Reprinted from

INFORMATION STORAGE AND RETRIEVAL

PERGAMON PRESS

OXFORD · LONDON · NEW YORK · PARIS

Inform. Stor. Retr. Vol. 6, pp. 387–399. Pergamon Press 1970. Printed in Great Britain

DIRAC: AN INTERACTIVE RETRIEVAL LANGUAGE WITH COMPUTATIONAL INTERFACE

J. F. VALLEE*

Manager, Information Systems,
Computation Center, Stanford University, Stanford, California 94305

Summary—An interactive file-oriented language that allows the user to interface with a text-editor and with his own FORTRAN or assembly language code has been used to illustrate the flexibility of non-procedural techniques in the scientific field. The language is the first in a family of prototypes used to test alternative formulations of file organization problems connected with the storage and retrieval of scientific records, medical data or library documents in an interactive mode. The applications described here use files of research data in astronomical and medical fields. It operates exclusively in a time-sharing environment. The article describes the system and its applications from the point of view of language design, and it gives a detailed discussion of the file organization upon which it relies.

WIDESPREAD activity has recently been directed at the implementation of non-procedural languages dedicated to data-base management. Typically, these systems allow their user to specify retrieval, extraction and update actions to be taken on his data, without requiring the intervention of a programmer. Not only are such systems financially attractive, they also offer an opportunity to accelerate the flow of information from its source (such as a market or a cost center) to the level where management decisions can be made most meaningfully [1].

TECHNICAL PROBLEMS

The impact of such languages on the design and utilization patterns of future data-bases is difficult to evaluate, but three interesting facts do stand out when they are replaced within the framework of traditional software: first, in spite of the convenience of their external features (that may include some on-line display capabilities) their design and implementation generally reflect the concepts of second-generation file processing rather than those of the time-sharing, interactive environment. Second, the user finds himself locked inside a set of language commands that may be very sophisticated indeed as long as he deals with basic file-oriented functions, but it is only with great difficulty that he can force information outside the system and into programs expressed in other high-level languages. Third, all language features are aimed at the business user: to our knowledge, no generalized file management system has yet been applied to the solution of a scientific problem; as a result, they do not take full advantage of the insight gained by the designers of scientific systems intended for both documentation and computation.

* This paper is based in part on experiments conducted by the author and on a language he designed prior to his association with Stanford University.

As the level of sophistication of the user community rises, and as the frontier between business and scientific processing becomes less sharply defined, we feel that the three problem areas we have mentioned can be expected to appear prominently among the obstacles facing the developers of new data-base systems. The purpose of this article is to explore these implementation difficulties from a technical point of view, not to propose a universal solution. This can be best achieved by describing a complete language prototype and by reporting on the assets and liabilities of the alternative formulations we have hypothesized for the three points mentioned above.

We shall first briefly describe a modular prototype system that serves as the basis for the current experiments. This description will center on the language design aspects of the system and on its user interface.

1. THE DIRAC LANGUAGE FAMILY

Activities and levels of users

The language used in the current interactive experiments, DIRAC-1, is the first prototype in the family of information-oriented languages we have designed. The objective is to facilitate flexible interaction with large files of scientific data. The language is of the non-procedural type and demands no previous computer experience on the part of the user. It allows creation, updating, bookkeeping and validating operations as well as the querying of data files [4]. These activities take place in conversational mode exclusively. To the more sophisticated user the DIRAC languages offer a simple interface with the Stanford text editor (WYLBUR) and to the systems programmer they make available a straightforward interface with FORTRAN that does not require intermediate storage of the extracted information outside of the direct-access memory.

The name DIRAC (DIRect ACcess) is intended to remind the user of this fact. It also summarizes the five data types handled by the language, respectively: Date, Integer, Real, Alphanumeric, Code.

