DOCUMENT RESUME

ED 050 798 LI 002 861

AUTHOR	Vallee, Jacques; Ludwig, Herbert
TITLE	The DIRAC Language: Concepts and Facilities.
INSTITUTION	Stanford Univ., Calif. Computation Center.
REPORT NO	R-1
PUB DATE	May 70
NOTE	53p.
AVAILABLE FROM	Dr. J. Vallee, Stanford Electronics Laboratories, Stanford University, Stanford, California 94305
EDRS PRICE	EDRS Price MF-$0.65 HC Not Available from EDRS.
DESCRIPTORS	Data Bases, *Information Networks, *Information Processing, Information Retrieval, *Information Systems, *On Line Systems, *Programing Languages
IDENTIFIERS	DIRAC, Direct Access Project

ABSTRACT

The three documents contained in this report describe an interactive retrieval language implemented for the IBM 360/67 of the Campus Faculty at Stanford University, between October 1969 and May 1970. The three reports are: (1) DIRAC--An Interactive Retrieval Language with Computational Interface, (2) DIRAC--An Overview of an Interactive Retrieval Language, and (3) Preliminary User's Guide. Two related documents are "Medical Data Management in Time-Sharing: Findings of the DIRAC Project" (see LI 002 823) and "Scientific Information Networks: A Case Study" (see LI 002 829). (MM)

THE DIRAC LANGUAGE
CONCEPTS AND FACILITIES

REPORT NUMBER 1
MAY 1970

JACQUES VALLEE
and
HERBERT LUDWIG

STANFORD UNIVERSITY
COMPUTATION CENTER
INFORMATION SYSTEMS

ED050798

This report contains three documents describing an interactive retrieval language implemented for the IBM 360/67 of the Campus Facility at Stanford University, between October 1969 and May 1970.

1. DIRAC--An Interactive Retrieval language with Computational Interface.
2. DIRAC--An Overview of an Interactive Retrieval language.
3. Preliminary User's Guide.

DIRAC :

AN INTERACTIVE RETRIEVAL LANGUAGE WITH COMPUTATIONAL INTERFACE

Jacques F.Vallee
Stanford University

Address: Dr.J.F.Vallee, Manager
Information Systems
Computation Center
Stanford University
Stanford, California 94305

DIRAC :

AN INTERACTIVE RETRIEVAL LANGUAGE WITH COMPUTATIONAL INTERFACE

Jacques F.Vallee
Stanford University

ABSTRACT

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 implemented for the IBM 360/67 computer of the Campus Facility at Stanford University. 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 in an interactive mode. The current applications of DIRAC described in this article use files of research data in astronomical and medical fields. It operates exclusively in a time-sharing environment under the Stanford time-sharing monitor. The article describes the system and its applications from the point of view of language design and of operating system support requirements.

DIRAC :

AN INTERACTIVE RETRIEVAL LANGUAGE WITH COMPUTATIONAL INTERFACE

Jacques F.Vallee

Stanford University

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 ..le 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.

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 some prototype experimentations currently conducted at Stanford University, 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 th]e 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 of this project 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; 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. (2)

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 inter-related 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 at this point . A rather complex chaining structure is then superimposed to the information which is apparent to the user, and 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 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:

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 interpretive 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

