Use of an Inverted File Structure For Interactive Retrieval of Computer Program Abstracts

A. Camille Buschman,
Standard Oil Company of California,
San Francisco

Abstract

Standard Oil Company of California's Program Abstracts System will interactively search for, and list by title and mnemonic, all Company computer programs belonging to a particular category or technical application area in the field of petroleum exploration and production. If one or more titles are of interest, the user can request a display of those abstracts. Retrieval is by search of an inverted index made up of a fixed set of terms covering application areas, programming languages, processing mode, and machine designations.

Introduction

Standard Oil Company of California has many geographically-scattered exploration and producing offices. Technical people in those offices frequently need computer application programs, and they need a means to determine which programs are available, and how to obtain them. The Program Abstracts System was designed to meet these needs. It is a directory of programs, providing a brief abstract of function, what kind of machine it will run on, source language, and where (and from whom) to get more detailed information.

This information is obtained by on-line interrogation of a computer file. The Program Abstracts System resides on an IBM System/360-67 in San Francisco, and operates under an in-house installation of a time-sharing service of Conversational Software System, Inc. The Program Abstracts System may be accessed from any of our offices in the U.S. and Canada. New information can be inserted and changes in records can be made quickly, thus by-passing the tedious distribution of never-up-to-date hard copy indexes.

Data Description: Relational Levels

Inputs

Inputs to the System consist of fixed-length EBCDIC-coded records. Each record consists of the following types of information for each computer program entry: mnemonic, title, programming language(s), machine(s) on which it will run, whether batch mode or time-shared mode (or both), sources of further information (documentation, source program, and "contact individual"), a short abstract, and descriptor(s) of the program application area.

This information can be recorded either interactively using a terminal, or on an input form, from which cards are punched and entered into the system in batch mode. Batch mode is the method generally used, except for minor corrections made centrally by the person responsible for system maintenance.

Input forms were designed--like the whole system--to be as simple as possible for both the collection and retrieval of information. This was done in order to encourage wide use, not only by data processing personnel, but also by the geologist or engineer looking for a computer program to help him in his work. For this reason, some sophistication was sacrificed, and descriptors for program application areas were kept few and general. These application areas include terms for some general mathematical and statistical techniques, for instance, but for the most part are terms in the field of petroleum exploration and production technology.

The indexing--filling out input forms-- is done throughout our Company by people of varying technical levels and diverse backgrounds. Distributed with the forms they use are a set of instructions and definitions of applications areas, giving the limits and scope of the descriptors. These definitions are also used to aid the user in the choice of descriptors when retrieving information. The information on the input forms is indicated by circling the appropriate descriptor rather than by writing it out. This is easier and less subject to typographical error by indexer and keypuncher. Additionally, each descriptor entry is checked by the update program of the System against a dictionary of allowable terms.

Outputs

There are two types of retrieval output: "descriptor" and "mnemonic."

Descriptor Retrieval: This results in a display of program titles and related mnemonics, from which list the user can request the full abstract to be displayed.

Possible search descriptors include terms from the lists of application areas, the source language, the machine, and processing mode (viz., "batch"/"time-shared".) If several descriptors are used, they are implicitly connected by the logical operator "AND."

Figure 1 (below) shows an example of a time-shared retrieval session with our System.

Mnemonic Retrieval: If the user already knows the name of a program in which he is interested, he can request that the abstract be printed without a search.

```
EXECUTION:
DO  YOU WISH TO SEARCH (1) OR PRINT (2)
2
MNEMONIC
quic

.....................
MNEMONIC-QUIC        /  /71    BATCH

-KEY WORD IN CONTEXT INDEXING

ABSTRACT-
 READS "TEXT" (TITLES, AUTHOR NAMES, ABSTRACTS, ETC) FROM CARDS
 AND PRODUCES PERMUTED LINES OF TEXT WITH EACH KEYWORD AS THE CENTER
 WORD OF THE LINE.  OVER 20 OPTIONS INCLUDE SPECIFYING WORDS NOT TO
 BE INDEXED; THE ONLY WORDS TO BE INDEXED; WHICH CARD TYPES
 TO INDEX; WHICH CARD TYPES TO PRINT IN BIBLIOGRAPHY; WHICH
 CHARACTERS INDICATE BEGINNING & ENDING OF WORDS (BLANK,
```

