1-1-1983

# Merging of bibliographic data bases in the leader retrieval system.

Steven Alan Russell

# MERGING OF BIBLIOGRAPHIC DATA BASES IN THE LEADER RETRIEVAL SYSTEM

by

Steven Alan Russell

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Information Science

Lehigh University

1983

This thesis is accepted and approved in partial fullfillment of the requirements for the degree of Master of Science.

May 12, 1983
Date

Professor in Charge

Chairman of the Department

**Acknowledgements**

I wish to thank Dr. Donald J. Hillman for the many opportunities he has given me and for the valuable experience I have gained while employeed at the Center for Information and Computer Science. I also wish to thank Dr. Robert T. Meadowcroft of Marist College, who provided the key input for the completion of this project. But the people I owe most of my gratitude to are those who put up with my constant change of moods while I was trying to get this thing done. So to Mom, Dad, Henz, Bryce, Kerry, Mac, Rick, Dana, and most especially to Carol, I offer my deepest appreciation.

Table of Contents

## List of Figures

Abstract

The LEADER Document Retrieval System consists of two components, both of which are currently programmed in Pascal. The retreival component, which is responsible for processing user queries and returning relevant documents, is well suited for the language. The connectivity component, which is responsible for producing internally stored information, consists of file sorts and merges. These applications are not Pascal oriented. Another problem with Pascal is that lack of standardization causes lack of portability. This project involved the programming of connectivity in 1974 ANSI COBOL and producing the documentation necessary for its implementation. The final product is a highly structured, highly portable, and easily maintained system which provides the input necessary for the retrieval component of LEADER. This paper contains descriptions of all the programs, plus instructions for the use of the LEADER retrieval files.

## 1. Introduction

The LEADER Document Retrieval System was originally developed at Lehigh University in the late 1960's and early 1970's. Programmed in FORTRAN for a CDC 6000 series machine, LEADER was totally machine dependent and, due to the technology of the day, required huge amounts of storage space and processing power. The system was very well received and continued in operation into the mid 1970's. In 1981, work began on converting LEADER to run on a DECSYSTEM-20 with the possibility of moving the retrieval portion onto a minicomputer. The new version was written entirely in Pascal and, due to lack of standardization in the language, was once again machine dependent.

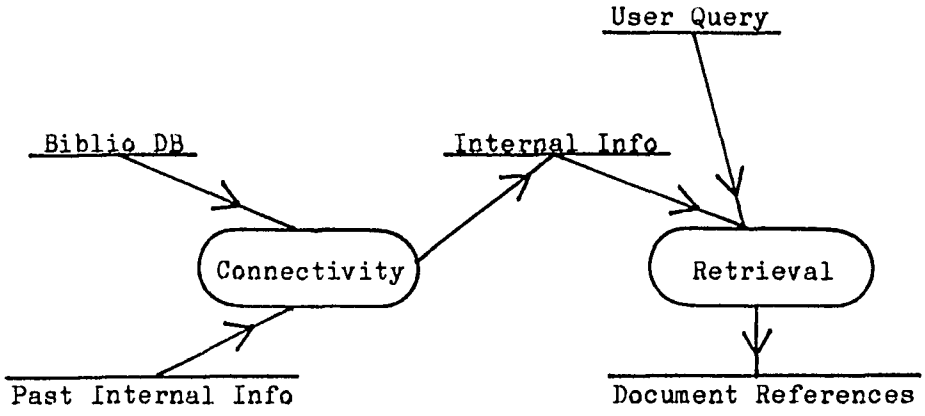



Figure 1-1: The LEADER Retrieval System

In both versions, LEADER consisted of two components. In the retrieval component, user requests are broken down into structures which can be recognized by the system. These structures are merged

with internally stored information to return documents for the user to choose from. The data structures provided by Pascal are very well suited to the processing which is performed in the retrieval component. Therefore, it was decided to leave retrieval in Pascal and go through the conversion when changing machines.

The connectivity component is responsible for producing the internally stored information which user queries are compared with. Connectivity is strictly a sequential application consisting mostly of file sorts and merges. Input consists of past connectivity runs and tapes provided by bibliographic data base vendors. In general, each tape consists of document references which include title, author, citation, CODEN designation, abstract, and a number of phrases which reference each document. After processing is completed, ten retrieval files are produced. These files are transferred onto the retrieval machine and serve as input to the system.

The second version of connectivity was also written in Pascal. There were a number of problems with the choice of this language. First, the input/output capabilities of Pascal are poorly suited to an application which deals strictly with file processing. Second, the use of Pascal is not widespread enough to allow connectivity runs to take place on a large majority of machines. And third, as was mentioned above, lack of standardization makes conversion a

difficult and tedious task.

There were a number of problems encoutered during the programming of the second version of connectivity due to lack of documentation from the first version. The only materials available were FORTRAN listings, file specifications, and operating instructions. Therefore, it was necessary to perform a substantial amount of additional work to finish version two. And like version one, the amount of documentation produced for version two was minimal.

This project was undertaken to provide solutions for the above problems. The programs necessary for connectivity have been rewritten in 1974 ANSI COBOL which provides input/output and file processing capabilities superior to those of Pascal. The new version of connectivity is a highly structured and highly portable system which produces a set of ten retrieval files necessary for the execution of the LEADER Retrieval System. What follows in the remainder of this report is the documentation necessary to implement and maintain connectivity on any machine which offers 1974 ANSI COBOL. Also included is a description of the function of the LEADER retrieval files.

## 2. Glossary of Unique Terms

A number of terms which are unique to LEADER must be defined to facilitate understanding of this report.

**Index Term:** A phrase which serves as an identifier for a given document. In LEADER, index terms are supplied along with each document reference. Each document must have at least one index term. Note that index terms are referred to as phrases throughout the programs.

**Affiliation:** The notion of affiliation can be expressed as follows. If X is an index term for document D then X is affiliated with D and D is affiliated with X.

**Affiliation Weight:** The value of an affiliation between a document and an index term. These weights are determined by what type of index term is being looked at. Normally, the bibliographic data base vendors provide different types of index terms for each document. For example, Engineering Index provides main headings, sub headings, cross reference terms, and free language terms. Each would be assigned a different affiliation weight.

**Association:** If X and Y are index terms and both are affiliated with document D then X is associated with Y and Y is associated with X.

**Association Weight:** The value of an association between two index terms. When two terms are initially associated, the association weight is defined to be the product of their affiliation weights. For each subsequent association of the same two terms, the association weight is equal to the sum of the old association weight and the last association weight minus the product of the two.

**Stem:** A stem is a word which has been extracted from an index term and conforms to the following characteristics. First, a stem must be at least three characters in length. Second, if a word ends in 'S', that character is truncated making the word into a stem. Third, the stem may not appear in the stop list (includes words such as THE, AND, BUT, FOR, etc...). And fourth, a word which is greater than fifteen characters in length is truncated to a stem of fifteen characters.

**Stem Position:** This denotes the position of a stem within an index term. For the purpose of comparing stems with user queries, stems are numbered from right to left. For example, the rightmost stem in any index term will be in position one.

**Association Weight:** The value of an association between two index terms. When two terms are initially associated, the association weight is defined to be the product of their affiliation weights. For each subsequent association of the same two terms, the association weight is equal to the sum of the old association weight and the last association weight minus the product of the two.

**Stem:** A stem is a word which has been extracted from an index term and conforms to the following characteristics. First, a stem must be at least three characters in length. Second, if a word ends in 'S', that character is truncated making the word into a stem. Third, the stem may not appear in the stop list (includes words such as THE, AND, BUT, FOR, etc...). And fourth, a word which is greater than fifteen characters in length is truncated to a stem of fifteen characters.

**Stem Position:** This denotes the position of a stem within an index term. For the purpose of comparing stems with user queries, stems are numbered from right to left. For example, the rightmost stem in any index term will be in position one.

## 3. Program Descriptions

What follows is a general description of the processing which occurs in the connectivity programs. For more detail, Appendix I and Appendix II can be referenced for structure charts and the system flow chart. Note that each program, except for the sorts, produces a file called STAT-FILE. This file contains statistics for the current connectivity run and must be printed out after execution of the program or else the next program will write over it. Also, in the input/output section of each program description, the name following the actual file name is the physical location of the file on a DECSYSTEM-20 directory. The listings for these programs are on file at the Lehigh University Center for Information and Computer Science.

### 3.1 Sorts A - E

These programs are responsible for sorting transaction files before they are used as input to a merge program. SORTA sorts the phrase transactions in phrase text order with the secondary key being document number. SORTB sorts the partial phrase key file in phrase number order. SORTC sorts the document term affiliation transactions into phrase number order within document number order. SORTD sorts the phrase association transactions in associated phrase number order within phrase number order. And SORTE sorts the stem transactions alphabetically by stem with phrase number and position

as a concatenated secondary key. Each of these programs uses the SORT feature offered in 1974 ANSI COBOL. Note that this feature will only be available on machines which have a sorting utility.

The only sort which differs from the others is SORTA. This is because SORTA outputs variable length phrases. Rather than using the full 150 characters alotted for each phrase, only the actual characters in the phrase are written. This processing occurs in the output procedure of the program.

### 3.2 EILOAD: Process the COMPENDEX Tape

```
INPUT   : COMPENDEX-FILE
          SKELETON-FILE (SKEL.CON)
OUTPUT  : NOUN-PHRASE-FILE (CNFLO1.CON)
          SKELETON-FILE (SKEL.CON)
          STAT-FILE (STATS.CON)
EXTEND  : DOCUMENT-FILE (CY3DC1.NEW)
          PARTIAL-DOCUMENT-KEY (CNFL2O.CON)
```

For EILOAD to run properly, entries have to exist for SKELETON-FILE and the files which are extended. If a previous connectivity run has taken place, those files will contain information from the past run. Otherwise, they will be empty.

When the program starts off, the operator is given a choice between three options, either process the entire tape, process the first n records, or process the last n records. After the choice is made, the appropriate number of documents is skipped, if any. Then the last document number assigned is read from SKELETON-FILE. This

is necessary for assignment of numbers to those documents which will be read off the current tape. If SKELETON-FILE is empty, the last document number assigned will be zero.

The tape consists of fixed length records of 2046 characters each. Within those records are variable length documents. Each document begins with a directory which gives the length of each of the eighteen available fields in characters. After the directory is read into an eighteen place array, processing of the document can begin. The fields processed, in order, are title, main heading, sub heading, author, citation, CODEN, affiliation, abstract, Card-A-Lert codes, cross references terms, and free language terms. The title, author, citation, CODEN, affiliation and abstract are written directly to DOCUMENT-FILE. Information concerning the lengths of these fields along with their location within DOCUMENT-FILE is written to PARTIAL-DOCUMENT-KEY.

The main heading and the sub heading are concatenated into one phrase. This phrase, along with the Card-A-Lert codes, cross reference and free language terms are written to NOUN-PHRASE-FILE along with the affiliated document number and the affiliation weight, The main/sub heading is assigned a weight of eight, Card-A-Lerts a weight of five, cross references a weight of seven, and free language terms a weight of six.

After processing of the tape is completed the last document number assigned is written to SKELETON-FILE for use in future connectivity runs. Final statistics are compiled and written to STAT-FILE. Errors, which will mainly be truncated phrases, will also be reported in STAT-FILE.

### 3.3 ZERO: Merge Phrases

```
INPUT : NOUN-PHRASE-TRAN (CNFLO2.CON)
        OLD-NP-MASTER (CNFLO4.CON)
        CARD-A-LERTS (CALCS.TXT)
        SKELETON-FILE (SKEL.CON)
OUTPUT: NEW-NP-MASTER (CNFLO5.CON)
        DOCUMENT-TERM-TRANS (CNFL10.CON)
        NEW-PHRASE-FILE (CNFL23.CON)
        SKELETON-FILE (SKEL.CON)
        STAT-FILE (STATS.CON)
```

Program ZERO is responsible for merging transaction phrases with previously existing phrases. Included in this process is the writing of two other files which serve as input to other connectivity programs. Processing begins with retrieval of the last phrase number assigned from SKELETON-FILE. For an initial run, this value will be zero.

The only difference between the transaction file and the master files is that the master files have a PHRASE-NUMBER field. Otherwise, both are sorted on DOCUMENT-NUMBER within PHRASE-TEXT. A copy of the last record written to the new master file is kept for the purpose of assigning phrase numbers to transaction phrases. The name of this record is PREVIOUS-RECORD.

There are two modules which write records to the new master file. 232-PROCESS-OLD-MASTER copies the old master record into PREVIOUS-RECORD and into the new master record which is then written to the new master file. 231-PROCESS-TRANSACTION is slightly more involved. If a transaction phrase is a Card-A-Lert code, the actual text of the phrase has to be retrieved from CARD-A-LERTS. The transaction phrase text is compared to the text of the previous phrase. If they are the same, the transaction receives the same phrase number as the previous phrase. Otherwise, the last phrase number is incremented and assigned to the transaction record and the transaction phrase is written to the NEW-PHRASE-FILE. After the phrase number is assigned, an affiliation record containing the transaction's document number and phrase number is written to DOCUMENT-TERM-TRANS. Then the transaction, along with its phrase number, are copied into PREVIOUS-RECORD and the new master record which is written to the new master file.

In the merge, the first record is read from both OLD-NP-MASTER and NOUN-PHRASE-TRAN. The phrase texts are compared to determine which record is to be processed. In short, the algorithm is expressed below.

```
IF transaction phrase < old master phrase
     Process transaction
     Read next transaction  ELSE
IF old master phrase < transaction phrase
     Process old master
     Read next old master ELSE
IF transaction document < old master document
```

```
        Process transaction
        Read next transaction  ELSE
    IF old master document < transaction document
        Process old master
        Read next old master.
```

As soon as the end of either file is reached, the other file is processed sequentially until it is empty. After the merge is completed, the last phrase number assigned is written to SKELETON-FILE. Then the statistics report is compiled and written to STAT-FILE.

### 3.4 ONE: Create Phrase Dictionary

```
INPUT : NOUN-PHRASE-MASTER (CNFL05.CON)
OUTPUT: PARTIAL-PHRASE-KEY (CNFL08.CON)
        PHRASE-TEXT-FILE (CY2RT2.NEW)
        DOCUMENT-STORAGE (CY4RT2.NEW)
        STAT-FILE (STATS.CON)
```

Program ONE processes the phrase master file sequentially and produces two retrieval files, along with a partial key for those files. For each record on the phrase master file, the document number and affiliation weight are written to DOCUMENT-STORAGE. The number of these records written for each phrase is kept track of and eventually becomes NUMBER-AFFILIATED of PARTIAL-PHRASE-KEY. DOCUMENT-LOCATION will be the record number of the first DOCUMENT-STORAGE record written for each phrase number.