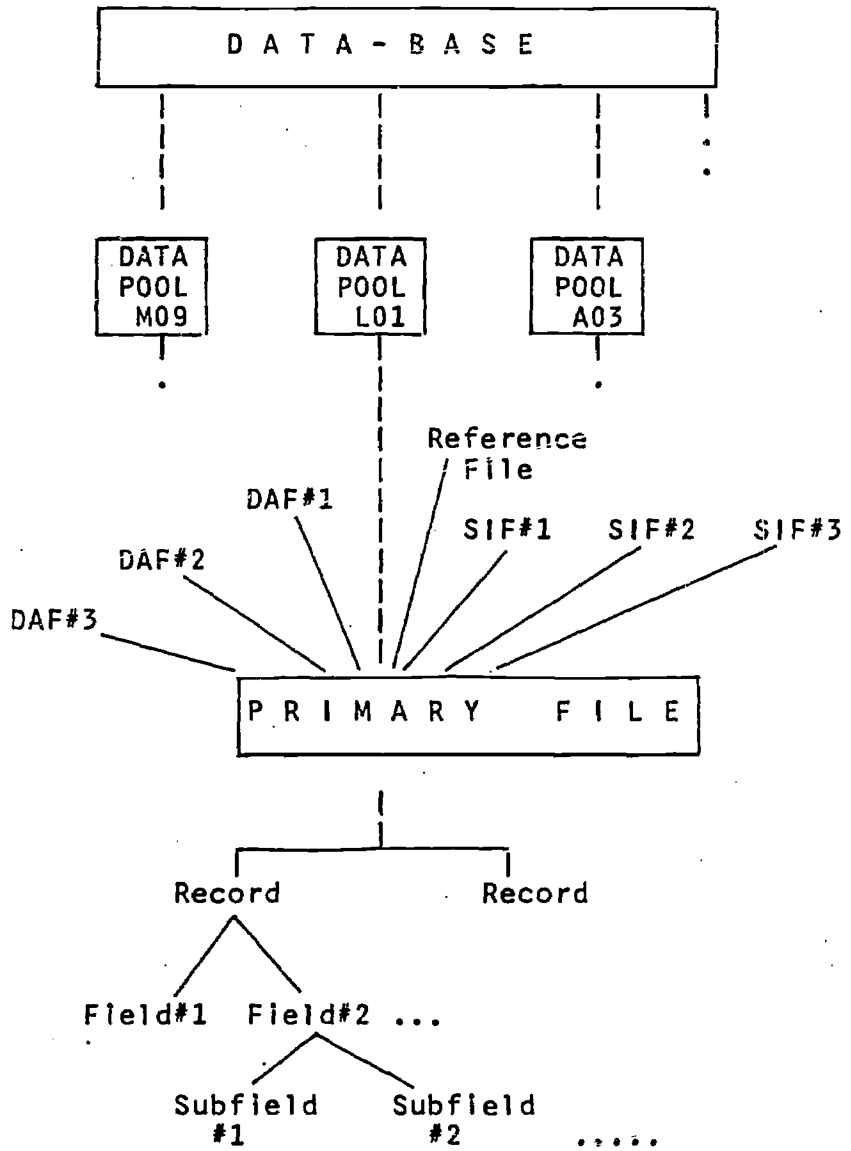


Figure 1: Structure of the DIRAC Data-Base.

'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.

Fields and Subfields.

Again, to minimize the confusion between DIRAC and the procedural languages in its environement, 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 inter-relationships between these satellite files and the primary data file may grow extremely complex.

DIRAC-1 recognizes an information organization displayed on fig.1. The primary file is 'assisted' by at least one and at most fifteen satellite files, in the sense just described. Some of them accumulate system information (SIF) while others serve as auxiliary data files and are mostly useful in supporting the inverting facilities; these merely point to the main file (DAF). 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. 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.

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, 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