Figure 1 (continued on page 6 following)

```
 HYPHEN, ETC).  UP TO 256 CARD TYPES ALLOWED

CONTACT          - T.A. REICHARDT, CSD, SAN FRANCISCO

SOURCE PROGRAM   - CSD, SAN FRANCISCO
DOCUMENTATION AT - CSD, SAN FRANCISCO
MACHINE(S)       - 360/OS
LANGUEAGE(S)     - PL/1
DESCRIPTOR(S)-
 FILE MGMT

.........................
SEARCH  (1), PRINT (2), OR END (3)  ?
1
GIVE FIRST DESCRIPTOR
basis
SECOND DESCRIPTOR

112 HITS. LIST (1), SEARCH AGAIN (2), OR END (3)?
2
GIVE FIRST DESCRIPTOR
assembly
SECOND DESCRIPTOR
t/s
THIRD DESCRIPTOR

SORRY, NO HITS
GIVE FIRST DESCRIPTOR
stat-bivariate
SECOND DESCRIPTOR
comshare
THIRD DESCRIPTOR

  6 HIT(s) -

MNEMONIC  TITLE
--------  -----
CONLEAS   CONSTRAINED LEAST SQUARES CURVE FIT
CURFIT6   LEAST SQUARES CURVE FIT - SIX CURVES
GENSTAT   GENERAL PURPOSE STATISTICAL ROUTINE
ORTHOFIT  CURVE FITTING - ORTHOGONAL POLYNOMIAL METHOD
PLOTCURV  PLOT LEAST SQUARES FIT, X ON Y OR Y ON X
```

Figure 1 (continued on page 7 following)

```
TRIGFIT   CURVE FITTING WITH TRIGONOMETRIC FUNCTIONS

SEARCH (1), PRINT (2), OR END (3) ?
2
MNEMONIC
curfit6

.............................
MNEMONIC-CURFIT6     01/01/71       T/S

-LEAST SQUARES CURVE FIT - SIX CURVES

ABSTRACT-
 PROVIDES LEAST SQUARES FITS FOR SIX CURVE TYPES AT ONCE:
 LINEAR, POWER, EXPONENTIAL, AND THREE TYPES OF
 HYPERBOLA.  USER SPECIFIES NUMBER OF VALUES (N) GIVEN
 AS DATA FOR TWO INPUT VARIABLES X AND Y.  PROGRAM
 LISTS CURVE TYPE (EG Y=A+B*X), INDEX OF DETERMINATION,
 AND A AND B.  USER CAN REQUEST FURTHER DETAILS ON
 ANY OF THE CURVES.

CONTACT              - J. SMITH, CSD, SAN FRANCISCO

SOURCE PROGRAM       - E&P, CSD, SAN FRANCISCO
DOCUMENTATION AT     - E&P, CSD, SAN FRANCISCO
MACHINE(S)           - GE T/S
                       COMSHARE
LANGUAGE(S)          - FORTRAN
DESCRIPTOR(S)-
 STAT-BIVARIATE

.......................
SEARCH (1), PRINT(2), OR END (3)  ?
3

END OF PRINT

11.36.26
```

Figure 1 (concluded)

(...Inverted File ... Interactive Retrieval ...,continued)

Data Description: Logical/Physical Levels

In a real-time environment fast retrieval is assumed to be essential. Because retrieval questions were expected to be fairly limited in type and scope and because file volatility was expected to be low, a completely inverted index seemed the most practical solution. With this design, all searching can be done "in" the indexes. When answers (record identifiers) are found, records can be accessed directly and displayed to users (2).

> (An inverted file is one whose design allows logical organization to be kept separate from physical organization. Here, each record of the master file is an abstract, including as part of the record certain attributes or descriptors. The inverted file contains a record for each attribute/descriptor, and includes as part of the record a list of all the master records describable by that attribute. In library terms, the books on shelves are master records, and the subject catalogue is the inverted file.)