The phrase text of each record is compared with the phrase text of the previous record. If they are not the same, the text of the

new record is written character by character to a page of PHRASE-TEXT-FILE. If tne length of the page becomes greater than 512, the page is written to the file and the counter is reset to one. Tne PAGE-NUMBER and LOCATION fields of PARTIAL-PHRASE-KEY represent the location of the phrase within the PHRASE-TEXT-FILE. LOCATION is the index of the first character of the phrase within the current PAGE-NUMBER.

### 3.5 TWO: Document Term Affiliation Merge

```
INPUT : DOCUMENT-TERM-TRANS (CNFL11.CON)
        OLD-DOCUMENT-TERM (CNFL12.CON)
OUTPUT: NEW-DOCUMENT-TERM (CNFL13.CON)
        NEW-ASSOCIATION-FILE (CNFL14.CON)
        STAT-FILE (STATS.CON)
```

Program TWO is not actually a merge. Since the input files are sorted on phrase number within document number, and the lowest document number on DOCUMENT-TERM-TRANS is greater than the highest document number on OLD-DOCUMENT-TERM, the transaction file is just appended to the end of the old master file.

First, OLD-DOCUMENT-TERM is processed sequentially with each record being written to NEW-DOCUMENT-TERM. Then, the same process is carried out with DOCUMENT-TERM-TRANS, along with some additional processing. Before a transaction record is written to the new master file, information to produce associations is gathered. There exists a fifteen place array which contains the phrase numbers affiliated with a single document. When a record containing a new

document number is read from DOCUMENT-TERM-TRANS, associations for the previous document are written to NEW-ASSOCIATION-FILE in the following manner:

```
FOR index-1 = 1 TO number-affiliated DO
  FOR index-2 = 1 TO number-affiliated DO
    IF index-1 NOT EQUAL TO index-2
      create-association([index-1] and [index-2])
```

When the merge is completed, statistics are compiled and written to STAT-FILE.

### 3.6 THREE: Associated Phrase Merge

```
INPUT : TR-ASSN-FILE (CNFL15.CON)
        OM-ASSN-FILE (CNFL16.CON)
OUTPUT: NM-ASSN-FILE (CNFL17.CON)
        STAT-FILE (STATS.CON)
```

In program THREE, TR-ASSN-FILE and OM-ASSN-FILE are merged to produce OM-ASSN-FILE. All three association files are sorted on associated phrase number within phrase number. The merge is very straight forward. If the TR-ASSN-FILE record is less than the OM-ASSN-FILE record, the TR-ASSN-FILE record is written to the new master file, and vice versa. When a transaction record is written to the new master file, a new field is added in. REAL-WEIGHT is the ASSOCIATION-WEIGHT of TR-ASSN-FILE divided by 100. Also, if the TR-ASSN-FILE record and the OM-ASSN-FILE record are equal, a new weight is calculated and the next transaction record is read. The formula for the new weight is given below:

(ASSOCIATION-WEIGHT/100 + REAL-WEIGHT) -

(ASSOCIATION-WEIGHT/100 * REAL-WEIGHT)

When the end of either file is reached, the remainder of the other file is sequentially processed until it is empty. Then statistics are compiled and written to STAT-FILE.

## 3.7 FOUR: Create Phrase Retrieval Files

```
INPUT : PARTIAL-PHRASE-KEY (CNFL09.CON)
        ASSOCIATION-MASTER (CNFL17.CON)
OUTPUT: PHRASE-KEY-FILE (CY1RT2.NEW)
        ASSOCIATION-STORAGE (CY3RT2.NEW)
        STAT-FILE (STATS.CON)
```

Program FOUR computes the number of associations and the location of the first association for each phrase. These fields are added on to PARTIAL-PHRASE-KEY forming PHRASE-KEY-FILE. For each phrase, the processing is as follows. Assuming the next ASSOCIATION-MASTER record is always available, a record is read from PARTIAL-PHRASE-KEY. The field PK-ASSOCIATION-LOCATION is set to the current record number in ASSOCIATION-STORAGE. The phrase numbers from PARTIAL-PHRASE-KEY record and ASSOCIATION-MASTER record are compared. If they are equal, ASSOCIATED-NUMBER and REAL-WEIGHT of ASSOCIATION-MASTER are written to ASSOCIATION-STORAGE, PK-NUMBER-ASSOCIATED is incremented, and the next ASSOCIATION-MASTER record is read. The same procedure is carried out until the phrase numbers are not equal. At that time, the PARTIAL-PHRASE-KEY record, along with the fields PK-ASSOCIATION-LOCATION and PK-NUMBER-ASSOCIATED, is written to PHRASE-KEY-FILE and the next PARTIAL-PHRASE-KEY record is read.

Processing has been completed when the end of PARTIAL-PHRASE-KEY has been reached. If there are any records left in ASSOCIATION-MASTER at this point, an error message will be displayed to the screen. The error will be "PREMATURE END OF PARTIAL PHRASE KEY FILE." The results of the run should be checked out manually to make sure the error will not effect the rest of the connectivity run. After the error check is performed, statistics are compiled and written to STAT-FILE.

### 3.8 FIVE: Create Document Retrieval Files

```
INPUT : PARTIAL-DOCUMENT-KEY (CNFL20.CON)
        AFFILIATION-MASTER (CNFL13.CON)
OUTPUT: DOCUMENT-KEY-FILE (CY2DC1.NEW)
        AFFILIATION-STORAGE (CY4DC1.NEW)
        STAT-FILE (STATS.CON)
```

Program FIVE computes the number of affiliated phrases and the location of the first affiliated phrase for each document. These fields are added on to PARTIAL-DOCUMENT-KEY forming DOCUMENT-KEY-FILE. The processing for each document is the same as the processing for each phrase in program FOUR, except that affiliated phrases and documents are being processed rather than associated phrases and phrases.

## 3.9 SIX: Phrase Analysis

INPUT : NEW-PHRASE-FILE (CNFL23.CON)
STOP-LIST-FILE (CNFL24.CON)
OUTPUT: NEW-STEM-FILE (CNFL25.CON)
STAT-FILE (STATS.CON)

Program SIX is responsible for breaking phrases up into stems and recording information about the occurrence of each stem within a phrase. The phrases to be analyzed are located on NEW-PHRASE-FILE, along with their phrase numbers and lengths. Each phrase is read in as a 150 character array and is processed one character at a time. As long as the characters are alphanumeric, they are entered into a fifteen place array which represents a stem. If the length of the stem gets to be greater than fifteen, the last characters are truncated. As soon as a non-alphabetic character is reached, the stem packing procedure is called. In this procedure, a trailing 'S' is truncated and the stem array is strung into a word which is compared with the words in STOP-LIST-FILE. If the word does not occur in STOP-LIST-FILE, it is added to an array of stems occurring in the current phrase.

When the end of a phrase is reached, the array containing all the stems is written to NEW-STEM-FILE. The fields in NEW-STEM-FILE record are determined as follows. PHRASE-NUMBER is the number of the phrase which has just been analyzed. NUMBER-OF-WORDS is the number of entries in the array containing all the stems for this phrase. STEM-POSITION is equal to one plus the index of the current

stem subtracted from the number of entries in the table.

After all the phrases have been analyzed, statistics are compiled and written to the STAT-FILE.

### 3.10 SEVEN: Stem File Merge

```
INPUT : STEM-TRANS-FILE (CNFL26.CON)
        OLD-STEM-MASTER (CNFL27.CON)
OUTPUT: NEW-STEM-MASTER (CNFL28.CON)
        STAT-FILE (STATS.CON)
WORK  : MERGE-FILE
```

Program SEVEN is a straight forward merge of STEM-TRANS-FILE and OLD-STEM-MASTER into NEW-STEM-MASTER. The MERGE feature of 1974 ANSI COBOL is used to accomplish this. After the merge is completed, an output procedure is called to check for errors and accumulate statistics. Upon completion of processing, statistics are written to STAT-FILE.

The files to be merged are sorted alphabetically in STEM order. The secondary key is a concatenation of PHRASE-NUMBER and STEM-POSITION.

### 3.11 EIGHT: Create Stem Retrieval Files

```
INPUT : STEM-MASTER-FILE (CNFL28.CON)
OUTPUT: PROFILE-FILE (CY4RT1.NEW)
        KEY-FILE (CY3RT1.NEW)
        INDEX-FILE (CY2RT1.NEW)
        STAT-FILE (STATS.CON)
```

Program EIGHT processes STEM-MASTER-FILE sequentially,

producing three stem retrieval files. For each record on STEM-MASTER-FILE, a profile is written to PROFILE-FILE which contains PHRASE-NUMBER, STEM-POSITION, and NUMBER-OF-WORDS, each of which comes from the STEM-MASTER-FILE record. Keys to PROFILE-FILE are stored in KEY-FILE. KE-STEM is the stem for which profiles are to be looked up. KE-STEM-PTR is the location of the first profile stored for KE-STEM in PROFILE-FILE. KE-NUMBER-RECS is the number of profiles which are stored for KE-STEM.

KEY-FILE tends to grow relatively large, so to facilitate searching of that file, an index is created. Every time twenty-six records are written to KEY-FILE, a record is written to INDEX-FILE. This record contains the last of the twenty-six stems previously written to the KEY-FILE, the location within KEY-FILE of the first of the twenty-six stems, and the number of records stored inclusively between the two. This value will be twenty-six, unless the last index record is being written for which it will be less than or equal to twenty-six. The value twenty-six was chosen because that many KEY-FILE records can fit on one page of memory at a time.

After processing is completed, statistics are compiled and written to STAT-FILE.

## 4. Performing a Connectivity Run

Note that since development of these programs occurred on a DECSYSTEM-20, some of the following instructions may seem machine oriented.

All of the record definitions for both the connectivity and retrieval files are located in a library which is stored on disk. Use of libraries is included in 1974 ANSI COBOL, but the procedure for their creation is a machine dependent function. Therefore, before a connectivity run can take place on a new machine, a library containing all the record definitions must be created.

For a connectivity run to be successfully completed, directory entries must previously exist for SKEL, CALCS, CNFL04, CNFL12, CNFL16, CNFL20, CNFL27, and CY3DC1. For an initial run, each of these will be empty files. For all subsequent runs, each will contain information from past connectivity runs.

The programs are executed in the following order.

1. EILOAD
2. SORTA
3. ZERO
4. ONE
5. SORTB

6. SIX

7. SORTE

8. SEVEN

9. EIGHT

10. SORTC

11. TWO

12. SORTD

13. THREE

14. FOUR

15. FIVE

After the run has been completed, the four new master files must be copied into the old master files for future connectivity runs. This is accomplished with the following commands:

```
RENAME (*FROM*) CNFL05 (*TO*) CNFL04
RENAME (*FROM*) CNFL13 (*TO*) CNFL12
RENAME (*FROM*) CNFL17 (*TO*) CNFL16
RENAME (*FROM*) CNFL28 (*TO*) CNFL27
```

These files, along with SKEL, CALCS, CNFL20, and CY3DC1 will serve as input to the next connectivity run.

Note that a number of transaction files can de deleted throughout the run, after they have served as input. Care must be taken in determining when deletion can occur. Also, if tapes other than COMPENDEX are to be run, their load programs can be executed right after EILOAD. These programs will append all the files which

are produced by EILOAD.

## 5. Maintenance and Portability

### 5.1 Non Standard Language Elements

There are three major machine dependencies in the new version of connectivity. The first occurs in EILOAD, where two of the output files are extended rather than rewritten. The statement used to perform this function is

```
OPEN EXTEND DOCUMENT-FILE PARTIAL-KEY-FILE
```

In changing machines, this statement may have to be altered according to the facilities offered for appending files.

The second machine dependency occurs in programs which write variable length phrases, specifically, EILOAD, SORTA, and ZERO. The facility for writing variable length fields is located in the record descriptions and is an extension of the OCCURS clause. The syntax for the extension is

```
PHRASE-TEXT  PICTURE IS X
             OCCURS 1 TO 150 TIMES
             DEPENDING ON PHRASE-LENGTH.
```

Each time one of these records is written, PHRASE-TEXT will take up PHRASE-LENGTH characters. Again, this syntax will have to be changed according to the version of COBOL being implemented.

In the FD statements, two non standard statements are added. They are

```
VALUE OF ID IS identifier
RECORDING MODE IS character-set
```

In the first statement, identifier is the name of the directory entry for which the file is listed. The directory entry consists of nine characters, the first six being the filename and the last three being the extension. In the second statement, character-set is the mode which the file is written in. All files, except for COMPENDEX-FILE in EILOAD, have ASCII as their recording mode. The recording mode of COMPENDEX-FILE is F which stands for fixed length industry compatible EBCDIC records.

## 5.2 Changes in Data Types

Appendix III contains data element descriptions for the connectivity and retrieval files. Each connectivity description lists the libraries, programs, and retrieval data elements which must be fixed for a data type change. To fix a library, the only change made will be in the record description for the data element being changed. For programs, previous records and statistical fields declared in WORKING-STORAGE will be the major fields which have to be changed. In the case of subscripted variables, certain indexes within the programs may also have to be changed. The RETRIEVAL-ITEMS field in the data element descriptions denotes the retrieval data elements which will be affected by changing the given connectivity data element. The libraries in which the retrieval items are located, are listed in the retrieval data element descriptions.

## 6. Use of the LEADER Retrieval Files

The retrieval files were defined using a relational approach. There are two distinct advantages to this approach. First, it is very useful for expressing single concepts along with their characteristics. And second, normalization of the relations reduces the entire structure down to one which has minimal redundancy and avoids harmful retrieval anomolies.

R

| KEY | A | B | C | D |

Figure 6-1: Form of a Relation

An example is shown in figure 6-1. The name of the relation is R. It has a primary key, which is underlined, and four attributes. The key uniquely defines an occurrence of R, while A, B, C and D are descriptors of that occurrence. In a set of relations, R would have a number of pointers entering or leaving it. A two headed arrow pointing to R represents multiple occurrences, whereas a single headed arrow represents a single occurrence.