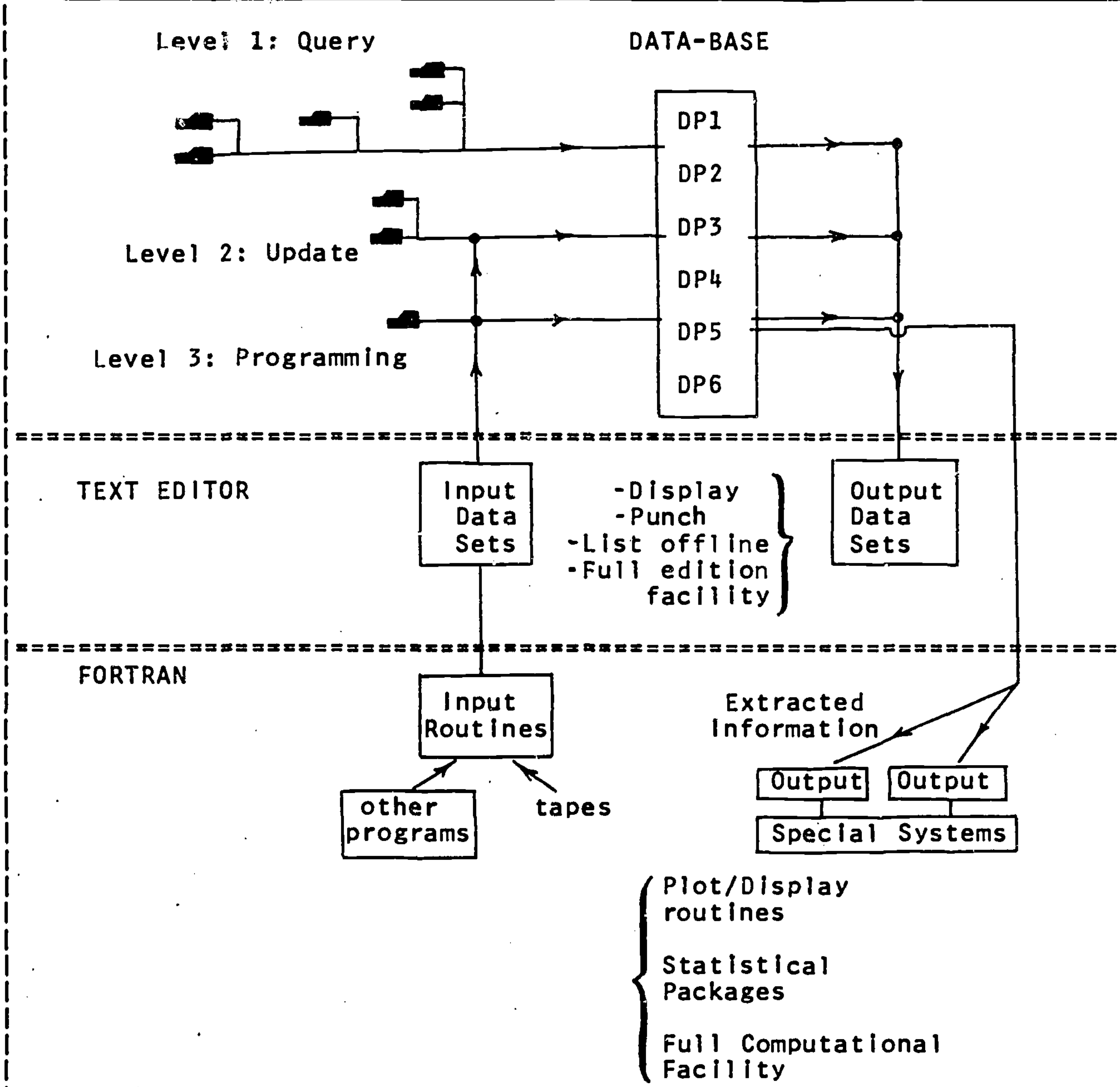


Figure 2: Hierarchy of access paths to the DIRAC Data-Base: A problem of interfaces.

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 miss its major objective, which is to facilitate the communication of information among its users.

Survey of interrogation commands

There are five fundamental commands utilized in QUERY mode:

- The SELECT command initializes the definition of a sequence of selection rules that define a subset of the primary data file.

- The DISPLAY command is used to type out information about the particular subset currently selected. When the volume of information is large, however, the DISPLAY action can be triggered through the text editor (The command typed at the terminal is then 'DISPLAY WYLBUR') and printing can be done off-line on a high-speed printer.

- The RETAIN command is used to save the current subset. The resulting records are usually processed again by further selection until the search has been narrowed to the desired information.

- The RELEASE command completes the browsing facility by allowing re-initialization of the search to the entire file. In later versions of DIRAC this command will be combined with a subset designation to allow a hierarchy of embedded subsets rather than the simple concept of a single filter, as currently implemented in DIRAC-1.

- The EXTRACT command, similar in form to the DISPLAY command, transmits specified information through a computational interface with FORTRAN. User's own code can then operate along with DIRAC modules to achieve complex computations that are not possible within the basic file-oriented commands. As a default, the current implementation generates cross-tabulation of extracted fields and can be expanded to include standard post-processing for any particular application.