Four operation modes

The user of DIRAC can apply to any file (that he is authorized to access) any command within one of the four sets grouped under the modes: CREATE, UPDATE, STATUS and QUERY. The first of these modes is a privileged one, but this privilege can be extended to any user by the data-base administrator at the time of file creation: it consists in the definition of a file or a series of interrelated files, according to a terminology to be defined below, in both nomenclature and structure. The result of the CREATE commands is the implementation of a file schema whose information content, for the moment, is nil. This schema can be evoked, however, by the UPDATE commands that will start filling the structured set with information drawn either from the working data set operated on by the text editor, or directly from the user's own terminal. Deletion and replacement commands are also available and a chaining structure that will be described in detail in part 4 is superimposed to the information which is apparent to the user; a number of measures, still triggered by the UPDATE commands, are taken to reduce the storage requirements and to guarantee the privacy of the information as it is validated and stored.

In QUERY mode, the user can obtain information from and about any SELECTed subset of his data files, at any level of the structure. The various commands that allow

selection and extraction are described below, after an overall summary of the data organizations recognized by DIRAC. Finally, the STATUS mode provides the user or the DB Administrator with up-to-date status reports where field identification, description, statistics and validation information are summarized within a standard report form.

Implicit and associative query

To illustrate the differences between the information processing concepts of DIRAC and those of traditional procedural languages, one could draw examples from a number of fields. Assume, for instance, that a certain attribute X of an object is measured by a real number, so that we might want to query the file for all objects having X greater than 13.7: this is naturally possible under any system. At the same time, the digits of this real number might have individual significance (in part designations and in some library or medical codes this situation is encountered). We may then be tempted to write something like:

$$X\ (>\ 13.7\ \text{AND DOES NOT CONTAIN}\ 9.2)$$

The above statement is a valid selection rule in DIRAC. It will exclude the values 19·2, 29.2, etc. from the list of X values that exceed 13.7.

The ability to specify implicitly the accessing of deep levels of the file structure, and to continue the query associatively, is also present in DIRAC-1. For instance, consider the following information stored in a list of file values called "Address" in a customer file:

Customer 1	Customer 2	Customer 3
1302 La Plata Ave New Brunswick Kansas	205 E 32 street Princeton New Jersey	13 Mission Blvd. Paris Illinois

Then the following DIRAC selection rules will be applicable:

Address(ANY) CONTAINS New —will select 1 and 2
Address(ALL) CONTAINS is —will select 3
Address(LAST) CONTAINS New—will select 2

We could then follow such a statement with a rule of the type:

$$\text{Transaction(ASSOCIATED)} = XYZ$$

The condition would then be applied only to those entries situated at the same level in the information tree of the "Transaction" list.

To enhance the string scanning capabilities of DIRAC, the character (!) is used as a wild symbol. Thus the statement

Address(2) CONTAINS "r!n"—will select 1 (run in Brunswick)
and 2 (rin in Princeton)

These features, combined with the interpretative nature of the system, serve to give the terminal user a capability for interacting with his data that cannot be achieved in the procedural, batch-processing environment.

2. THE DATA-BASE CONCEPT UNDER DIRAC

Files and records

The concept of file is retained in DIRAC in spite of the fact that its storage structure is never apparent to the user and in spite of the confusion it may create for programmers who tend to relate it to the file concept in procedural languages. It is difficult to propose a more commonly understood term for a collection of related records containing data needed for subsequent processing. Use of the term "Record" in this context raises fewer difficulties as long as it is understood that within a given file, a record is a set of attributes that serve to identify some entity in the real world. This set is structured according to the general schema that characterizes the file for DIRAC. To the file level is also attached the concept of "class" that is a measure of the total volume of the information it contains. This concept will be defined more precisely below (Part 4, expression (3)).

Fields and subfields

Again, to minimize the confusion between DIRAC and the procedural languages in its environment, we identify as "Field" an attribute whose value is stored within a Record. Thus the name of a patient or the date of an operation in a hospital file, the magnitude of a star or the morphology of a galaxy in an astronomical application are all examples of fields. Once identified by the user, the fields are declared to DIRAC and named during file creation. They are then available for any retrieval operation on the file.