Logically, or conceptually, two inverted indexes (or "directories") are used--one for searching by descriptor, and one for searching by mnemonic. Physically, four files are used.

The relationship between these four files and the master file are best summarized in Figure 2, which follows.

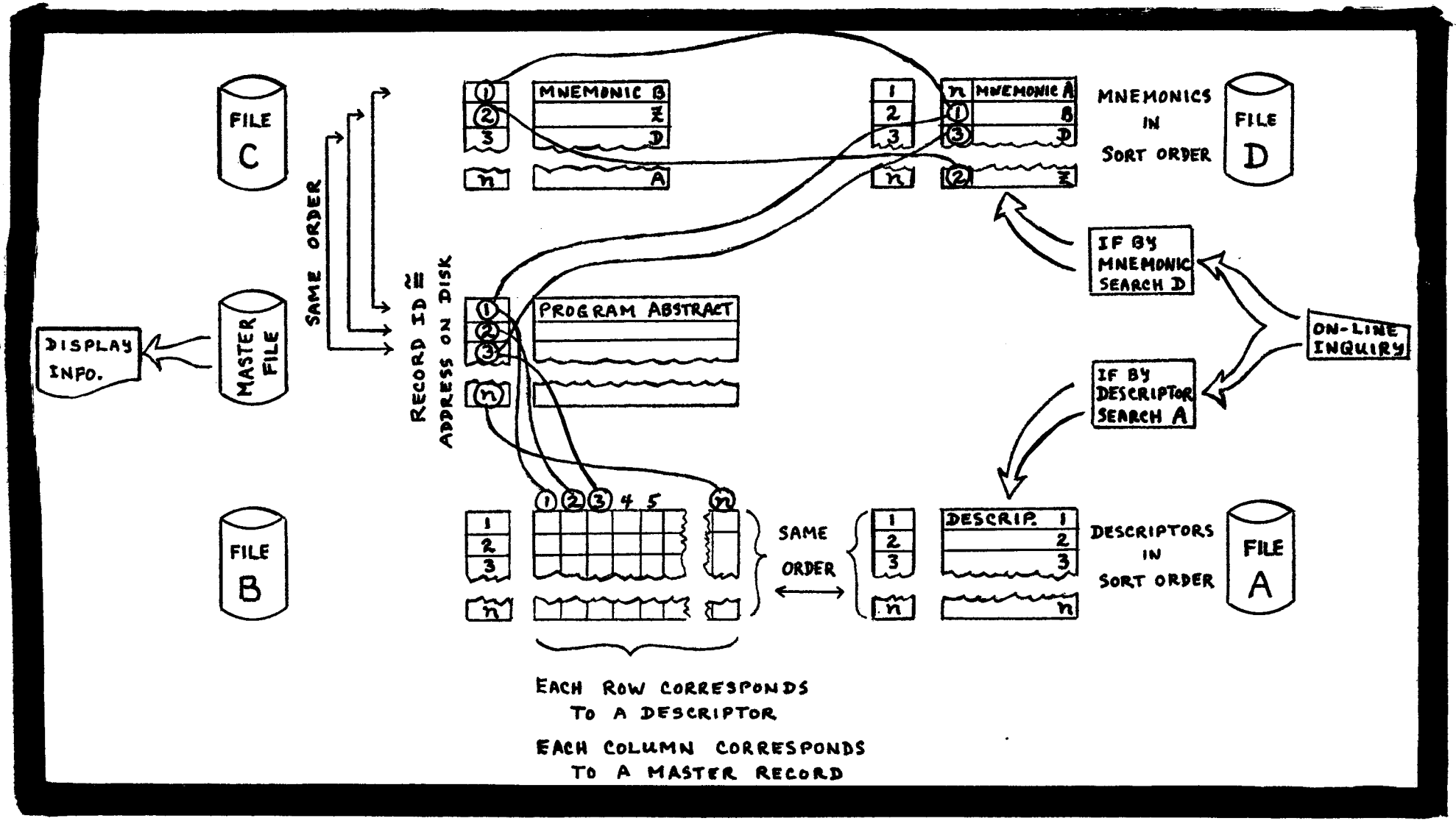

Figure 2: File Relationships and Retrieval

In the case of the Descriptor Directory ("File B") each descriptor has a list of ID's of programs having that characteristic. However, the descriptor itself is not written as part of the file. Instead, its meaning is implicit from the descriptor's relative position.