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 km/s <= Vs <= 2000 km/s

```
      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 l2 b2 Sources END
____________________
SN          1901b
Vs          1617
l2          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 .............
```

Figure 3: On-line interrogation of an astronomical catalogue

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 on figure 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 figure 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

```
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         Microangiopathic 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
```

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

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.

?

4. THE CURRENT IMPLEMENTATION

In its current state on the computer we have at our disposal, DIRAC relies on a time-sharing submonitor that operates under the OS/360-HASP system. This submonitor provides the ability to execute user programs in a time-shared mode, and it supports the DIRAC data-base on the 2314 disks.

The basic concept under this system 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.

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 difficulty.

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.

Reference:

(1) Survey of Generalized Data-Base Management Systems. CODASYL Systems Committee, 1969.

(2) WYLBUR Reference Manual. Stanford University Computation Center. Stanford, California.

D I R A C

An Overview of An Interactive Retrieval Language

by

J. Vallee and H. Ludwig
Stanford University

1. INTRODUCTION

The language described here is the first prototype in a family of information oriented languages studied at the Stanford Computation Center. The objective of the project is to expand the services currently offered by the Campus Facility in application areas that demand flexible interaction with large files and to generate ideas and techniques applicable to industrial situations. The language is called DIRAC. It is non-procedural and demands no previous computer experience on the part of the user. It allows creation, updating, bookkeeping operations, and the querying of data files in conversational mode under a time-sharing monitor on the IBM 360/67. It interfaces with the Stanford text editor, WYLBUR, and with the user's own FORTRAN code when complex computations on the contents of the files are required.

2. THE DIRAC SYSTEM

DIRAC (Date, Integer, Real, Alphanumeric, and Coded) is an information retrieval language which provides the user the ability to operate under four modes: CREATE, UPDATE, QUERY and STATUS.

(1) The CREATE mode allows the user to completely define the terminology and structure of his own file.

(2) The UPDATE mode allows such operations as adding, deleting or replacing records.

(3) The QUERY mode of DIRAC allows the user to obtain information about SELECTed subsets of his file at any level of the record structure. The different commands through which a file may be queried are described in this article.

(4) The STATUS mode provides the user with an up-to-date status report for his particular file. Field identification, description of the fields, statistics and validation information are displayed in a standard report form.

3. FILE STRUCTURES FOR DIRAC

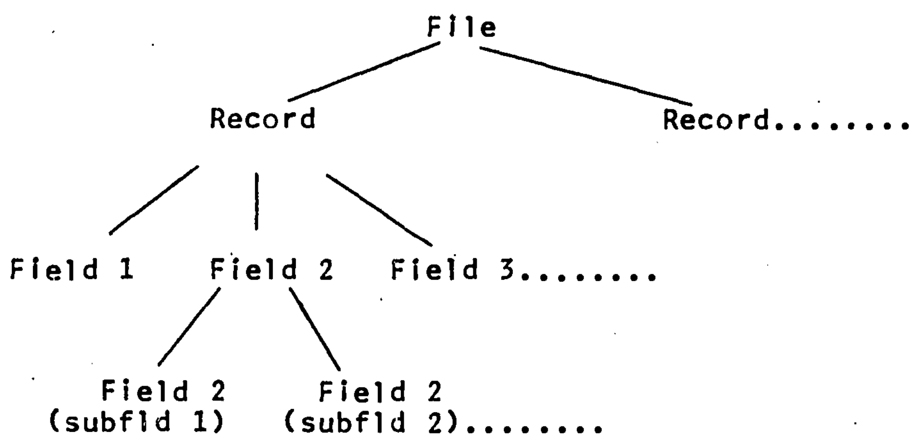
3.1 Files and Records

A file is defined here as a collection of related records containing data needed for subsequent processing. This need may arise in the regular course of a routine utilization of the data. Alternatively, it may be necessary to answer unpredictable queries about a file, and the latter situation causes many difficulties under standard, procedural languages. DIRAC addresses itself to the need of facilitating data retrieval in response to inquiries and requests for special analysis.