An important characteristic attached to the field level is the Type of the information it contains. This information may be real numeric, integral, alphanumeric, coded, or a date form. The Type of each field, as well as the number of basic fields that compose the Record, once declared, are fixed, although in any given data record fields may, of course, be missing (and the storage structure is such that the final physical record contains no space for that attribute). But any field may be multiple and it may then contain any number of values, possibly with missing data among the list, for any real record. Such values are called Subfields. They have the same type as the field itself and may be addressed individually, as will be seen below.

Structure is the main parameter that varies from one language to another in the DIRAC family. The first prototype does not allow the extension of the tree-structure subdivision below the subfield level. Deeper structures, such as non-cyclic graphs, have been designed and their implementation will begin with DIRAC-2 to permit systematic studies of system performance (overhead minimization in particular) as a function of structure complexity.

Structures above the file level

As convenient as it is for user communication, the concept of file is clearly inadequate in a non-procedural system. Since there is a severe limit to the amount of time the user of a so-called "conversational" system is willing to spend at a terminal waiting for a response, the interactive concept is not compatible with serial file processing. Besides, in a language that allows browsing, the system must dynamically retain information on the user and his past transactions with the data-base. Thus the state of the information at any given time is not necessarily predictable. Intermediate records have to be constructed and retained at several stages of the input/output process. These in turn may be viewed as true files in their own right, and the interrelationships between these satellite files and the primary data file may grow extremely complex.

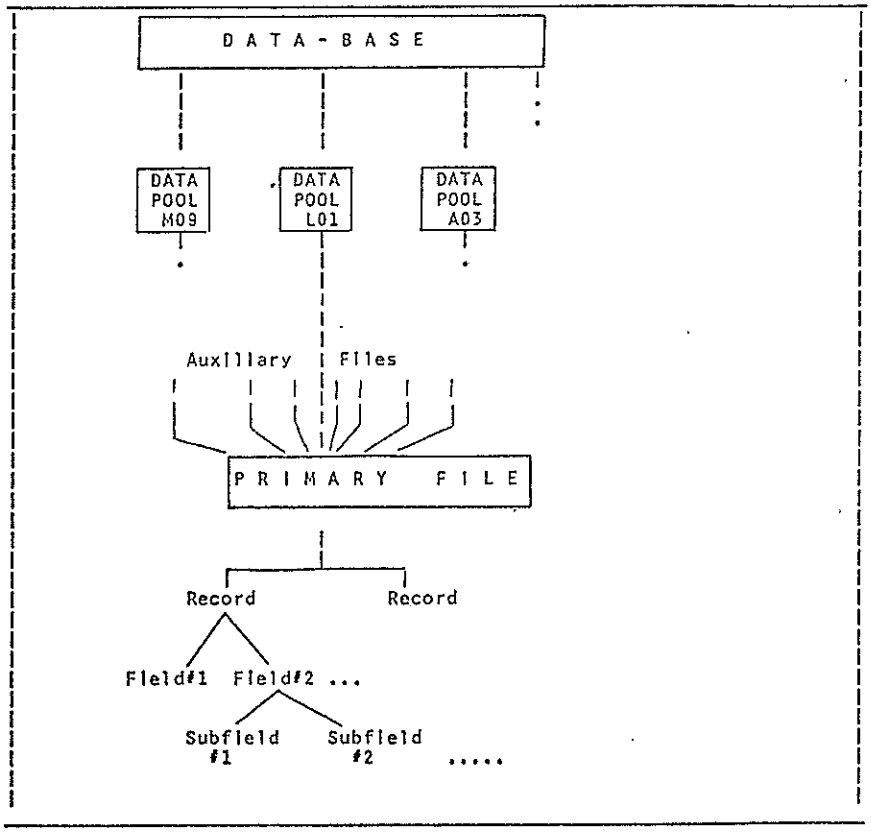

FIG. 1. Structure of the DIRAC Data-Base.

