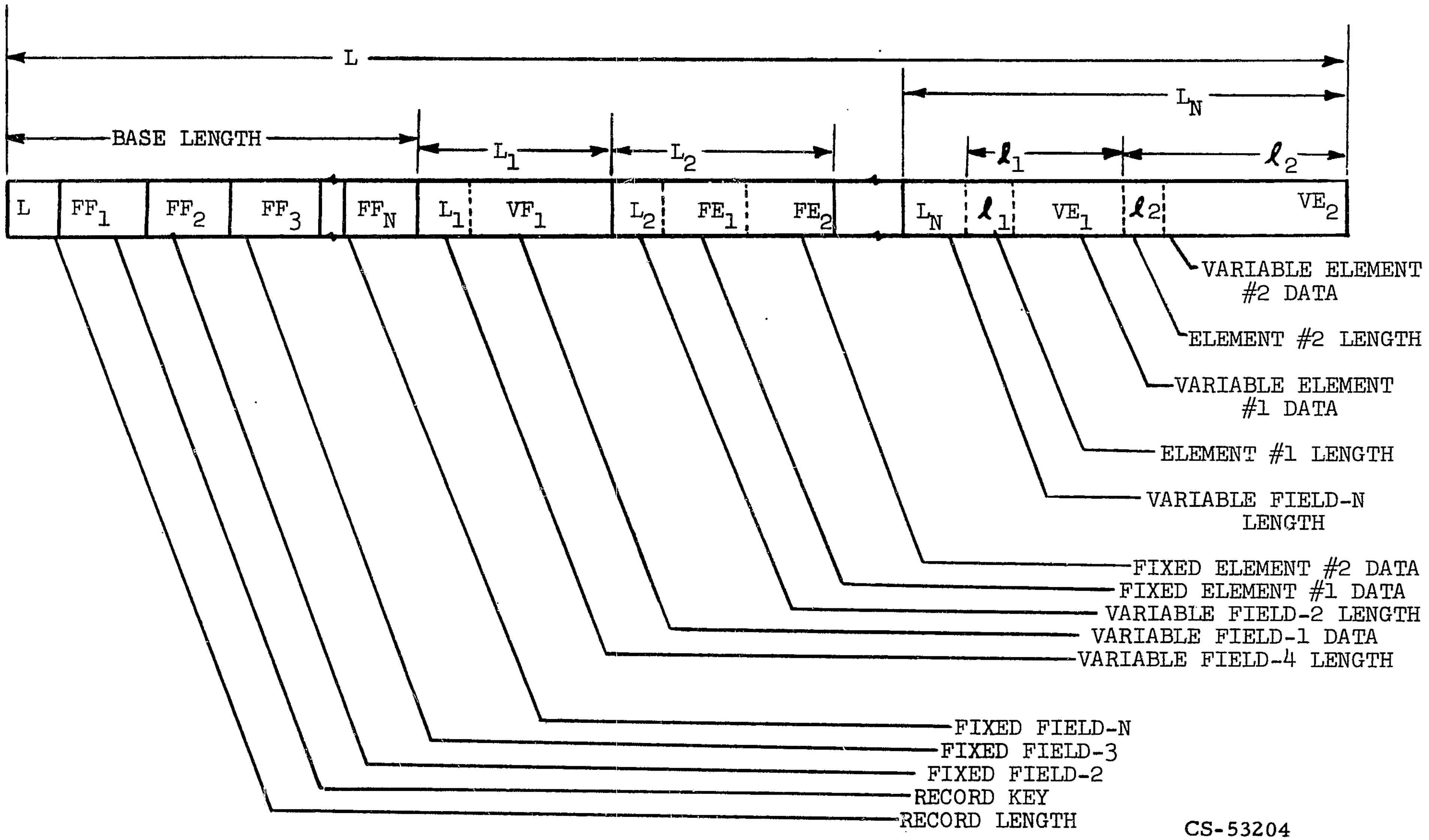
Figure 13



CS-53203

Figure 14

UNIVERSAL RECORD FORMAT



CS-53204

Figure 15

DBPAC

. PL/I INTERFACE BETWEEN USERS AND TSS DATA FILES

. PROVIDES USER WITH FACILITIES FOR:

1. FILE ACCESSIBILITY
2. RECORD RETRIEVAL AND POSTING
3. FIELD RETRIEVAL AND POSTING
4. LIST PROCESSING

CS-53205

Figure 16

DBPL/I

FILE LEVEL OPERATIONS

```
                                                    ┌┌DIRECT    ┐┐ ┌┌INPUT ┐┐
OPEN    FILE(file-name-1)    [TITLE (expression)]   ││SEQUENTIAL││ ││OUTPUT││
                                                    └└          ┘┘ │└UPDATE┘│
                                                                   └        ┘
        ┌                                           ┌┌DIRECT    ┐┐ ┌┌INPUT ┐┐┐
        │,FILE(file-name)    [TITLE (expression)]   ││SEQUENTIAL││ ││OUTPUT│││...  ;
        └                                           └└          ┘┘ │└UPDATE┘││
                                                                   └        ┘┘
CLOSE   FILE(file-name)    [ERASE]  [,FILE(file-name)    [ERASE]]   ...  ;

                                   ┌ SYSTEM                     ┐
ON      ERRORFILE(file-name)       └ GO TO  statement-label     ┘   ;
```

MISCELLANEOUS OPERATIONS

```
% INCLUDE  DB;

FINISH;
```

DBPL/I

RECORD LEVEL OPERATIONS

```
LOCATE   FILE(file-name)    KEYFROM(expression);

                            ┌┌KEY(expression)    ┐┐
READ     FILE(file-name)    ││BACKWARDS          ││ [NOLOCK]  ;
                            └└LIST(list-pointer) ┘┘

UNLOCK   FILE(file-name);

GET      FILE(file-name)    RECORD    INTO(variable) ;

WRITE    FILE(file-name)    FROM(variable) ;
```

CS-53206

Figure 17

DBPL/I

FIELD LEVEL OPERATIONS

```
GET   FILE(file-name)  FIELD(field-name-1 [,field-name-2,...] ) INTO(variable-1 [,variable-2,...] );
PUT   FILE(file-name)  FIELD(field-name-1 [,field-name-2,...] ) FROM(expression-1 [,expression-2,...] );
REPUT FILE(file-name)  FIELD(field-name-1 [,field-name-2,...] ) FROM(expression-1 [,expression-2,...] );
#FIELD¹ (file-name , field-name)
```

DBPL/I

LIST PROCESSING OPERATIONS

```
GET FILE(file-name)  LIST SET (list-pointer);
GET LIST(list-pointer)  KEY  INTO(variable);
                         ⎧ '|' ⎫
LIST  (list-pointer-1,   ⎨ '&' ⎬, list-pointer-2);
                         ⎩ '-' ⎭
#LIST¹ (list-pointer)
FREE  LIST [(list-pointer-1 [, list-pointer-2] ...)];
```