3.2 Fields and Subfields

Within a DIRAC record every attribute is identified as an individual field: a patient's name in a hospital record, a social security number, a charge account number 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 type of retrieval response from the file. Fields of a record can be numeric integer such as a charge number, numeric real such as purchases within that charge account (xx.xx), alphabetic such as name or address; they can also be dates or codes.

A record consists of fields which may themselves be formed from two or more subfields. This process of subdivision (tree structure) can theoretically be continued.



However, in the first version of DIRAC representations will not be supported beyond the subfield level. Such data structures will be introduced beginning with DIRAC-2 when a suitable data base has been constructed. (full compatibility between the two languages being preserved).

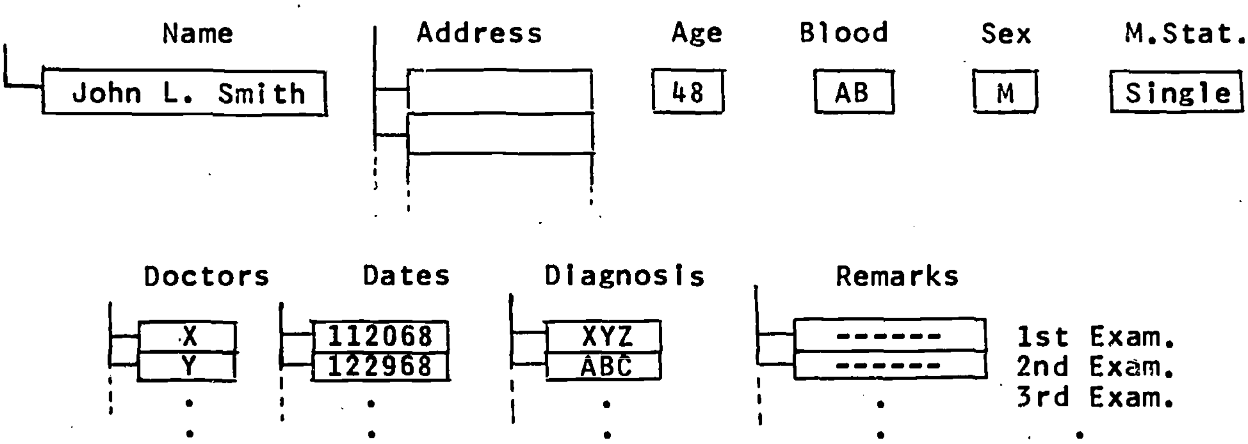
3.3 Setting up a File Under DIRAC

DIRAC provides the user with the opportunity to completely specify his own file organization. Thus, the user does not have to be concerned about using a fixed field or fixed word type of format. The user is not bound by a set of rigid rules pertaining to record size, length, etc., and these parameters are not even apparent to him.

The user should first compile a working list of all fields which he wants contained in a record, specifying whether or not a field is singular or multiple (subfields). Example: Suppose that we were to create a DIRAC file of patients for a hospital; we have determined that we wanted to include the following information (fields) in a patient's record:

Patient's Name
Home Address
Age
Blood Type
Sex
Marital Status
Doctor(s)
Date(s) of Examination
Diagnosis
Remarks or Impressions

A typical Patient Record would have the structure:



Note that the fields Address, Doctor, Date, Diagnosis, and Remarks are multiple. In other words a given patient might have seen several doctors over the past year(s); some of the doctors possibly appearing several times in the list. In each examination, which took place on a given date, a diagnosis was made and some remarks were recorded by the doctor.

The user must also determine the "type" of each field which he includes as part of a record. For example, patient's name would be alphanumeric (ALPHA), whereas age probably would be integer Blood type and sex could be either alpha or coded in the example given above.

After determining the type of each field and whether or not that field is singular or multiple, the fields can be numbered as follows:

FIELD	NAME	DESCRIPTION
1	Name	Patient's Name
2	Address	Patient's Home Address
3	Age	---
4	Type	Blood Type
5	Sex	---
6	Status	Marital Status
7	Doctors	Doctors Seen by Patient
8	Date	Date(s) Seen
9	Diagnosis	---
10	Impression	General Remarks by Doctor

A delimiter will be picked from a set of special characters (such as @,$,#) to denote a field in DIRAC. (The user can pick any delimiter out of the list which is convenient to him, thus avoiding the need for a rigid standard notation imposed by most existing systems.)