DIRAC-1 recognizes an information organization displayed in Fig. 1. The primary file is "assisted" by at least one and at most fifteen satellite files, in the sense just described.

A primary file, together with its satellites, is called a DATA POOL. The set of all data pools constitutes the data-base. A DIRAC-1 user with full update and query privileges (such as the DB administrator) can query in turn any data pool that has ever been CREATEd under the language; he can also change its contents down to the subfield level without having to issue any operating system command and without having to reinitialize or reload DIRAC. The implications of this language constraint on the system which supports the physical files generated by DIRAC are studied in Part Four of this article. Before we turn to the implementation mechanism, however, it is necessary to discuss in more detail the interactions between such a system and its on-line users.

3. SOFTWARE SUPPORT OF PUBLIC INFORMATION NETWORKS

The environment

One of the major application areas of a language such as DIRAC is found in the support of information systems, in particular those that give remotely-located scientific users a direct link to their data-bases while providing them with a computational facility. In this section

we shall describe the flow of information through such a network in the light of the processing operations that are at the disposal of a DIRAC user in QUERY mode.

In order to illustrate this discussion, examples will be drawn from two data pools that have been sufficiently tested under the DIRAC system in recent months to guarantee that they do in fact indicate patterns of general interest. The first application centers on a hematology file where each record contains all the information obtained in a bone marrow analysis, including textual data such as clinical history of a patient and doctor's impression [3]. The second application uses the Preliminary Warsaw Catalogue of Supernovae, that was converted to machine-readable form in the course of this project; this astronomical catalogue is an ideal test as it contains all the available physical parameters on the known supernovae as well as the titles, authors, references and coded contents of the articles that have been published about them [5].

Access to the data-base

Figure 2 illustrates the hierarchy of access paths to the data-base under DIRAC-1. In addition to the DB Administrator, three levels of network users are recognized. At level 1,