There are a number of characteristics which are desirable for a set of relations. First, a relation should have no repeating groups. The reasons for this are that repeating groups cause unnecessary redundancy and the number in the group would have to remain static. To remove repeating groups, the following procedure

DOCUMENT

| DOCUMENT NUMBER | SOURCE | PHRASE NUMBER | PHRASE NUMBER | PHRASE NUMBER |
|---|---|---|---|---|

Figure 6-2: Relation With a Repeating Group

is performed. A document relation is shown in figure 6-2. To remove the repeating group, a new relation is created and the structure in figure 6-3 is produced.

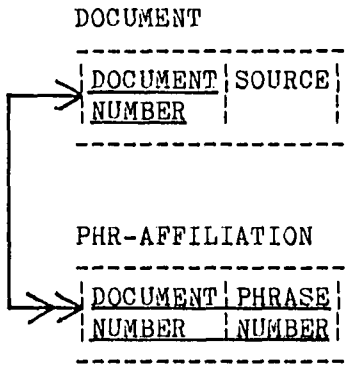



Figure 6-3: Relation in First Normal Form

Note that figure 6-3 is a logical representation, not a physical one. Physically, the location of the first PHR-AFFILIATION occurrence and the number of occurrences for that document would be stored in the DOCUMENT relation. Therefore, the actual physical structure is shown in figure 6-4.

A relation without repeating groups is said to be in first normal form. The other characteristics which are desirable are for the relation to be in second and third normal form. For a relation

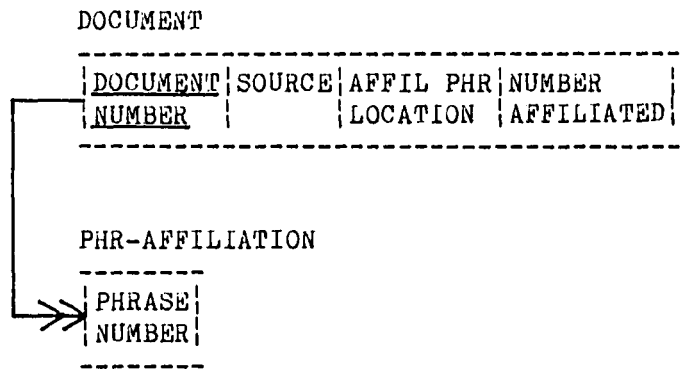



Figure 6-4: Physical Representation of a Relation

to be in second normal form, it must be in first normal form and every attribute must be fully determined by the primary key of the relation. For a relation to be in third normal form, it must be in second normal form with all transitive dependencies removed. Given three attributes of relation R, A, B, and C, a transitive dependency occurs if B is determined by A and C is determined by both A and B. To remove the dependency, the relation is replaced by two relations, one with B determined by A, and the other with C determined by B.

The relations represented in the retrieval files have been put into third normal form and are implemented in a partially inverted file structure. The organization of the files is shown in figure 6-5. Note that in PHRTXT, DOCTXT, PHRASE and DOCMNT, the keys are not actually stored, but the records are in referential position. That means that to retrieve phrase number 87, record number 87 must be accessed in PHRASE.

There are six retrieval functions which can be performed with this type of structure. They are initial phrase retrieval, document retrieval, affiliated phrase retrieval, associated phrase retrieval, phrase text retrieval, and document text retrieval. Each is described in detail below.

## 6.1 Initial Phrase Retrieval

Initial phrase retrieval is the entry point into the retrieval system. Specifically, a user query undergoes the same process which index terms are subjected to in program SIX. The resulting structure is shown in figure 6-6.

This structure is merged with STMIDX which is an index into the STMKEY relation. An index match occurs when STEM of QUERY is alphabetically less than STEM of STMIDX. If there is an index match, RECORD NUMBER of STMINX is the location of the first stem on the desired page. That location is randomly accessed and the first NUMBER OF KEYS records are retrieved. Those records are searched for an exact match between STEM of QUERY and STEM of STMKEY. If a match occurs, a record is written to a new relation expressed in figure 6-7.

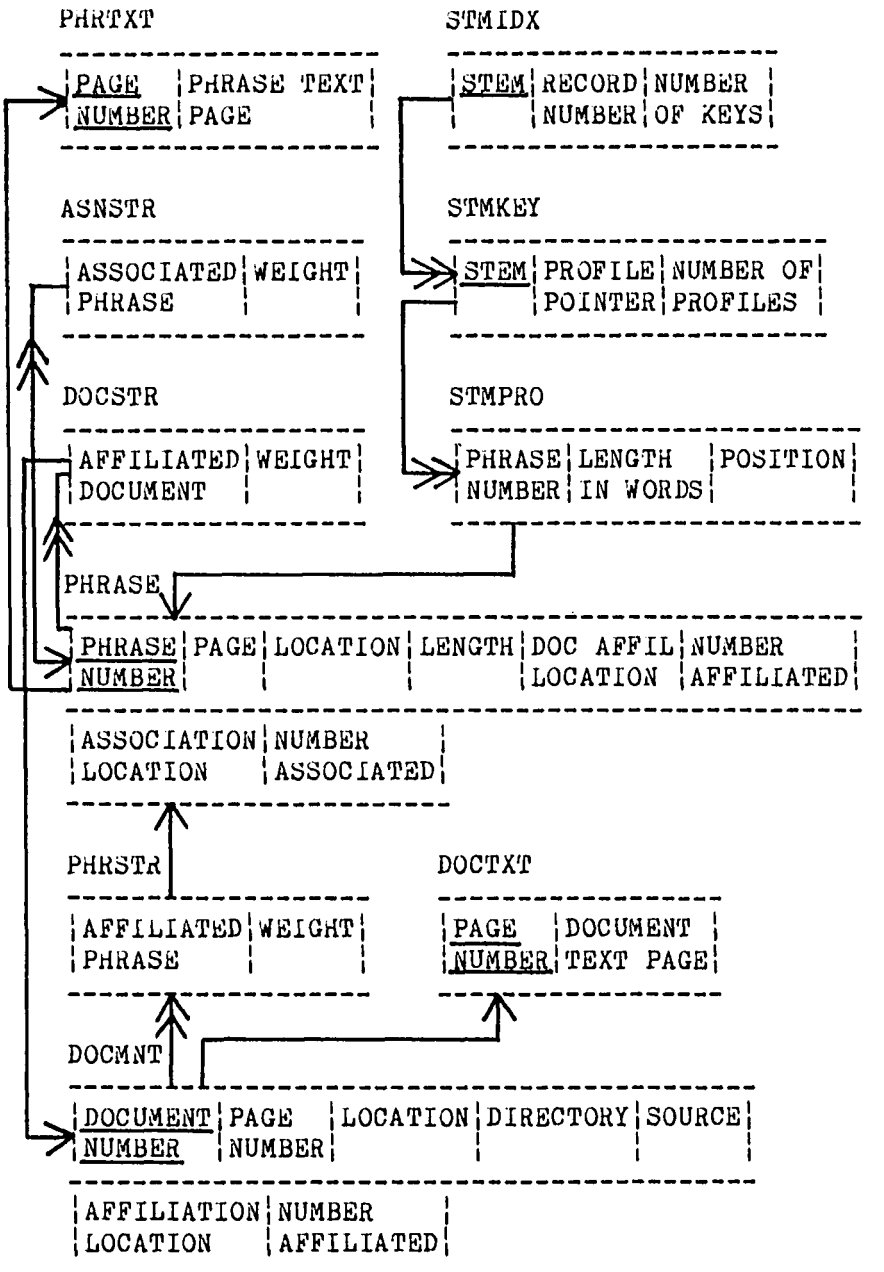



Figure 6-5: Organization of the Retrieval Files

QUERY

| STEM | PHRASE NUMBER | POSITION | LENGTH IN WORDS |
|---|---|---|---|

Figure 6-6: Query Relation

MATCH

| STEM | QUERY PHR # | QUERY POSITION | QUERY LENGTH | PROFILE POINTER | NUMBER OF PROFILES |
|---|---|---|---|---|---|

Figure 6-7: Match Between System and Query Stems

The MATCH relation is processed sequentially. PROFILE POINTER is the location of the first system profile for STEM. That loction is accessed and NUMBER OF PROFILES records are retrieved from STMPRO. For each of these records, a new relation , shown in figure 6-8, is built.

PROFILES

| NUMBER OF PROFILES | QUERY | | | SYSTEM | | |
|---|---|---|---|---|---|---|
| | PHRASE NUMBER | POSITION | LENGTH | PHRASE NUMBER | POSITION | LENGTH |

Figure 6-8: Profile of System/Query Stem Match

When completed, this structure is subjected to a weight calculation procedure where a similarity measure is computed for each system phrase. This procedure is explained in detail by Meadowcroft [5]. The result is a new relation which serves as the

input to phrase text retrieval. The new relation is shown in figure 6-9.

IPR

| SYSTEM PHRASE NUMBER | WEIGHT |
|---|---|

Figure 6-9: Final Output of Initial Phrase Retrieval

## 6.2 Document Retrieval

The input to document retrieval is a list of phrases which have been selected by the user, along with their weights. First, the list is sorted into PHRASE NUMBER order and then is processed sequentially. For each PHRASE NUMBER, the PHRASE relation is accessed and the relation shown in figure 6-10 is built.

DOCINFO

| PHRASE NUMBER | WEIGHT | DOC AFFIL LOCATION | NUMBER AFFILIATED |
|---|---|---|---|

Figure 6-10: Affiliated Document Relation

DOC AFFIL LOCATION is the location of the first document in DOCSTR affiliated with PHRASE NUMBER. That location is accessed and NUMBER AFFILIATED records are retrieved. For each affiliation, a weight is calculated and assigned to each AFFILIATED DOCUMENT. This weight is equal to the product of WEIGHT of DOCSTR and WEIGHT of

DOCINFO. If the same AFFILIATED DOCUMENT is retrieved more than once, the weights are added using a probabilistic sum which is the product of the weights subtracted from the sum of the weights. The resulting structure is shown in figure 6-11.

DOCRET

| AFFILIATED DOCUMENT | WEIGHT |
|---|---|

Figure 6-11: Retrieved Document Relation

The top two hundred weights are stored and serve as input to document text retrieval.

## 6.3 Affiliated Phrase Retrieval

The processing in affiliated phrase retrieval is basically the same as document retrieval. The input consists of a list of documents which have been chosen by the user, along with their weights. The list is sorted by DOCUMENT NUMBER and then processed sequentially. For each DOCUMENT NUMBER, the DOCMNT relation is accessed and the relation shown in figure 6-12 is created.

AFFPHRINFO

| DOCUMENT NUMBER | WEIGHT | PHR AFFIL LOCATION | NUMBER AFFILIATED |
|---|---|---|---|

Figure 6-12: Affiliated Phrase Information

PHR AFFIL LOCATION is the record number of the first PHRASE NUMBER in PHRSTR affiliated with DOCUMENT NUMBER. That location is accessed and NUMBER AFFILIATED records are retrieved. Each AFFILIATED PHRASE is assigned a WEIGHT equal to the product of WEIGHT of PHRSTR and WEIGHT of AFPHRINFO. The result is the relation shown in figure 6-13.

AFFRET

| AFFILIATED PHRASE | WEIGHT |
|---|---|

Figure 6-13: Output of Affiliated Phrase Retrieval

Once again, the records having the top two hundred weights are stored and are used as input to phrase text retrieval.

## 6.4 Associated Phrase Retrieval

Associated phrase retrieval uses the same type of processing as the previous two retrieval types. The input consists of a list of phrases which have been selected by the user. The list is sorted by PHRASE NUMBER and then processed sequentially. For each PHRASE NUMBER, the PHRASE relation is accessed and the relation shown in figure 6-14 is created.

ASSOCIATION LOCATION represents the location of the first phrase in ASNSTR associated with PHRASE NUMBER. That location is accessed and NUMBER ASSOCIATED records are retrieved. Each

ASPHRINFO

| PHRASE NUMBER | WEIGHT | ASSOCIATION LOCATION | NUMBER ASSOCIATED |
|---|---|---|---|

Figure 6-14: Associated Phrase Relation

ASSOCIATED PHRASE is assigned a WEIGHT equal to the product of WEIGHT of ASPHRINFO and WEIGHT of ASNSTR. If an ASSOCIATED PHRASE appears more than once, its weights are summed using the probabilistic sum described in section 6.2. The resulting relation is shown in figure 6-15.

ASNRET

| ASSOCIATED PHRASE | WEIGHT |
|---|---|

Figure 6-15: Output of Associated Phrase Retrieval

Again, the records having the top two hundred weights are saved and used as input to phrase text retrieval.

## 6.5 Document and Phrase Text Retrieval

The input to phrase/document text retrieval consists of a list of phrase/document numbers and their assigned weights. To access the text the following procedure is performed. The appropriate record from PHRASE/DOCMNT is accessed randomly. The PAGE NUMBER field in PHRASE/DOCMNT is the location of the record within which the phrase/document text is located in PHRTXT/DOCTXT. LOCATION is the index of the first character of the phrase/document on the

PHRASE/DOCUMENT TEXT PAGE. To retrieve phrase text, return the next LENGTH of PHRASE characters. To retrieve document text, check the DIRECTORY field of DOCMNT. This field consists of six integers, each representing the length of a different field within a document. In both cases, if the page index ever becomes greater than five hundred twelve, read the next PHRASE/DOCUMENT TEXT PAGE and change the value of the index to one.

## Bibliography


[1] Hillman, Donald J.
Negotiation of Inquiries in an On-Line Retrieval System.
Information Storage and Retrieval 4:219-238, 1968.

[2] Martin, James.
Computer Data-Base Organization.
Prentice-Hall, Inc., Englewood Cliffs, 1977.

[3] Meadowcroft, Robert T.
A Model and a Method for Phrase Matching in the LEADERMART System.
Master's thesis, Lehigh University, 1972.