To find the correct position **(or "row," or "record")** in which to look, an immediate search is made to find the given descriptor in the list in File A. Its relative position in File B is then known, and exactly.

Searching by mnemonic is simple. A list--called "File D"--is searched and a one-to-one-correspondence dictionary gives the appropriate record ID in the master file.

To speed the process, a binary search is used in all cases. This necessitates ordered files, and requires sorting of File C to recreate File D with each update ⟨1, 3, and 4⟩.

However, as mentioned earlier, file volatility is low, so that the cost of occasional sorting is outweighed by the added speed of retrieval in terms of user satisfaction.

When searches by descriptors involve more than one descriptor, separate directory lists are obtained. The records which will meet the search criteria are those that are common to all the lists. The method chosen for storing these lists was governed by several factors. The first approach was to use variable-length (unsorted) lists, in an attempt to save storage space. Each list--one for each descriptor--was a record; each word was a master-record ID.

However, the length of the lists was extremely variable. Because these lists were unsorted, the process of AND-ing lists had to compare each item of one list with each item of another until an equal compare was found, or the end of the list was reached. Another sort during update was undesirable, yet without such a sort, AND-ing of the lists took too much time. Therefore, a decision was made to use the much simpler form of a "Peek-a-boo Array," in which a master record is represented by a position, and the presence or absence of every attribute is indicated by turning an indicator "on" or "off" ⟨5⟩.

> ⟨The term "Peek-a-boo" derives from an older, manual system, in which each card represents some concept or attribute. Cards in the files of such older systems have dedicated spaces for ID numbers

of documents. If a space on the the card is punched, the attribute applies to that document. For retrieval, numerous cards can be superimposed. Documents which possess all the attributes are the ones where all the cards have spaces--one can see through the card deck.)

This form has proved to be ideal for AND-ing; it is easy to maintain, and its use cut retrieval time by a very large percentage.

Size of the Data Base

The Program Abstracts System can currently handle 500 abstracts. However, a much larger data base can be built and accessed, with only modest changes in our source programs. (See core requirements below.)

Programming Considerations

The system is not a general retrieval system, but a special application written in FORTRAN. It is therefore limited in flexibility, but is quite efficient in operation. A retrieval question using three descriptors typically requires .25 to 1.00 cpu seconds. (This excludes the overhead of time needed to "sign on.") The lower figure represents the search time, regardless of how many "answers" are found. The cpu time tnen increases with the number of I/O's required to display the answers. The only variable of which the user is aware is the time required to print or otherwise display the answers.

The retrieval routines--searching and printing--occupy less than 50K bytes. The updating routines are divided into four separate procedures--receiving and checking input, updating two mnemonic files, updating the directory list, and updating the master file. Each procedure is run as a separate job step, and each occupies considerably less than 50k bytes.

The system uses a FORTRAN source-coded, in-core sorting subroutine.

The total number of source statements is approximately 1,500 (for all job-steps.)

Acknowledgement

I wish to express my thanks to Standard Oil Company of California for permission to publish this information, and to Mr. T. A. Reichardt for his many helpful comments.

Bibliography

1. Davis, Charles H., "The Binary Search Algorithm," American Documentation 20, No. 2 (April, 1969), p. 167.

2. Dodd, George C., "Elements of Data Management Systems," Computing Surveys 1, No. 2 (June, 1969), pp. 117 - 133.

3. Lowe, Thomas C. and Roberts, David C., "Analysis of Directory Searching," Journal of the A.S.I.S. 2, No. 3 (May - June, 1972), pp. 143 - 149.

4. Price, C. E. "Table Lookup Techniques," Computing Surveys 3, No. 2 (June, 1971), pp. 49 - 65.

5. Spitzer, John H., "Storing the Directory for an Inverted List System," Data Base 1, No. 4 (Winter, 1969), pp. 12 - 14.