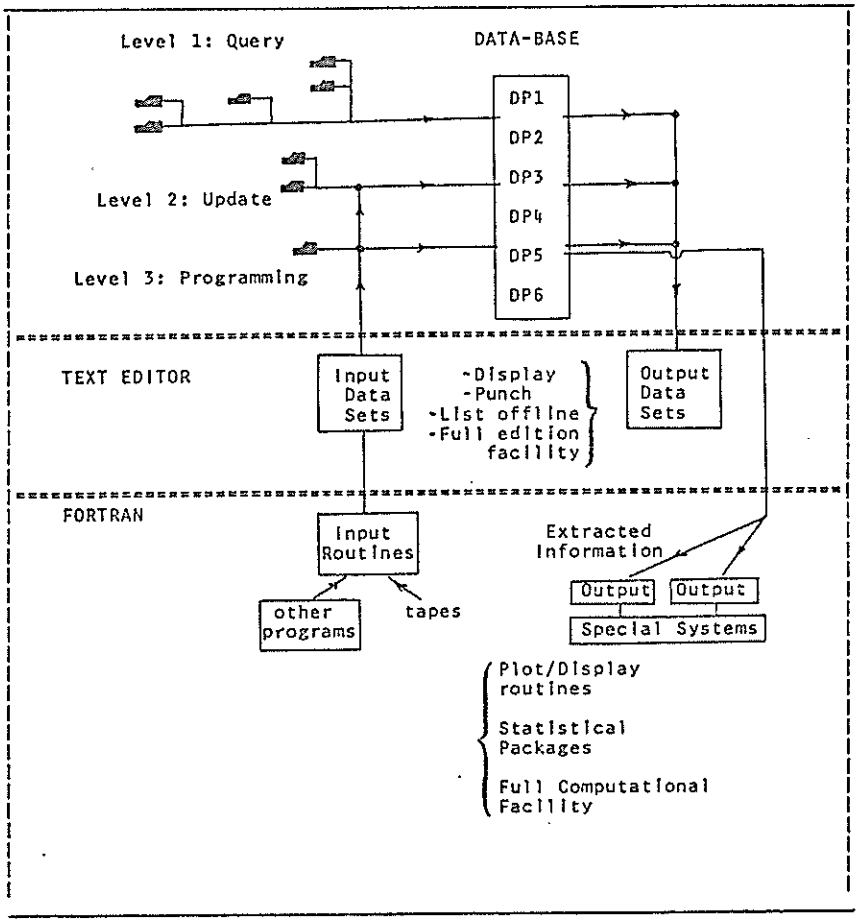

FIG. 2. Hierarchy of access paths to the DIRAC Data-Base: A problem of interfaces.

```
      QUERY
FILE IDENTIFICATION
:     A010
ACTION
:     SELECT
SELECTION RULES
:     Cluster CONTAINS Virgo END
_____________________
24 RECORDS SELECTED
ACTION
:     RETAIN
ACTION
:     SN CONTAINS s END
_____________________
 1 RECORDS SELECTED
ACTION
:     DISPLAY SN Vs Cluster
_____________________
SN          s1922alpha
Vs          1243
Cluster     Virgo
_____________________
 1 RECORDS SELECTED
ACTION
:     RELEASE
ACTION
:     Cluster CONTAINS Virgo AND SN DOES NOT
:     CONTAIN s END
_____________________
23 RECORDS SELECTED
ACTION
:     RETAIN
ACTION
:     Vs EXISTS END
_____________________
19 RECORDS SELECTED
ACTION
:     Vs (<=2000 AND >=1000)END
_____________________
11 RECORDS SELECTED
ACTION
:     Sources(FIRST) CONTAINS "Mt.Wilson" END
_____________________
 1 RECORDS SELECTED
ACTION
:     DISPLAY SN Vs 12 b2 Sources END
_____________________
SN          1901b
Vs          1617
12          271.15
b2          76.90
Sources   1 Ap.J.,88(1938),285-304- Contr.Mt.Wilson, 25 (1938) No.600
          2 XIV Colloque Intern.Astrophys., Paris (1941), 186, 188.
          3 Annales Observ.de Paris, 9 (1945) fasc.1, 165-179.
          4 Astronomie 55 (1941), 78, 106.
          5 Astronomie 63 (1949), 68.
          6 ..............
```

FIG. 3. On-line interrogation of an astronomical catalogue.

the QUERY mode is the only one invoked. At level 2, UPDATE takes place, with an input interface with the text editor (WYLBUR). At level 3, the users are systems programmers who have full use of the text editor like level 2 users, but also utilize the FORTRAN/DIRAC interface to apply statistical routines or other computational packages to information extracted from one or several data pools. Under the text editor, all users have at their disposal display, list, punch and edition facilities that can be used to enhance the report generator supplied under DIRAC. Thus it is quite conceivable that, at one end of the spectrum, we shall find people querying data files exclusively within DIRAC commands, while others will simply view the whole Data-base management system as an input–output channel towards the text editor or towards FORTRAN. Nothing should prevent such a variety of usage, since the pure "retrieval" phase may be only a step in a very complex processing activity which takes place outside the scope of DIRAC. In attempting to cover such complex activities within a single framework, a generalized system would necessarily become cumbersome and would fail its major objective, which is to facilitate the communication of information among its users.

Figure 3 is an example of the on-line query of the Supernovae Catalogue implemented under DIRAC-1. The user is an astronomer who studies supernovae in the Virgo cluster. He first wants to know how many are false or suspected. The system finds one, and he displays the supernova number and the recession velocity, Vs. It will be noted that DIRAC processes information in both upper and lower case, thus simplifying the handling of textual data, especially in the scientific field.