## I. Connectivity Program Structure Charts

Structure charts are provided for the following connectivity programs:

- SORTA: Sort Phrase Master File Transactions
- EILOAD: Load Engineering Index
- ZERO: Phrase Merge
- ONE: Create Phrase Dictionary
- TWO: Document Term Affiliation Merge
- THREE: Association Merge
- FOUR: Create Phrase Retrieval Files
- FIVE: Create Document Retrieval Files
- SIX: Phrase Analysis
- SEVEN: Stem Merge
- EIGHT: Create Stem Retrieval Files

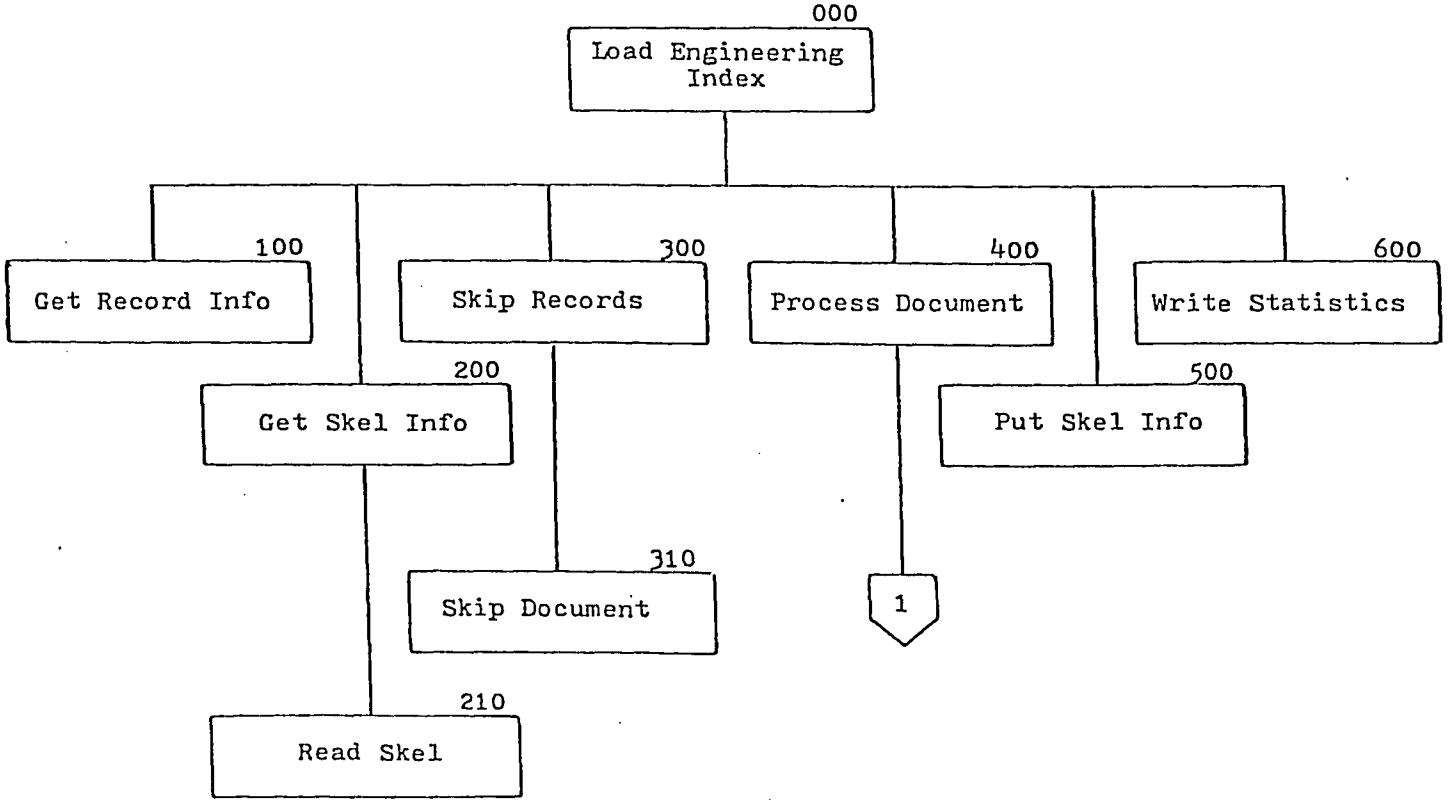
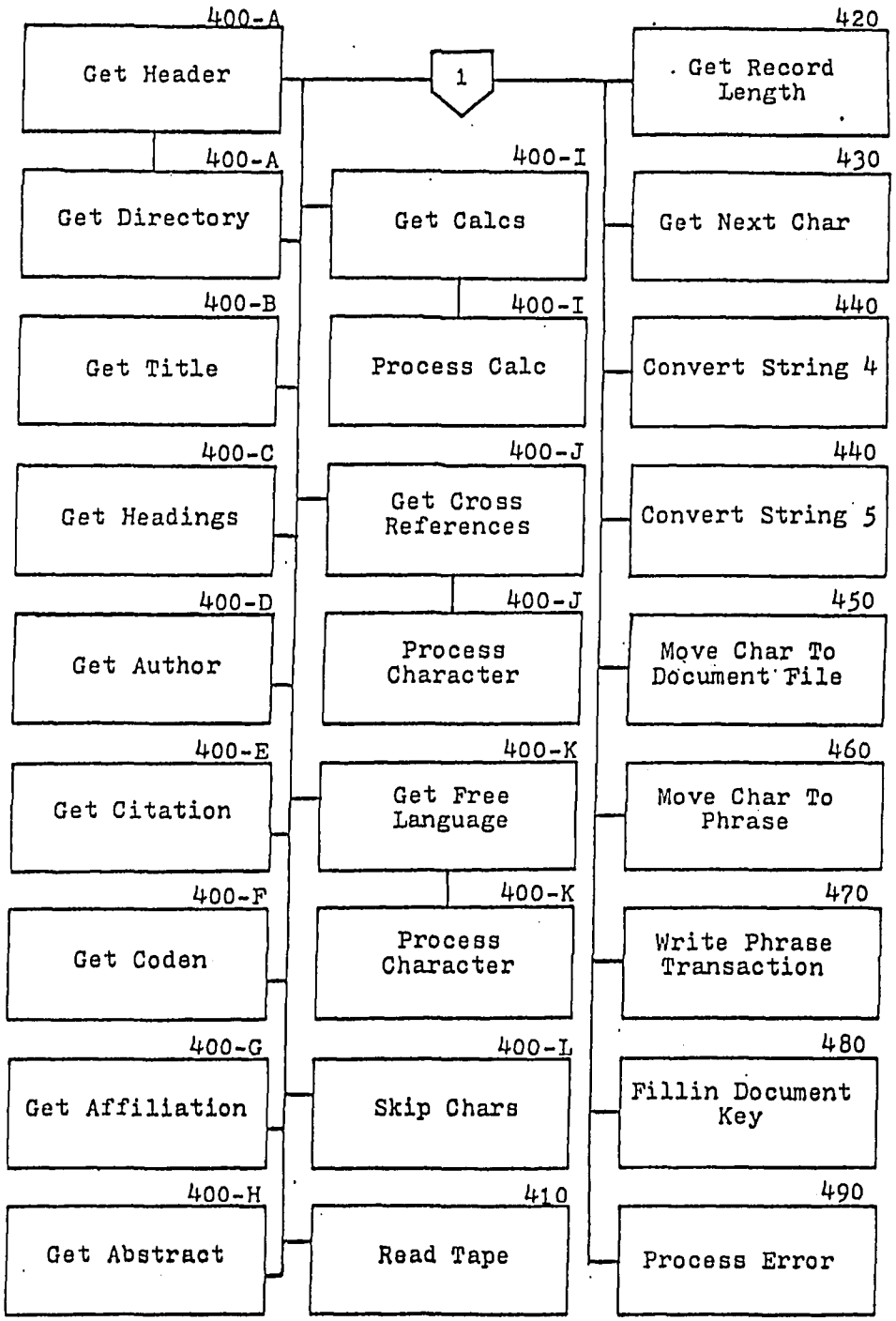



Figure 7-1: EILOAD


400-A
Get Header
1
420
Get Record Length
400-A
Get Directory
400-I
Get Calcs
430
Get Next Char
400-B
Get Title
400-I
Process Calc
440
Convert String 4
400-C
Get Headings
400-J
Get Cross References
440
Convert String 5
400-D
Get Author
400-J
Process Character
450
Move Char To Document File
400-E
Get Citation
400-K
Get Free Language
460
Move Char To Phrase
400-F
Get Coden
400-K
Process Character
470
Write Phrase Transaction
400-G
Get Affiliation
400-L
Skip Chars
480
Fillin Document Key
400-H
Get Abstract
410
Read Tape
490
Process Error