DIRAC will prompt the user for Type and Multiplicity of the fields within a record. In our example the following information would then be typed at the terminal: (the underlined portions are the prompts of DIRAC) prompts of DIRAC)

TYPE AND MULTIPLICITY

INTEGER SINGLE @3
ALPHA SINGLE @1 @2 @4 @6 @5
ALPHA MULTIPLE @7 @9 @10
DATE MULTIPLE @8

The user should note that field specifications can be input in any order. Also note that the delimiter "@" was used to specify fields. "Integer Single" means that the value to be stored in field 3 will be a single integer number. "Alpha Multiple" means that there EXISTS a multiple field in which alphanumeric information is stored. From the example we note that fields @7 - @10 are multiple. Thus, when reference is made to @7(1) -- the name of a doctor -- the date, diagnosis, and impression for that visit are contained in @8(1), @9(1), @10(1), respectively.

3.4 Actual Input into a DIRAC File

Once the file has been specified by the user to DIRAC, the user will start updating this empty structure. DIRAC file. DIRAC will prompt the user with "NEW". The user can now input information into the DIRAC file under the following rules:

(1) Fields can be listed in any order and without regard for information length.
(2) Empty fields need not be listed.
(3) In the "multiple" case subfields can be listed in any order and empty subfields need not be defined.
(4) Alpha values must be enclosed in quotes if the string contains a delimeter or a blank.
), <, >, =, /, ?. *

EXAMPLE:

NEW

```
@1 "John Smith"
@2 "1426 So. Magnolia St., San Francisco, Calif."
@3 28
@5 M
@4 A
@10(2) "Prescribed long rest in bed"
@10(3) "Quarentined for one month"
@7(1) "Dr. Jones"
@7(2) "Dr. Paul Woodward"
@7(3) "Dr. William Lowell"
@9(2) "Minor Cold"
@9(3) Measles
@9(1) Flu
@8(2) "3-2-68"
@8(3) "4-3-69"
@8(1) "2-4-68"
```

One record has now been generated and input into the DIRAC file. To start a new record the user must type the word NEW (All commands to DIRAC must be capitalized. The information that goes into the file, however, may contain any character, in upper or lower case, from the terminal character set, with the exception that quotes may not appear within a string). All following records are treated in a similar manner. In the above example John Smith visited Dr. Jones on April 3, 1968. It was diagnosed that he had the flu and no remarks were made!

4. DIRAC "QUERY" MODE

In this general presentation of the language we shall describe only the five fundamental commands utilized by the DIRAC query mode.

(1) SELECT - Initializes the definition of a sequence of SELECTion rules that define a subset of the data file.

(2) EXTRACT - Used to transmit specific field information from a record through a computational interface with FORTRAN. As a default, this command will generate cross-tabulations among the extracted fields.

(3) RETAIN - Used after the Select command has been executed to save the current subset. The resulting records are usually processed again by further SELECTion until the search has been narrowed to the desired information--this is equivalent to a a "start browsing" command.

(4) DISPLAY - Used to print out information obtained through Select commands. If the volume of information is large then printing can be done offline on high speed printer.

(5) RELEASE - In contrast to the RETAIN command, this re-initializes the search to the entire data file.

4.1 The SELECT Command

The SELECT command permits interrogation of a set of specified fields by the following SELECTion rules. The user may write:

```
(Field Name or Number) DOES NOT CONTAIN (value)

                      --- CONTAINS (Value)  for alpha, coded
                                            or real fields

                      --- =,<,>,<=,>= (Value)

                      --- EXISTS                for any field

                      --- DOES NOT EXIST
```

where "Value" is real, integer, or alpha, depending on the mode of the operand. The above SELECTion rules can also be combined into a logical expression of any length and complexity.

EXAMPLE:

```
ACTION
        SELECT
SELECTION RULES
:       @7<19691126 END
```

Field 7 is tested and all records where field 7 EXISTS and has a value less than 19691126 are SELECTed.