The user then wants to determine how many true supernovae in Virgo have a known Vs. The answer is 19. Restricting the search by use of the RETAIN command, he adds the rule:

$$1000 \text{ km/s} <= \text{Vs} <= 2000 \text{ km/s}$$

The answer is 11. Among these, the astronomer wants DIRAC to locate a supernova for which the first article given as reference has "Mt. Wilson" as its source. DIRAC locates supernova number 1901b. The user is now able to have the velocity, galactic coordinates, and all the literature about the object typed out on the terminal.

Under the DISPLAY command, it is possible to restrict the output to the LIST of selected records, or even to their NUMBER only. Alternatively, the DISPLAY ALL command will generate a complete listing of the information in the current subset. When combined with the text editor interface, these commands give the user a flexible report generation capability.

A second example, shown in Fig. 4, will serve to illustrate further the usefulness of the system in dealing with textual information expressed in natural-language strings rather than in codes or numbers. This situation is typical of many medical applications where very few queries indeed can be anticipated at the time of file implementation, and where the researcher must rely on the ability of the system to allow flexible interaction with the data at run time.

On the example of Fig. 4, the commands RETAIN and RELEASE have not been used; one can see alternative formulations of the selection rules as well as the nesting facility allowed in DIRAC. It should be noted that the query commands of an interactive system need not be as sophisticated as those of a batch system: in the latter case, the user must be able to anticipate very minute details of the information he is addressing; in the interactive mode, general queries can be refined by successive selection rules until the desired subset is obtained, and the process is continuously controlled by the user.

```
ACTION
:     SELECT
SELECTION RULES
:     date < 19691126 AND date >= 19691115
:     END
_____________________
 7 RECORDS SELECTED
_____________________
ACTION
:     date<691126 AND date >=19691115
:     AND (  History CONTAINS "Hodgkin"
:     OR Smear CONTAINS "red cell") END
_____________________
 6 RECORDS SELECTED
_____________________
ACTION
:     date ( < 691126 AND >= 691115) AND (History
:     CONTAINS "Hodgkin" OR Smear CONTAINS "red cell")
:     AND Aspirate EXISTS AND Impression CONTAINS thrombocytopenia
:     END
_____________________
 1 RECORDS SELECTED
_____________________
ACTION
:     DISPLAY ALL
_____________________
Record       305847
Patient      XXXXXXXX
Age          48 yr
Room         E2A
Marrow       B69-687
Doctor       Dr.Z.Lucas
Date         24/NOV/1969
History      48-yr old male 2 months post renal transplant. Decreased
             platelets, WBC and PCV, but increased retics. Hemolysis
             workup in progress.
Smear        Microanglopathic changes are seen. Polychromatophilia is
             noted. Red cells are of varying size and shape. Nucleated
             red cells are present. Platelets are low. There are
             immature myeloid elements.
Aspirate     The red cell activity is increased. Occasional
             megakaryocytes are present.
Impression   There is thrombocytopenia with some megakaryocytes in
             marrow. The smear suggests marked red cell activity, as
             seen with hemolysis. The possibility of extramedullary
             hematopoiesis is also to be considered.

_____________________
ACTION
:     END

AT THIS POINT YOU CAN EXIT (BY TYPING AN EXCLAMATION MARK)
OR SPECIFY A NEW EXECUTION MODE
```

FIG. 4. On-line interrogation of a medical file showing various levels of query complexity.

4. PRINCIPLES OF SYSTEM ORGANIZATION

Early in 1970, DIRAC-1 was implemented on the IBM 360/67 computer of the Campus Facility at Stanford University for a series of tests that demonstrated the flexibility and the cost-effectiveness of the non-procedural approach to retrieval problems. The results of these tests are the subject of separate publications [3, 5]. This version of DIRAC relied on a time-sharing submonitor named ORVYL, designed by R. Fredrickson and his co-workers, and operating under the OS/360-HASP system. This submonitor provides the ability to execute user programs in time-shared mode and it supports the DIRAC data-base on the 2314 disk units.