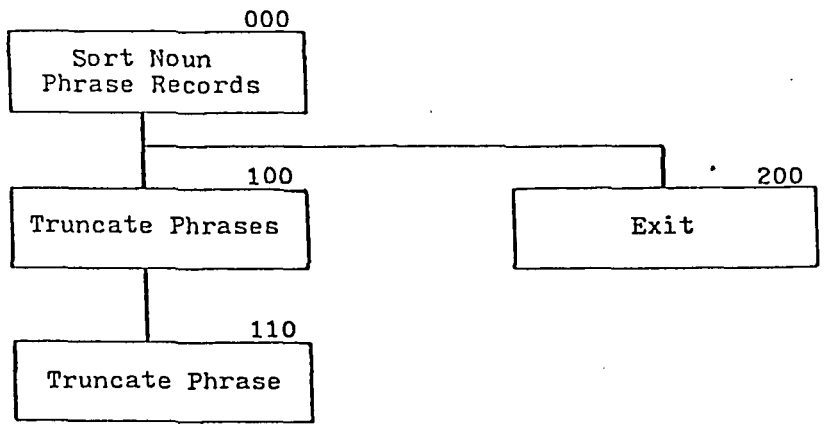



Figure 7-2: SORTA

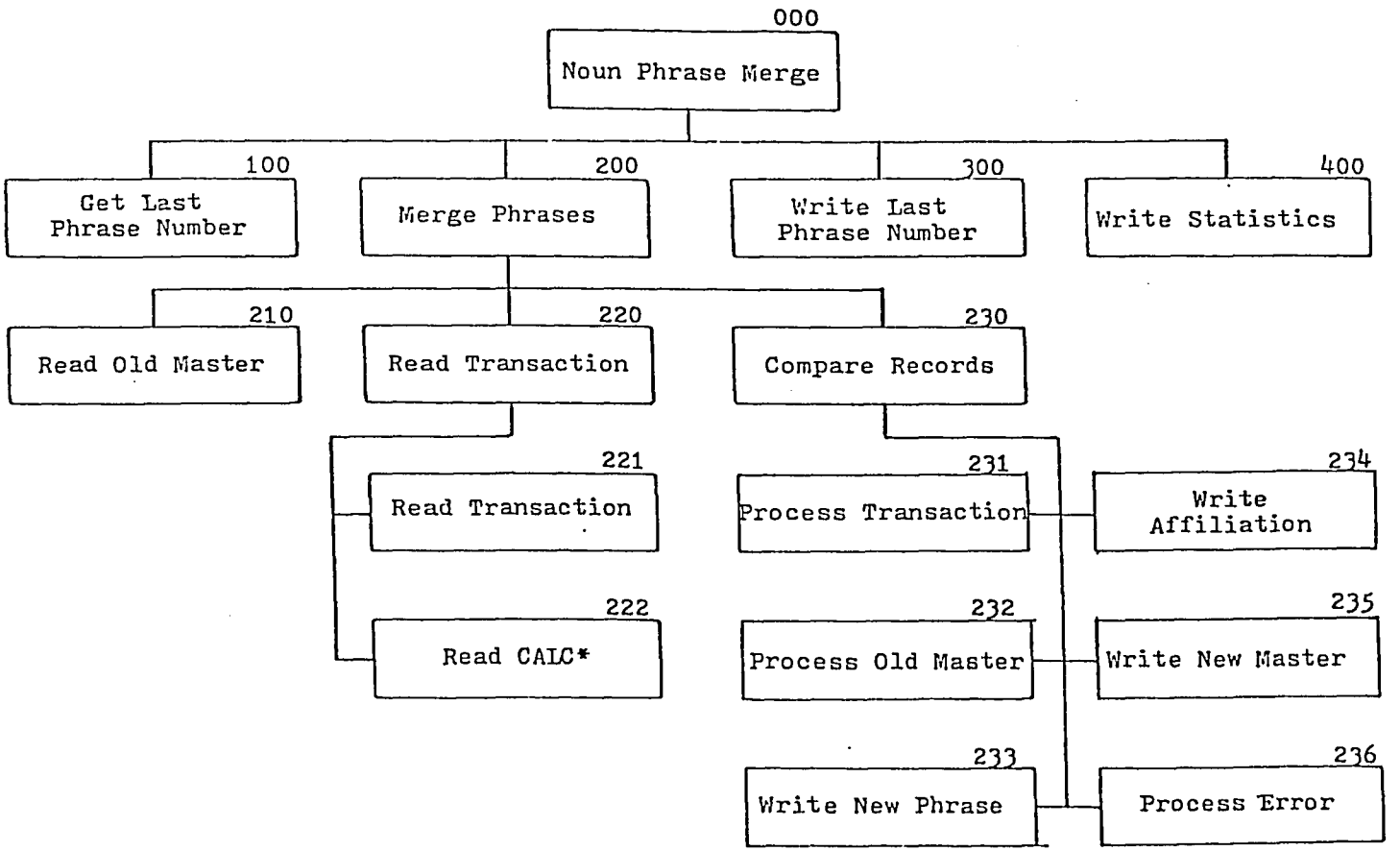



Figure 7-3: ZERO

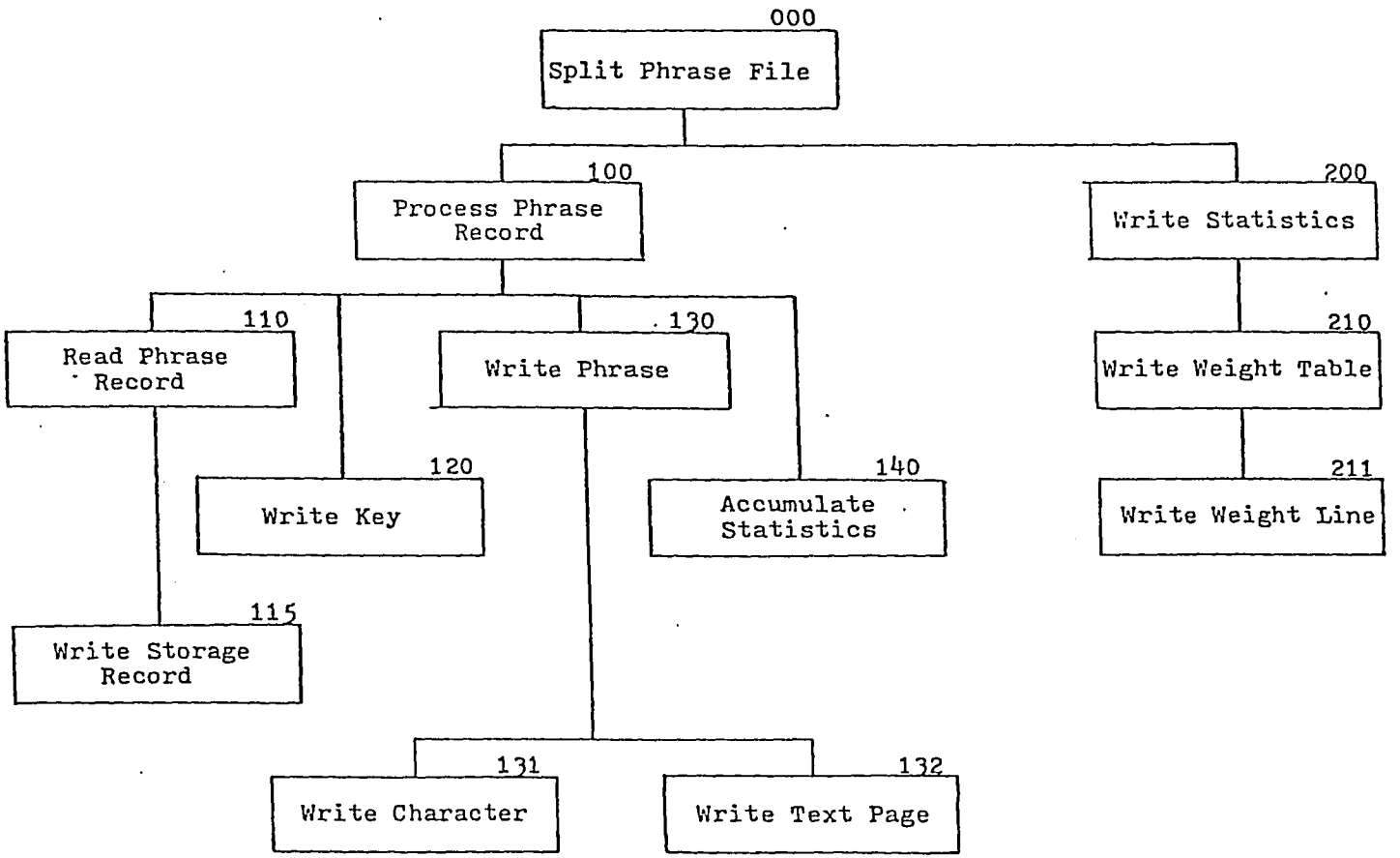



Figure 7-4: ONE

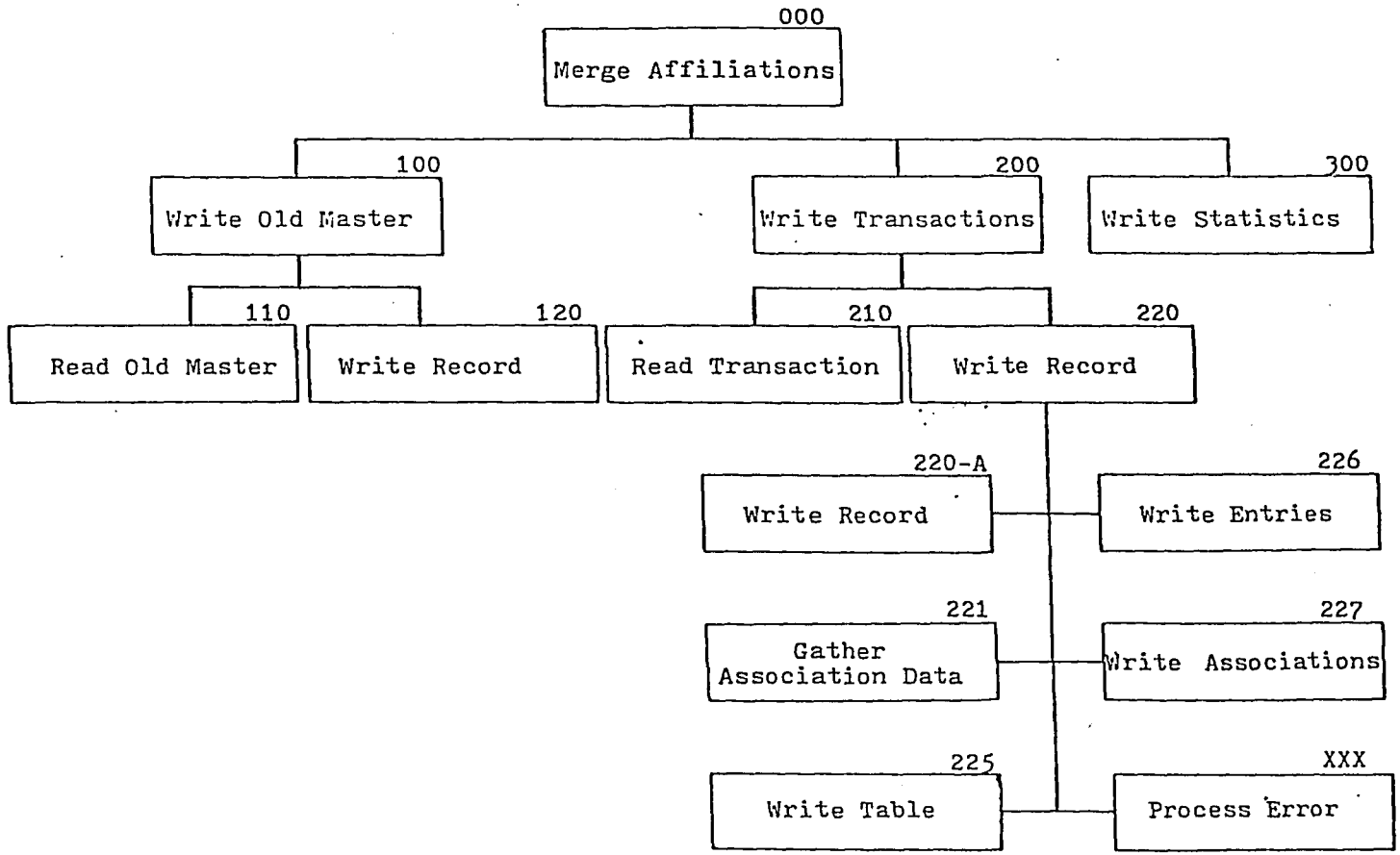



Figure 7-5: TWO

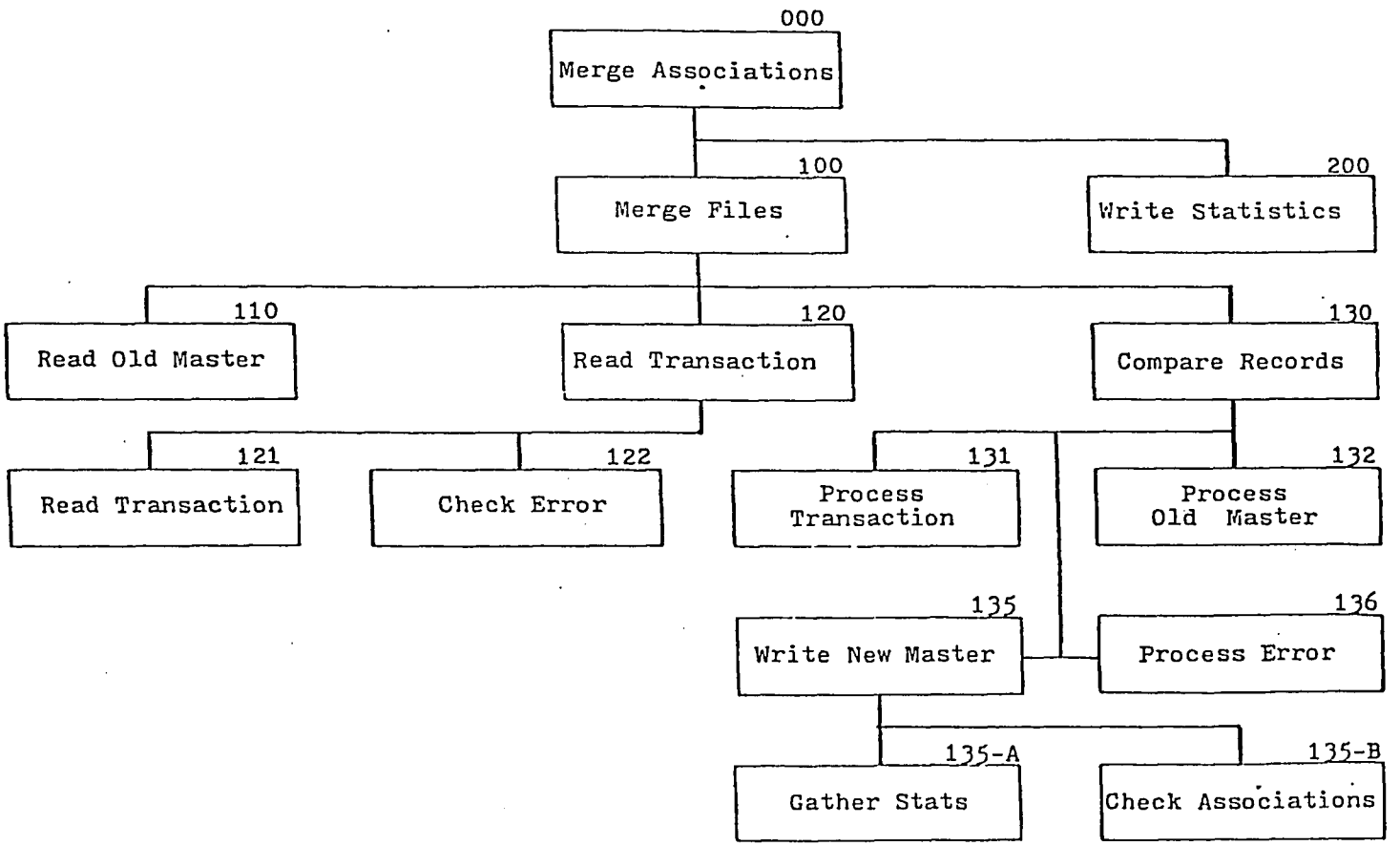



Figure 7-6: THREE

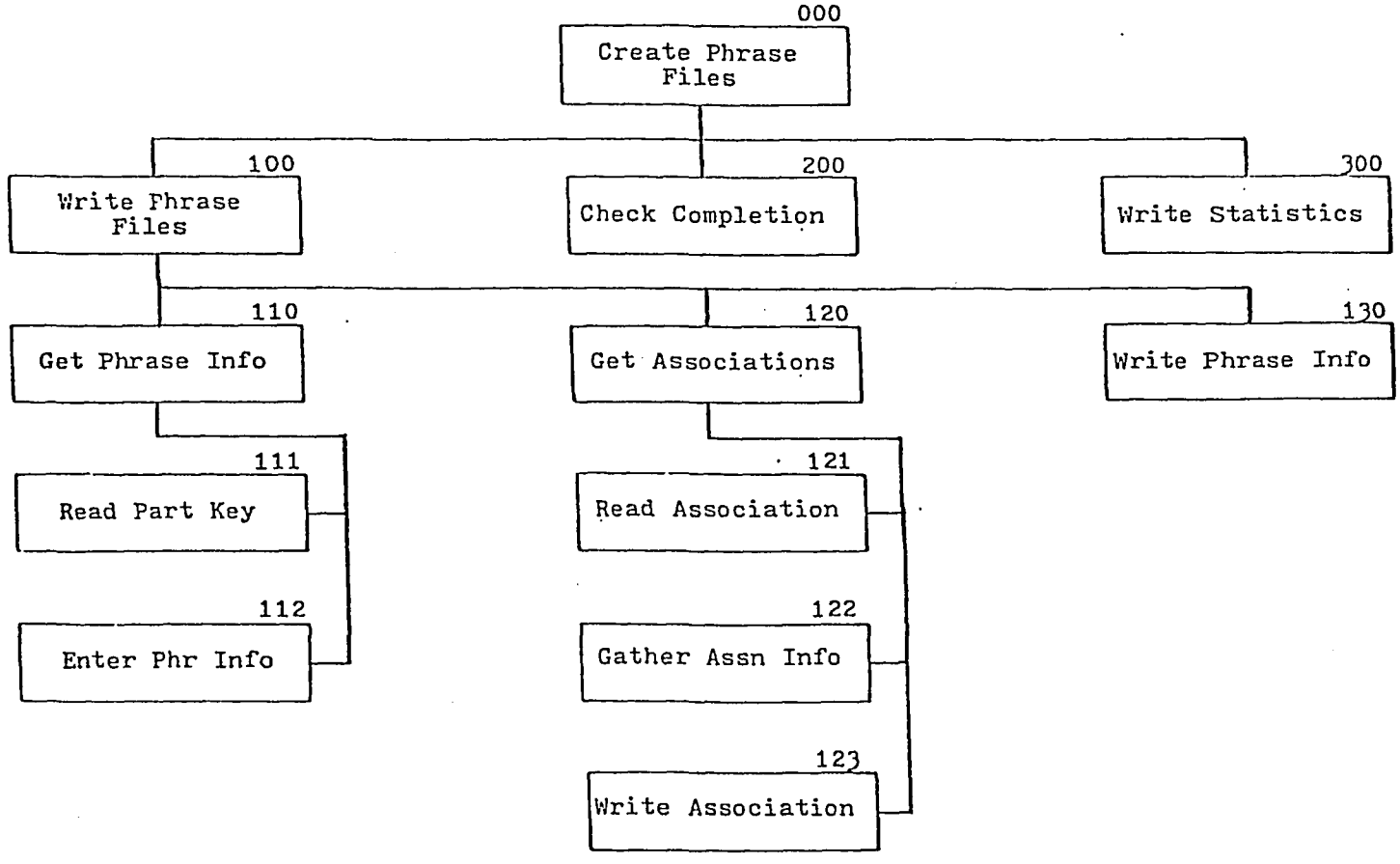