¹ FUNCTION REFERENCES

Figure 18

CS-53207

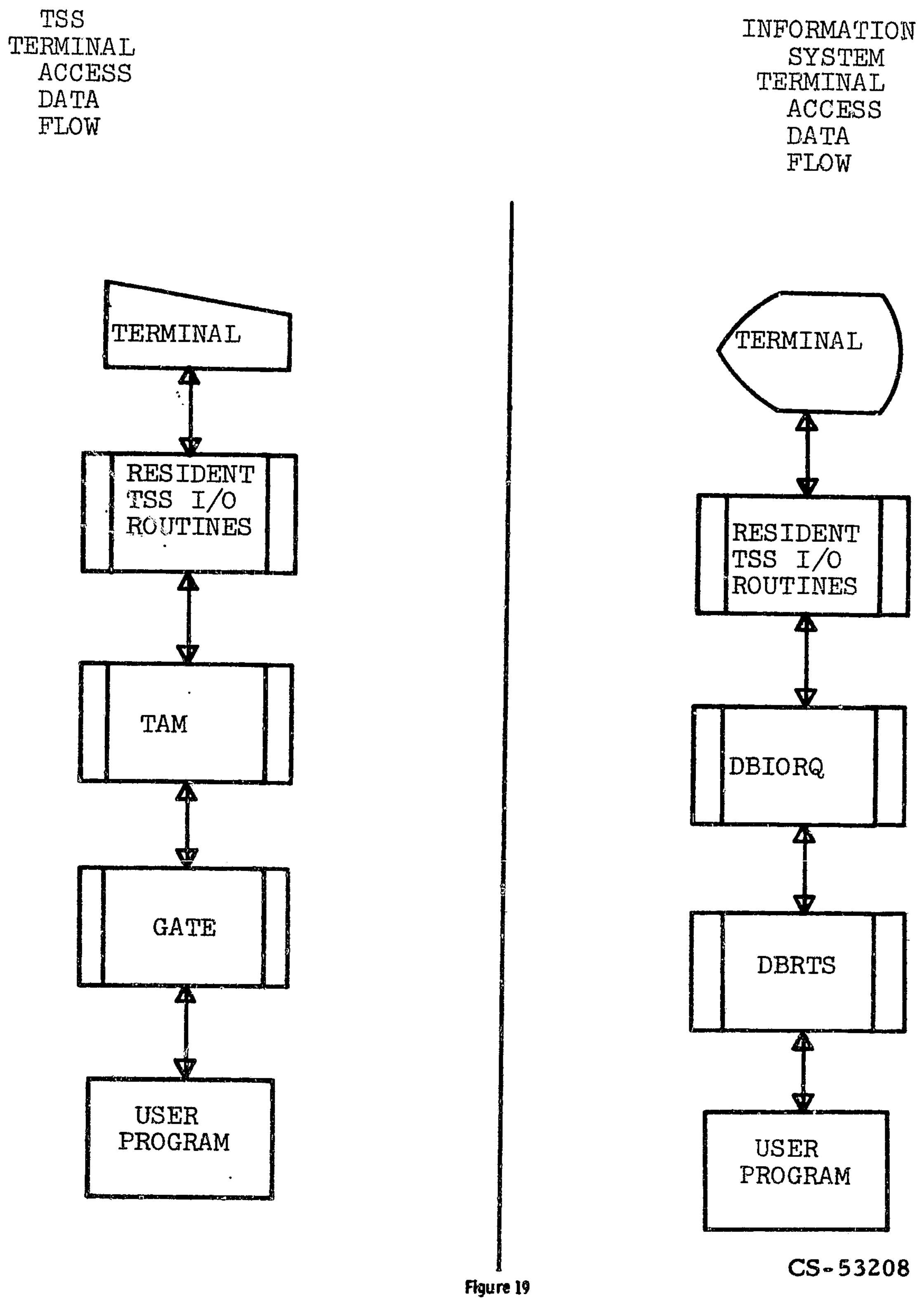

TSS
TERMINAL
ACCESS
DATA
FLOW
TERMINAL
RESIDENT TSS I/O ROUTINES
TAM
GATE
USER PROGRAM
INFORMATION
SYSTEM
TERMINAL
ACCESS
DATA
FLOW
TERMINAL
RESIDENT TSS I/O ROUTINES
DBIORQ
DBRTS
USER PROGRAM
CS-53208

Figure 19

TERMINAL SUPPORT

- MAINLINE
- OPEN
- CLOSE
- READ
  - LINE
  - SCRN
- WRITE
  - ROLL
  - PAGE
  - WRAP
  - INIT
  - LINE
  - SCRN
  - SAVE
  - TEXT
  - PRMT
- CONTROL