Although a detailed description of the DIRAC-1 processor is beyond the scope of this article, it is of interest to examine in some detail the principles of the file system, its disk reference minimization algorithm and the chaining structure of its primary file, in order to clarify the remarks we have made above concerning its properties.

The basic concept under ORVYL is that of ownership of files by a group of users, the disk space held by the group being charged to the account number by which it is known to the computer. Access to a file may be extended by the owner of a file to any other group, and the owner may also deny such access, or extend more privileges to the public (defined as the "group" that consists of all account numbers validated for terminal use).

Index records are used to keep pointers to those records that exist. Input/output under the system consists of a request for a service, followed by a wait for completion. DIRAC passes an ATTACH command to the system for every file it uses. This is accomplished by executing a macro that specifies:

—The class of device to be attached
—The name of the file
—The availability of the file to other tasks in execution

All files under DIRAC are attached in shared mode.

The system actually maintains records of 2048 bytes, core storage being divided into pages of 4096 bytes each. A buffer area may not cross more than one page boundary: thus, a $4K$ buffer may begin anywhere but an $8K$ buffer must begin on a $4K$ boundary. DIRAC records are blocked into such $8K$ buffers, and indeed a single data record may use all of 8192 bytes if the user so specifies. The I/O operations result in the handling of four physical records under the system.

Reliance on this physical file implementation in DIRAC is limited in fact to only two modules. The interface has been defined in such a way as to allow DIRAC to run under a different system with a minimum amount of recoding.

In the following, let $\mathscr{F}_0$ and $\mathscr{F}_1$ designate respectively the primary and the reference file in a given data pool with n fields, and let e_{ij} be a bit string of length l identifying the existence property of field i in record j; furthermore, let N_R be the number of records in $\mathscr{F}_0$ and let

$$\mathscr{F}_1 = \{\delta_1, \ldots, \delta_k, \ldots \delta_M\}$$

with

$$M = n.N_R$$

and

$$e_{ij} = \delta_k$$

for

$$k = n(j-1)+i. \tag{1}$$

Let R be a buffer of m_0 words, i.e. a bit string of length $m = b.m_0$ on a computer with b-bit words. Noting that n is known to DIRAC at the end of the CREATE phase, we compute:

$$L = b\left[\frac{m_0}{n}\right] \tag{2}$$

as the elementary bit string in the reference file. The number of overlays of R that will be used in the course of an interrogation or update run is a basic measure of the volume of information that exists in the file and can be written as

$$K = \left\lceil\frac{N_R}{L}\right\rceil. \tag{3}$$

This is the quantity we define as the "class" of the file. If j_0 is the number of the current $\mathscr{F}_0$ record in core, and a decision is to be made whether or not to retrieve record j, we examine bit ρ_k in R such that:

$$k = L(i-1)+(j-j_0+1). \tag{4}$$

In our FORTRAN-based implementation it was convenient to compute the corresponding word number and to examine $R(m_1)$ with

$$m_1 = \left\lceil\frac{k}{b}\right\rceil \tag{5}$$

with the appropriate mask on bit $(k-b.m_1)$.

Within $\mathscr{F}_0$ the chaining structure is generated by the discrete update processor. We propose to illustrate its operation to conclude this discussion of DIRAC by giving both a set of productions from which a simplified state diagram can be constructed, and an example.

The following input symbols are accepted:

N precedes a new record

O designates an old record and is followed by either *D* (delete) or *R* (replace) and an integer indicating the number of the record to be deleted or replaced.