Figure 7-7: FOUR

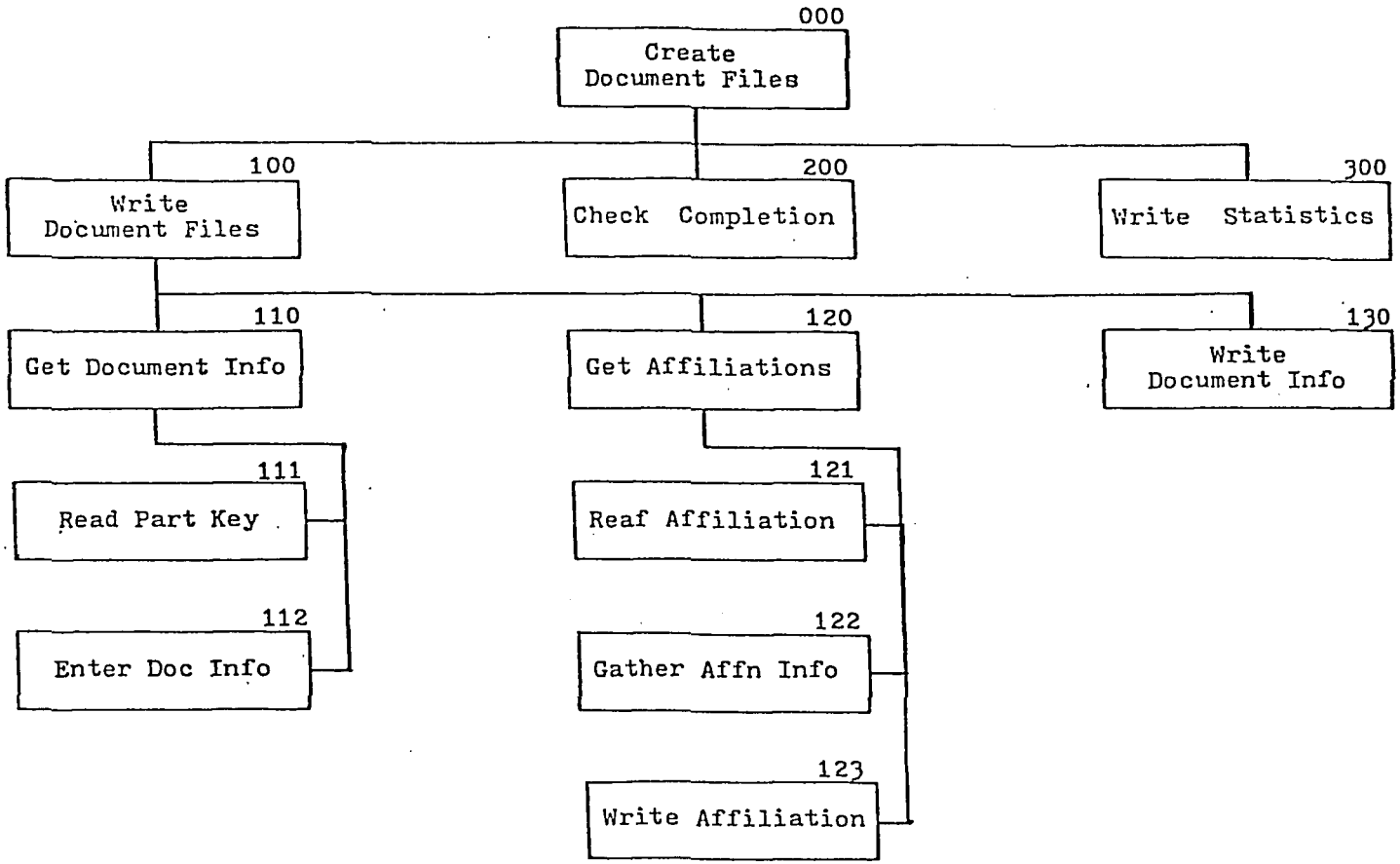



Figure 7-8: FIVE

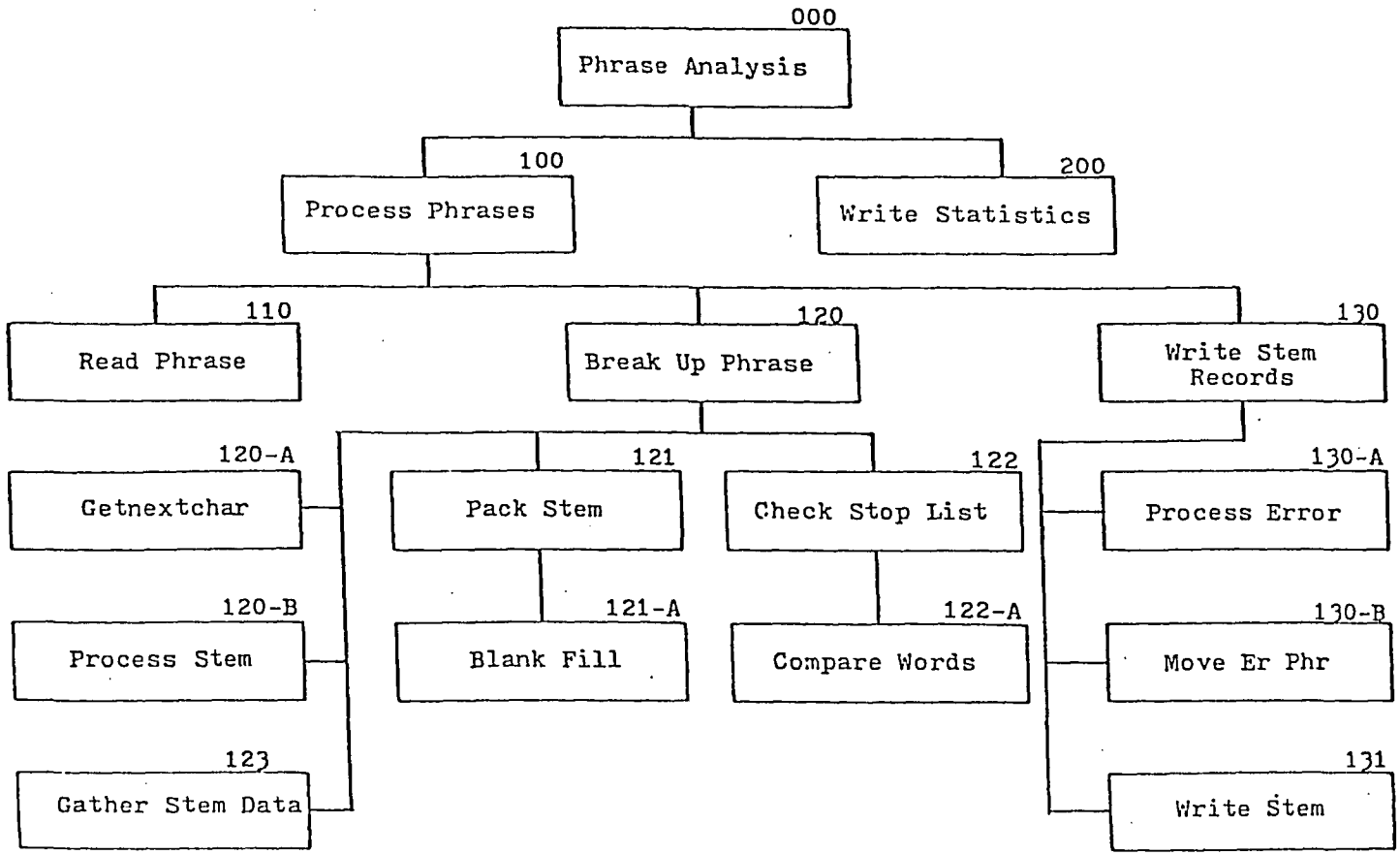



Figure 7-9: SIX

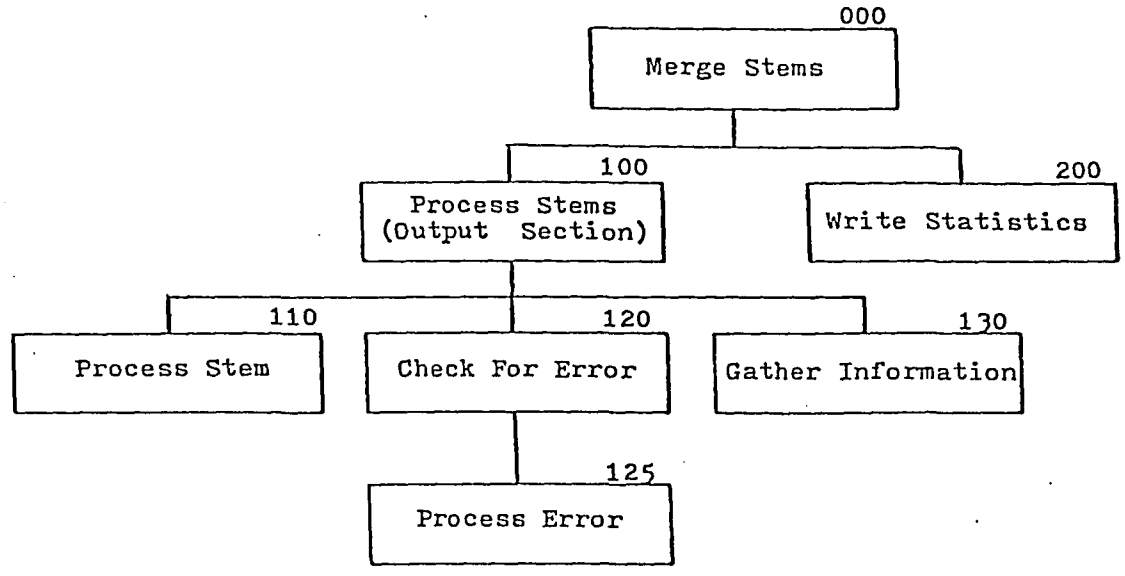



Figure 7-10: SEVEN

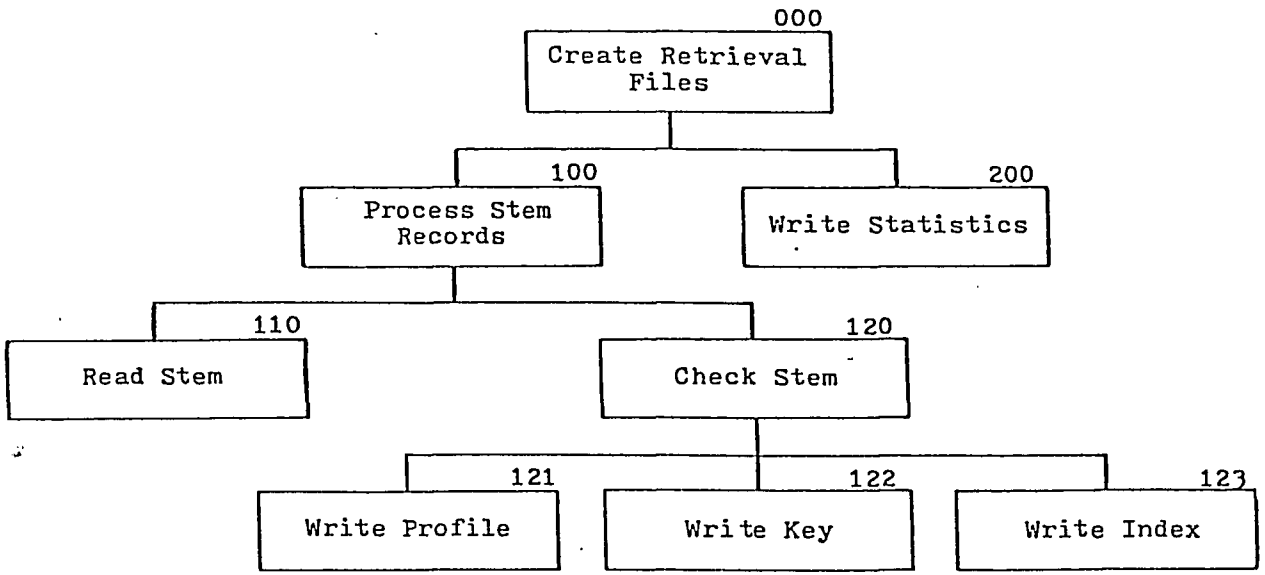
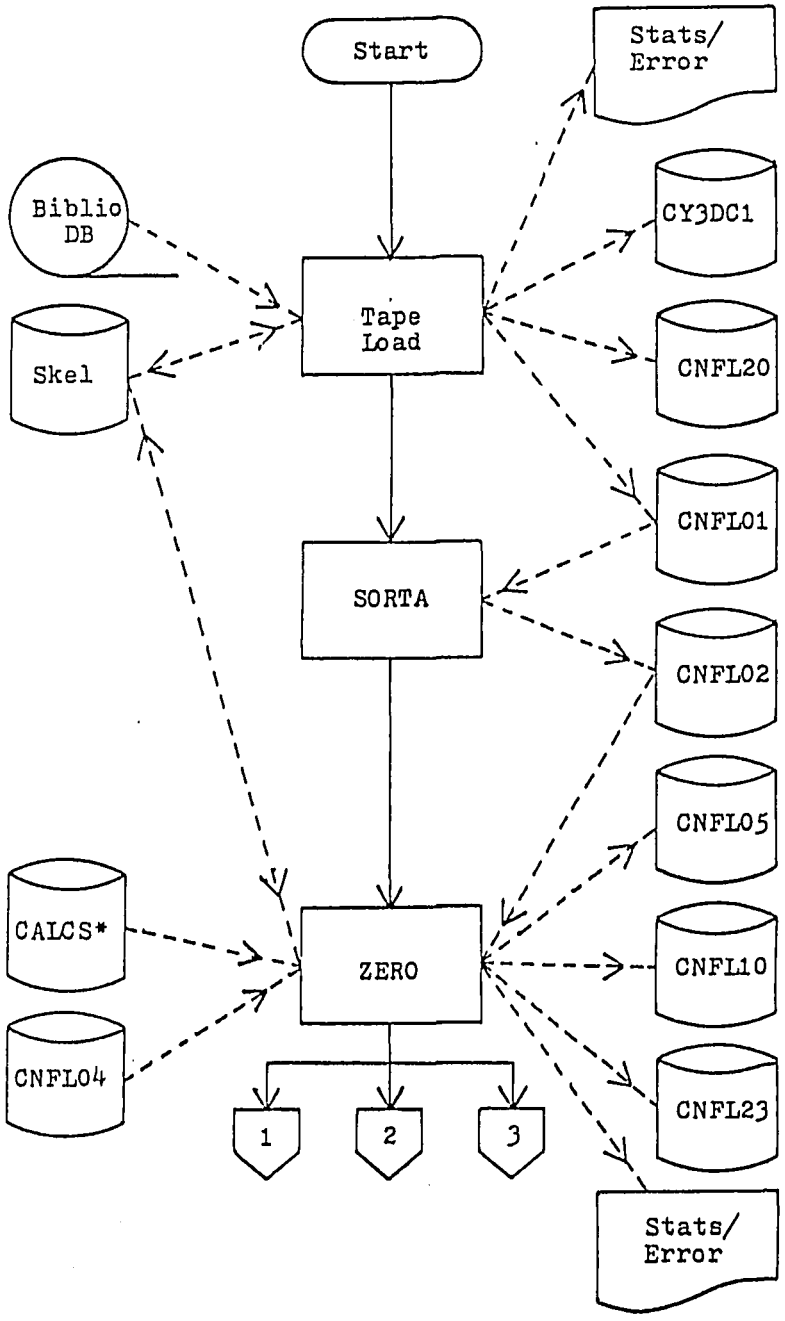
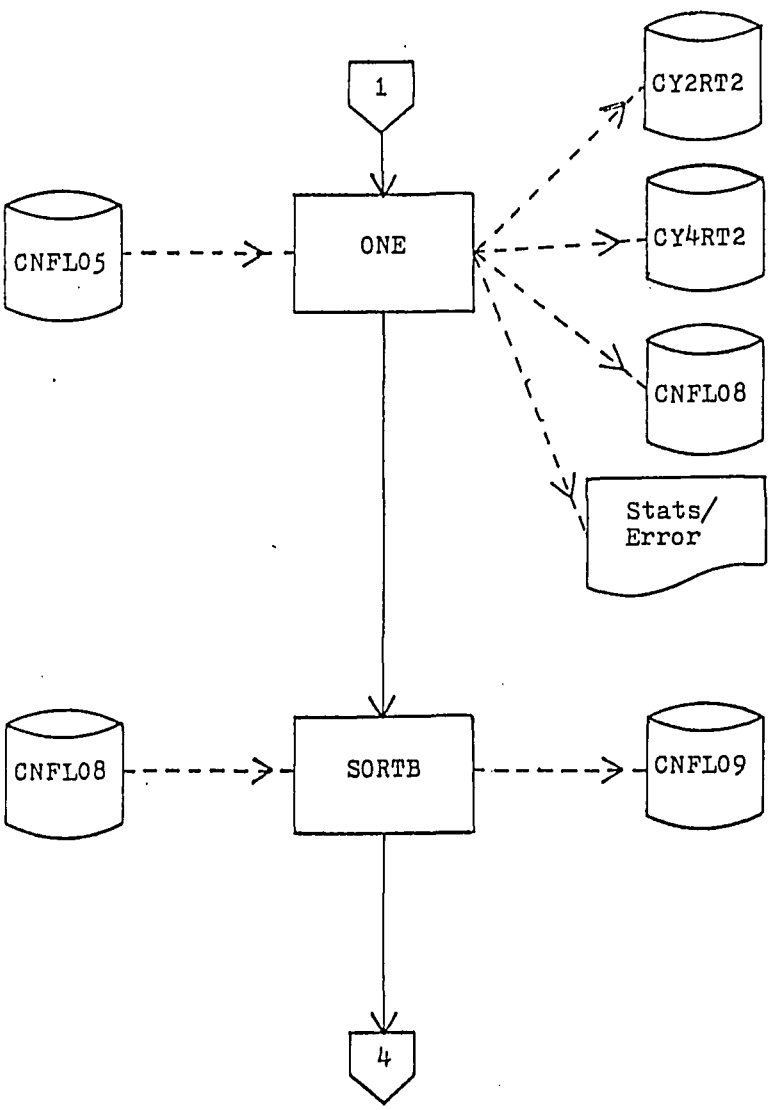