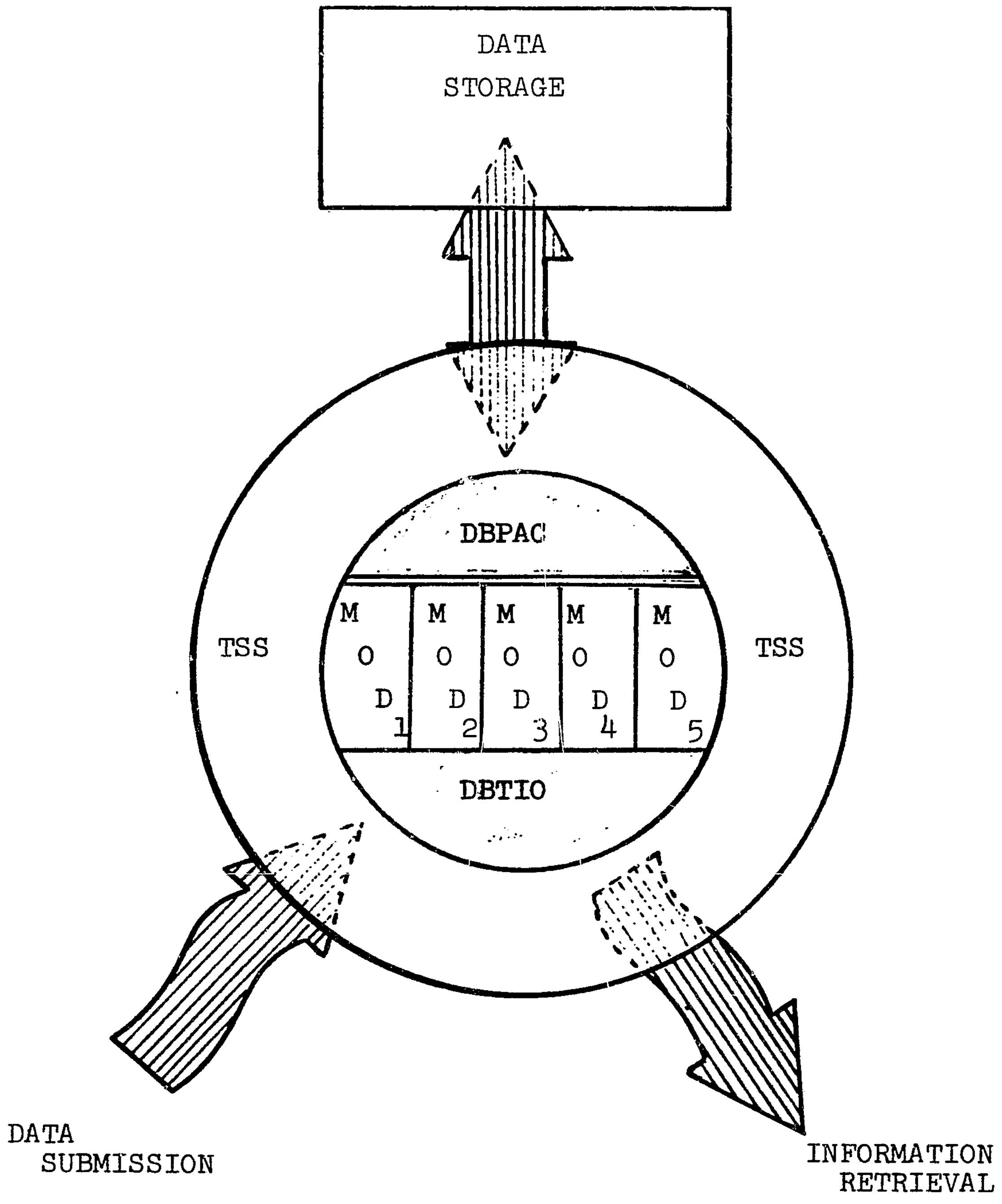
CS-53209

Figure 20

```
                                     TSPL/I

WRITE   (character-expression);

READ    (character-variable);

                                               [ {(character-variable)     } ]
PROMPT  (msgid  [,character-expression]...)    [ {                         } ]  ;
                                               [ { CODE (binary-variable)  } ]

                 [SYSTEM                  ]
ON  TSERROR      [                        ]
                 [GO TO statement-label   ]
```

CS-53210

Figure 21



LEWIS
INFORMATION STORAGE AND RETRIEVAL
SYSTEM

CS-53211

Figure 22