$\alpha_0 \to @/\omega_1 = 1, \alpha_1$
$\alpha_1 \to \phi/\omega_2 = 0, \alpha_2$
$\alpha_2 \to @/\phi, \alpha_0$
$\quad\to N/A(Y), \alpha_3$
$\quad\to O/\phi, \alpha_9$
$\alpha_3 \to P/\phi, \alpha_4$
$\quad\to \neg P/\phi, \alpha_8$
$\alpha_4 \to \phi/\omega_2 = \omega_2+1, \alpha_5$
$\alpha_5 \to \phi/i = \omega_1+1, \alpha_6$
$\alpha_6 \to \phi/X_{\omega_1} = Y, \alpha_7$
$\alpha_7 \to \phi/\omega_1 = i, \alpha_2$
$\alpha_8 \to \phi/i = X_{\omega_1}, \alpha_6$
$\alpha_9 \to D/A(i), \alpha_{10}$
$\quad\to R/A(i), \alpha_{11}$
$\alpha_{10} \to Q/X_i = \#\omega_1, \alpha_7$
$\quad\to \neg Q/F, \alpha_{13}$
$\alpha_{11} \to Q/A(Y), \alpha_{12}$
$\quad\to \neg Q/F, \alpha_{13}$
$\alpha_{12} \to \phi/X_i = Y, \alpha_2$
$\alpha_{13} \to @/\phi, \alpha_0$
$\quad\to \neg @/\phi, \alpha_{13}$

Let $\mathcal{F}_0$ be a sequence of records $X_1, X_2, \ldots$

$\#i$ designate a record pointer containing a single integer i

Y designate a buffer

ω_1, ω_2 designate two counters.

Furthermore, let $A(x)$ be the action of reading an input token into storage area x

P be the logical proposition: $\omega_1 > \omega_2$

Q be the proposition "X is not a record pointer".

Other symbols follow the usual conventions.

Example:

Consider the input string:

@NaNbOR1cNdOD2Ne@.

The resulting file contains the three records: (*c,e,d*) after the steps detailed in the table below:

step	ω_1	ω_2	Y	i	X_1	X_2	X_3
1	1	0					
2	1	0	*a*				
3	1	0	*a*		*a*		
4	2	1	*b*		*a*		
5	2	1	*b*		*a*	*b*	
6	3	2	*b*		*a*	*b*	
7	3	2	*b*		*a*	*b*	
8	3	2	*b*	1	*a*	*b*	
9	3	2	*c*	1	*c*	*b*	
10	3	2	*d*	1	*c*	*b*	
11	3	2	*d*	1	*c*	*b*	*d*
12	4	3	*d*	1	*c*	*b*	*d*
13	4	3	*d*	1	*c*	*b*	*d*
14	4	3	*d*	2	*c*	#4	*d*
15	2	3	*d*	2	*c*	#4	*d*
16	2	3	*e*	2	*c*	*e*	*d*
17	4	3	*e*	2	*c*	*e*	*d*

CONCLUSION

The main novelty in the design of DIRAC is the concept of a generalized file management system that interfaces with, and can be driven from, an interactive text editor. This concept makes it possible to implement catalogued interrogations and complex report generation with minimum complexity.

The second feature in DIRAC that we feel points to a solution of the scientific data-base problem is the opportunity given the user to branch freely into his own code once the basic retrieval function has been accomplished, on a record-by-record basis. Thus an environment is created where non-procedural commands can interface optimally with user-supplied routines.

REFERENCES

[1] Survey of Generalized Data-Base Management Systems. CODASYL Systems Committee, 1969.
[2] WYLBUR Reference Manual. Stanford University Computation Center, Stanford, 1969.
[3] P. L. WOLF, H. R. LUDWIG and J. F. VALLEE: "Progress Towards a Direct-access Hematology Data-Base: Stanford's experience with the DIRAC Language". To be published.
[4] J. F. VALLEE: "Scientific Information Networks: A Case Study", Research Report II, Information Systems Group, Stanford Computation Center, September 1970.
[5] J. F. VALLEE and J. A. HYNEK: "DIRAC and astronomical data retrieval", Proceedings of ACM'70, New York, September 1970.