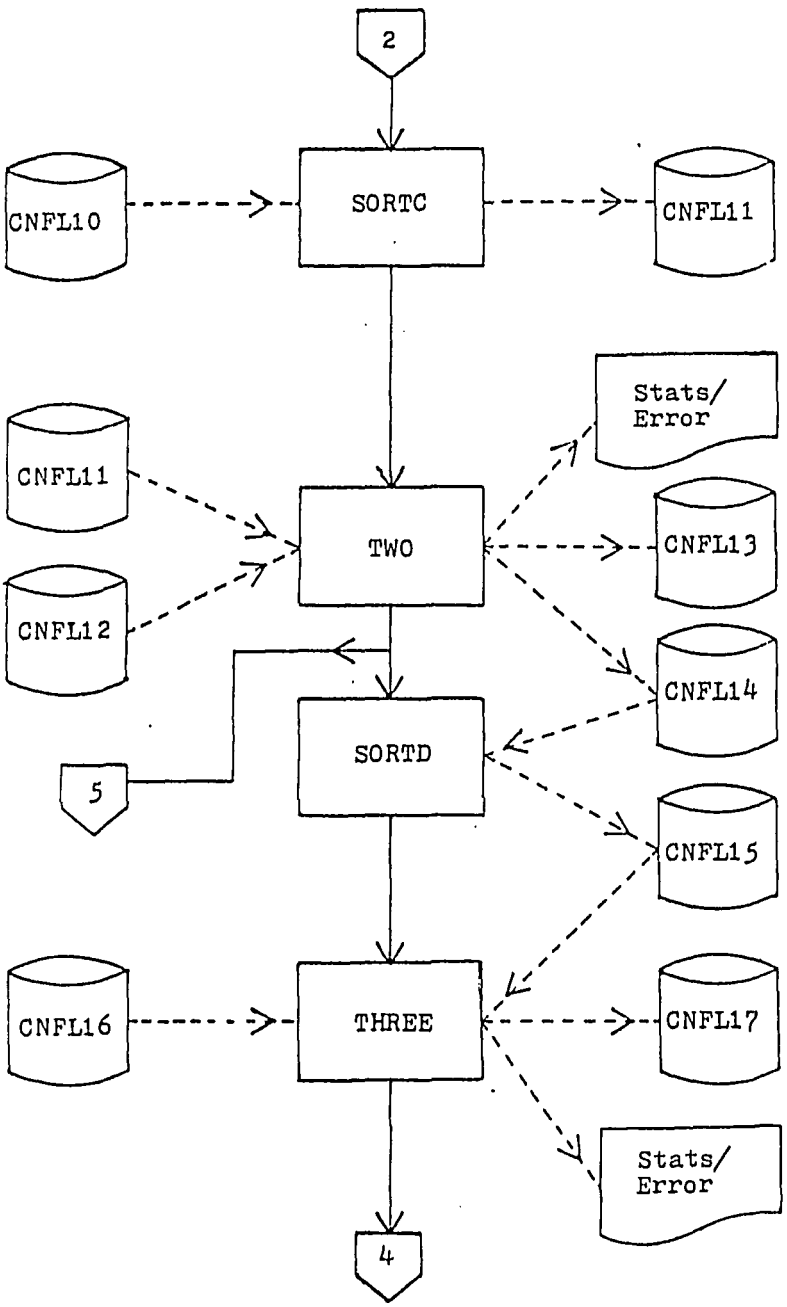

Figure 7-11: EIGHT

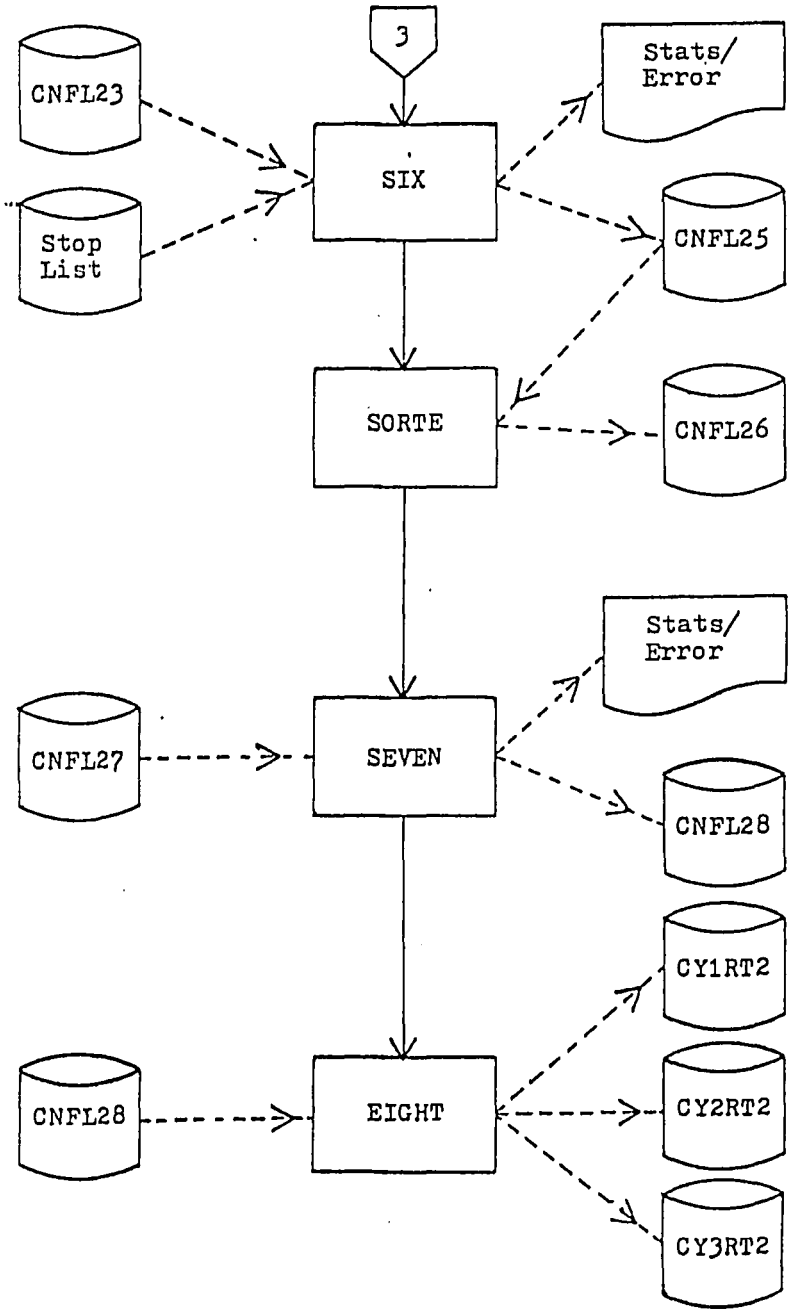
II. Connectivity System Flow Chart

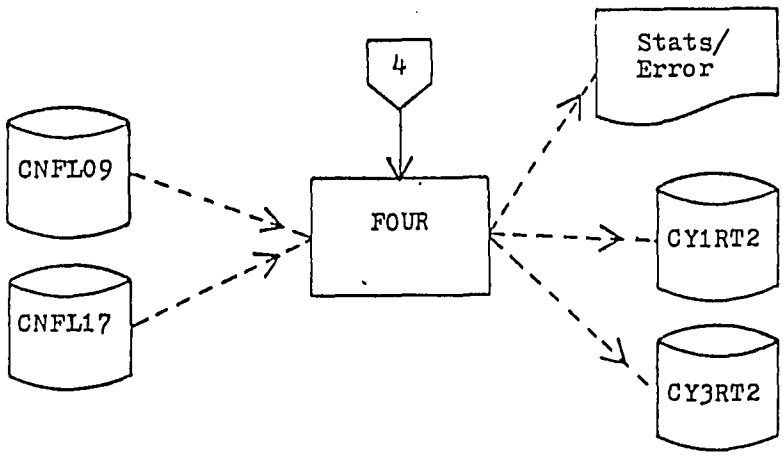
Figure 8-1: System Flow Chart

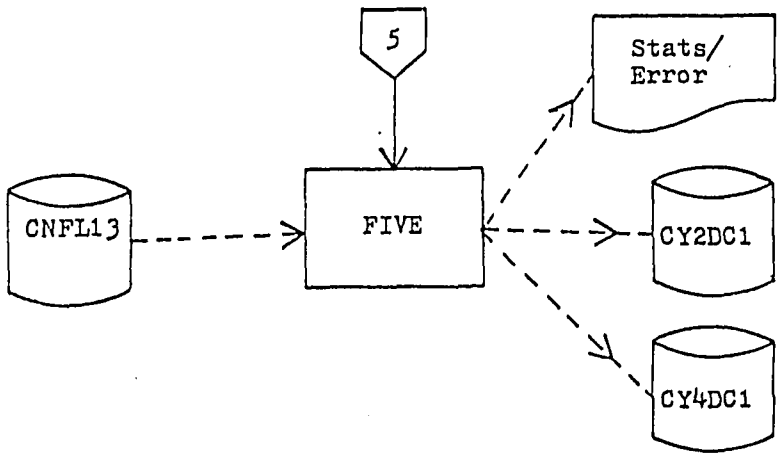



1
CY2RT2
CNFL05
ONE
CY4RT2
CNFL08
Stats/
Error
CNFL08
SORTB
CNFL09
4


2
CNFL10
SORTC
CNFL11
Stats/
Error
CNFL11
TWO
CNFL13
CNFL12
CNFL14
SORTD
5
CNFL15
CNFL16
THREE
CNFL17
Stats/
Error
4


3
CNFL23
Stats/
Error
SIX
Stop
List
CNFL25
SORTE
CNFL26
Stats/
Error
CNFL27
SEVEN
CNFL28
CY1RT2
CNFL28
EIGHT
CY2RT2
CY3RT2


4
Stats/
Error
CNFL09
CNFL17
FOUR
CY1RT2
CY3RT2


5
Stats/
Error
CNFL13
FIVE
CY2DC1
CY4DC1

## III. Data Element Descriptions

### III.1 Connectivity Data Elements

NAME: Aff-Weight

COBOL DATA TYPE: 9

DEFINITION: Weight at which a document and a phrase are affiliated

LIBRARIES: DOCPHR, NPMSRI, NPMSRO, NPTRSI, NPTRSO

PROGRAMS: SORTA, SORTC, EILOAD, ZERO, ONE, TWO, FIVE

RETRIEVAL ITEMS: DS-AFFN-WEIGHT, AF-AFFN-WEIGHT

NAME: Associated-Number

COBOL DATA TYPE: 9(5)

DEFINITION: Phrase which is associated with the current phrase being looked up. or processed.

LIBRARIES: ASPHR1, ASPHR2

PROGRAMS: TWO, THREE, FOUR, SORTD

RETRIEVAL ITEMS: AS-PHRASE-NUMBER, AF-PHRASE-NUMBER, PR-PHR-NUMBER

NAME: Association-Weight

COBOL DATA TYPE: 99

DEFINITION: Represents the weight assigned to each pair of associated phrases.

LIBRARIES: ASPHR1

PROGRAMS: SORTD, TWO, THREE

RETRIEVAL ITEMS: AS-ASSN-WEIGHT

NAME: Calc-Code

COBOL DATA TYPE: X(4)

DEFINITION: Identifier for each Card-A-Lert code phrase

PROGRAMS: ZERO

NAME: Calc-Length

COBOL DATA TYPE: 999

DEFINITION: Length of a Card-A-Lert Phrase

PROGRAMS: ZERO

NAME: Calc-Actual-Phrase

COBOL DATA TYPE: X(150)

DEFINITION: Text of a Card-A-Lert Phrase

PROGRAMS: ZERO

NAME: Dir-Abstract

COBOL DATA TYPE: 9(4)

DEFINITION: Length of the abstract field within the current document.

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DIR-ABSTRACT

NAME: Dir-Affiliation

COBOL DATA TYPE: 9(4)

DEFINITION: Length of the Author Affiliation field within the current document.

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DIR-AFFILIATION

NAME: Dir-Author

COBOL DATA TYPE: 9(4)

DEFINITION: Length of the Author field within the current document.

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DIR-AUTHOR

NAME: Dir-Citation

COBOL DATA TYPE: 9(4)

DEFINITION: Length of the Citation field within the current document.

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DIR-CITATION

NAME: Dir-Coden

COBOL DATA TYPE: 9(4)

DEFINITION: Length of the CODEN field within the current document.

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DIR-CODEN

NAME: Dir-Title

COBOL DATA TYPE: 9(4)

DEFINITION: Length of the Title field within the current document.

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DIR-TITLE

NAME: Doc-Source

COBOL DATA TYPE: XX

DEFINITION: Supplier of the document, e.g. EI = Engineering Index, CA = Chemical Abstracts, etc...

LIBRARIES: DOCKEY

PROGRAMS: EILOAD, FIVE

RETRIEVAL ITEMS: DK-DOC-SOURCE

NAME: Document-Location

COBOL DATA TYPE: 9(5)

DEFINITION: Key to stored affiliated document numbers

LIBRARIES: PHRKEY

PROGRAMS: SORTB, ONE, FOUR

RETRIEVAL ITEMS: PK-DOCUMENT-LOCATION

NAME: Document-Number

COBOL DATA TYPE: 9(5)

DEFINITION: Each document is assigned a unique document number for identification of that document within all files.

LIBRARIES: DOCKEY, DOCPHR, NPMSRI, NPMSRO, NPTRSI, NPTRSO

PROGRAMS: SORTA, SORTC, EILOAD, ZERO, ONE, TWO, FIVE

RETRIEVAL ITEMS: DS-DOCUMENT-NUMBER

NAME: Integer-Weight

COBOL DATA TYPE: 9(5)

DEFINITION: Weight assigned to two phrases which are associated more than once.

LIBRARIES: ASPHR2

PROGRAMS: THREE, FOUR

NAME: Last-Number-Assigned

COBOL DATA TYPE: 9(5)

DEFINITION: Last phrase/document number assigned in the previous connectivity run.

PROGRAMS: ZERO

NAME: Number-Affiliated

COBOL DATA TYPE: 999

DEFINITION: The number of documents affiliated with the current phrase

LIBRARIES: PHRKEY

PROGRAMS: SORTB, ONE, FOUR

RETRIEVAL ITEMS: PK-NUMBER-AFFILIATED

NAME: Number-Of-Words

COBOL DATA TYPE: 99

DEFINITION: Indicates the number of stems in the current phrase. Must be less than 16.

LIBRARIES: STEMRD

PROGRAMS: SIX, SEVEN, EIGHT, SORTE

RETRIEVAL ITEMS: PR-NUM-WORDS

NAME: Page-Location

COBOL DATA TYPE: 999

DEFINITION: The location of the current phrase/document within the page of text which has been retrieved from the phrase/document text file.

LIBRARIES: DOCKEY, PHRKEY

PROGRAMS: SORTB, EILOAD, ONE, FOUR, FIVE

RETRIEVAL ITEMS: DK-PAGE-LOCATION, PK-PAGE-LOCATION

NAME: Page-Number

COBOL DATA TYPE: 9(5)

DEFINITION: The page number in which the current phrase/document text exists in the phrase/document text file.

LIBRARIES: DOCKEY, PHRKEY

PROGRAMS: SORTB, EILOAD, ONE, FOUR, FIVE

RETRIEVAL ITEMS: DK-PAGE-NUMBER, PK-PAGE-NUMBER

NAME: Phrase-Length

COBOL DATA TYPE: 999

DEFINITION: Length of the current phrase in characters.

LIBRARIES: NUPHRI, NUPHRO, NPMSRI, NPMSRO, NPTRSI, NPTRSO, PHRKEY

PROGRAMS: SORTA, SORTB, EILOAD, ZERO, ONE, FOUR, SIX

RETRIEVAL ITEMS: PK-PHRASE-LENGTH

NAME: Phrase-Number

COBOL DATA TYPE: 9(5)

DEFINITION: Each phrase is assigned unique phrase number for identification of that phrase within the files.

LIBRARIES: ASPHR1, ASPHR2, DOCPHR, NUPHRI, NUPHRO, NPMSRI, NPMSRO, PHRKEY, STEMRD

PROGRAMS: SORTB, SORTC, SORTD, SORTE, ZERO, ONE, TWO, THREE , FOUR, FIVE, SIX, SEVEN, EIGHT

RETRIEVAL ITEMS: AS-PHRASE-NUMBER, AF-PHRASE-NUMBER, PR-PHR-NUMBER

NAME: Phrase-Text (for INPUT files)

COBOL DATA TYPE: X(150)

DEFINITION: Text of the current phrase. Must be less than 151 characters.

LIBRARIES: NUPHRI, NPMSRI, NPTRSI

PROGRAMS: SORTA, EILOAD, ZERO, ONE, SIX

NAME: Phrase-Text (for OUTPUT files)

COBOL DATA TYPE: X OCCURS 1 TO 150 TIMES DEPENDING ON length-var

DEFINTIION :Text of the current phrase. Must be less than 151 characters.

LIBRARIES: NUPHRO, NPMSRO, NPTRSO

PROGRAMS: SORTA, EILOAD, ZERO, ONE, SIX

NAME: Real-Weight

COBOL DATA TYPE: V99

DEFINITION: The actual weight of an association between two phrases.

LIBRARIES: ASPHR2

PROGRAMS: THREE, FOUR

RETRIEVAL ITEMS: AS-ASSN-WEIGHT

NAME: Stem

COBOL DATA TYPE: X15

DEFINITION: A component of a phrase which is greater than 2 but less than 16 chracters in length, does not appear in the STOP LIST, and does not end in 'S'.

LIBRARIES: STEMRD

PROGRAMS: SIX, SEVEN, EIGHT, SORTE

RETRIEVAL ITEMS: IN-STEM, KE-STEM

NAME: Stem-Position

COBOL DATA TYPE: 99

DEFINITION: Position of a stem within a phrase where the rightmost stem is in position one.

LIBRARIES: STEMRD

PROGRAMS: SIX, SEVEN, EIGHT, SORTE

RETRIEVAL ITEMS: PR-STEM-POSN

III.2 Retrieval Data Elements

| Data Element Name | Library | COBOL Data Type |
|---|---|---|
| AF-AFFN-WEIGHT | PHRSTR | 9 |
| AF-PHRASE-NUMBER | PHRSTR | 9(5) |
| AS-ASSN-WEIGHT | ASNSTR | V99 |
| AS-PHRASE-NUMBER | ASNSTR | 9(5) |
| DK-AFF-LOCATION | DOCMNT | 9(5) |
| DK-DIR-ABSTRACT | DOCMNT | 9(4) |
| DK-DIR-AFFILIATION | DOCMNT | 9(4) |
| DK-DIR-AUTHOR | DOCMNT | 9(4) |
| DK-DIR-CITATION | DOCMNT | 9(4) |
| DK-DIR-CODEN | DOCMNT | 9(4) |
| DK-DIR-TITLE | DOCMNT | 9(4) |
| DK-NUMBER-AFFILIATED | DOCMNT | 99 |
| DK-PAGE-LOCATION | DOCMNT | 999 |
| DK-PAGE-NUMBER | DOCMNT | 9(5) |
| DK-SOURCE | DOCMNT | XX |
| DOC-TEXT | DOCTXT | X 512 TIMES |
| DS-AFF-WEIGHT | DOCSTR | 9 |
| DS-DOCUMENT-NUMBER | DOCSTR | 9(5) |
| IN-INDEX | STMIDX | 9(5) |
| IN-NUMBER-RECS | STMIDX | 99 |
| IN-STEM | STMIDX | X(15) |
| KE-NUMBER-RECS | STMKEY | 9(5) |
| KE-STEM | STMKEY | X(15) |
| KE-STEM-PTR | STMKEY | 9(5) |
| PHR-TEXT | PHRTXT | X 512 TIMES |
| PK-ASSOCIATION-LOCATION | PHRASE | 9(5) |
| PK-DOCUMENT-LOCATION | PHRASE | 9(5) |
| PK-NUMBER-AFFILIATED | PHRASE | 999 |
| PK-NUMBER-ASSOCIATED | PHRASE | 999 |
| PK-PAGE-LOCATION | PHRASE | 999 |
| PK-PAGE-NUMBER | PHRASE | 9(5) |
| PK-PHRASE-LENGTH | PHRASE | 999 |
| PR-NUM-WORDS | STMPRO | 99 |
| PR-PHR-NUMBER | STMPRO | 9(5) |
| PR-STEM-POSN | STMPRO | 99 |

## IV. File Descriptions

Note that file names are given as DECSYSTEM-20 directory entries which can be cross-referenced to the file names within the programs in the program descriptions. Primary keys are shown in boldface; secondary keys are underlined.

Connectivity Files

SKEL(LAST-NUMBER-ASSIGNED)

CALCS(CALC-CODE, CALC-LENGTH, ACTUAL-CALC-PHRASE)

CNFLO1(DOCUMENT-NUMBER, PHRASE-LENGTH, AFF-WEIGHT, ACTUAL-PHRASE-TEXT)

CNFLO2(DOCUMENT-NUMBER, PHRASE-LENGTH, AFF-WEIGHT, ACTUAL-PHRASE-TEXT)

CNFLO4(PHRASE-NUMBER, DOCUMENT-NUMBER, PHRASE-LENGTH, AFF-WEIGHT, ACTUAL-PHRASE-TEXT)

CNFLO5(PHRASE-NUMBER, DOCUMENT-NUMBER, PHRASE-LENGTH, AFF-WEIGHT, ACTUAL-PHRASE-TEXT)

CNFLO8(PHRASE-NUMBER, PAGE-NUMBER, PAGE-LOCATION, PHRASE-LENGTH, DOCUMENT-LOCATION, NUMBER-AFFILIATED)

CNFL09(PHRASE-NUMBER, PAGE-NUMBER, PAGE-LOCATION, PHRASE-LENGTH, DOCUMENT-LOCATION, NUMBER-AFFILIATED)

CNFL10(DOCUMENT-NUMBER, PHRASE-NUMBER, AFF-WEIGHT)

CNFL11(DOCUMENT-NUMBER, PHRASE-NUMBER, AFF-WEIGHT)

CNFL12(DOCUMENT-NUMBER, PHRASE-NUMBER, AFF-WEIGHT)

CNFL13(DOCUMENT-NUMBER, PHRASE-NUMBER, AFF-WEIGHT)

CNFL14(PHRASE-NUMBER, ASSOCIATED-NUMBER, ASSOCIATION-WEIGHT)

CNFL15(PHRASE-NUMBER, ASSOCIATED-NUMBER, ASSOCIATION-WEIGHT)

CNFL16(PHRASE-NUMBER, ASSOCIATED-NUMBER, INTEGER-WEIGHT, REAL-WEIGHT)

CNFL17(PHRASE-NUMBER, ASSOCIATED-NUMBER, INTEGER-WEIGHT, REAL-WEIGHT)

CNFL20(DOCUMENT-NUMBER, PAGE-NUMBER, PAGE-LOCATION, DOC-SOURCE, DK-DIRECTORY(DIR-TITLE, DIR-AUTHOR, DIR-CITATION, DIR-CODEN, DIR-AFFILIATION, DIR-ABSTRACT))

CNFL23(PHRASE-NUMBER, PHRASE-LENGTH, ACTUAL-PHRASE-TEXT)

CNFL24(STOP-WORD)

CNFL25(STEM, PHRASE-NUMBER, STEM-POSITION, NUMBER-OF-WORDS)

CNFL26(STEM, PHRASE-NUMBER, STEM-POSITION, NUMBER-OF-WORDS)

CNFL27(STEM, PHRASE-NUMBER, STEM-POSITION, NUMBER-OF-WORDS)

CNFL28(STEM, PHRASE-NUMBER, STEM-POSITION, NUMBER-OF-WORDS)

**Retrieval Files**

Note that a data element in "[ ]" represents a key in referential position.

CY2DC1([DOCUMENT-NUMBER], DK-AFF-LOCATION, DK-NUMBER-AFFILIATED, DK-PAGE-NUMBER, DK-PAGE-LOCATION, DK-SOURCE, DK-DOC-DIRECTORY(DK-DIR-TITLE, DK-DIR-AUTHOR, DK-DIR-CITATION, DK-DIR-CODEN, DK-DIR-AFFILIATION, DK-DIR-ABSTRACT))

CY5DC1([PAGE-NUMBER], DOC-TEXT)

CY4DC1(DS-DOCUMENT-NUMBER, DS-AFF-WEIGHT)

CY1RT2([PHRASE-NUMBER], PK-PAGE-NUMBER, PK-PAGE-LOCATION, PK-PHRASE-LENGTH, PK-DOCUMENT-LOCATION, PK-NUMBER-AFFILIATED, PK-ASSOCIATION-LOCATION, PK-NUMBER-ASSOCIATED)

```
CY2RT2([PAGE-NUMBER], PHR-TEXT)

CY3RT2(AS-PHRASE-NUMBER, AS-ASSN-WEIGHT)

CY4RT2(DS-DOCUMENT-NUMBER, DS-AFF-WEIGHT)

CY2RT1(IN-STEM, IN-INDEX, IN-NUMBER-RECS)

CY3RT1(KE-STEM, KE-STEM-PTR, KE-NUMBER-RECS)

CY4RT1(PR-PHR-NUMBER, PR-STEM-POSN, PR-NUM-WORDS)
```

## Vita

Steven Alan Russell, born June 5, 1960, is the son of David and Judith Russell. After graduating from White Plains High School in June 1978, he was admitted to Lehigh University. As an undergraduate at Lehigh, Mr. Russell graduated with honors in June 1982, receiving a B. S. degree in Computing and Information Science. He remained at Lehigh the following year to receive his M. S. degree in June 1983. While at Lehigh, Mr. Russell was employed as a Research Assistant at the Center for Information and Computer Science where he was involved in the rejuvenation of the LEADER Document Retrieval System.