EXAMPLE:

```
ACTION
:       SELECT
SELECTION RULES
:       @7<1961126 AND @7 >= 1961115 END
```

All records whose field 7 is less than 19691126 and greater than or equal to 19691115 are SELECTed; the first date form has been automatically restored to year 1969.

EXAMPLE:

```
ACTION
:       SELECT
SELECTION RULES
:       @3<35 AND @3 >=25
:       AND (@7(1) CONTAINS "Jones" OR @9(1) CONTAINS "Flu") END
```

All records whose field 3 is less than 35 and greater than or equal whose field 9, subfield 1, CONTAINS the word "Flu" are SELECTed.

EXAMPLE:

```
ACTION
:       SELECT
SELECTION RULES
:       @3 (<35 AND >=25) AND (@7(1) CONTAINS "Jones"
:       OR @9(1) CONTAINS "Flu")
:       AND @10 EXISTS
:       AND @2 CONTAINS "Calif." END
```

All records whose field 3 is less than 35 and greater than or equal to 25 AND whose field 7, subfield 1, CONTAINS the word "Jones" OR whose field 9, subfield 1, CONTAINS the word "Flu" AND whose field 10 EXISTS and whose field 2 CONTAINS the word "Calif." are SELECTed.

The need to actually type the command SELECT after the prompt ACTION is optional: To speed up user-machine interaction, DIRAC assumes that anything that does not begin with a command at this point must be a SELECTion rule. If an error is encountered, it is then diagnosed as an error in a SELECTion rule and recovery proceeds accordingly.

EXAMPLE:

```
ACTION
:       @9 CONTAINS .5 END
```

In every record where it EXISTS, field number 9 will be scanned to determine whether it CONTAINS a decimal point followed by the digit 5. This will retrieve records where field 9 contains a real number such as .51, 19.595, 0.519622, etc. (This rule may appear obscure in a strictly numerical sense. In library or medical applications, however, the digits of a real number may have individual meaning and may be susceptible to SELECTion as such)

4.2 The EXTRACT Command

In some cases the user wishes to access DIRAC records only as a preliminary step in a more complex computational program. Such a computational interface exists in DIRAC and functions as follows. The user writes

```
EXTRACT(List of fields) END
```

```
ACTION
:       Name EXISTS AND Age<25
:       AND Type CONTAINS AB END

5 RECORDS SELECTED

ACTION
:       EXTRACT Name   END
```

All records are SELECTed for which Name (@1 - Name of Patient) EXISTS AND Age (@3 - Age of Patient) is less than 25 AND Type (@4 - Blood Type of Patient) contains the letters AB. Five records were found to satisfy this logical expression. From these 5 records "Name" was extracted. (Exhibit A)

4.3 The RETAIN/RELEASE Commands

The RETAIN command allows the user to keep (RETAIN) those records which have just been SELECTed and apply another SELECT command to that set. The user can thus narrow down a given set of records until the desired set is obtained by using the RETAIN command.

EXAMPLE:

```
ACTION
:        @4 CONTAINS AB END

24 RECORDS SELECTED

ACTION
:        RETAIN
ACTION
:        @3 < 25 END

5 RECORDS SELECTED

ACTION
:        @5 CONTAINS F OR @5 CONTAINS FEMALE END

3 RECORDS SELECTED

ACTION
:        RELEASE
ACTION
:        @3 <25 END

13 RECORDS SELECTED
```

The different blood types stored in field 4 are scanned for the letters 'AB'. 24 records are found to exist with this blood type. These 24 records are now RETAINed. From these 24 records now, field 3 is tested for an age less than 25. 5 records are found to exist with Age less than 25 in field 3. Field 5 for these 5 records is now tested for a value of F or the word FEMALE. One record is found. Note that the RETAIN command need only be exercised once to successively RETAIN following SELECTed records. It serves essentially to define a "filter" over the file while giving the user an interactive browsing facility. When the whole file was tested for @3 < 25, 13 records were obtained, thus the RELEASE command allows the user to address his SELECTion rules to the whole file again after working under the RETAIN command as shown above.

EXAMPLE OF EXTRACT COMMAND

```
ACTION
:        SELECT
SELECTION RULES
:        Name EXISTS AND (Age<25)
:        AND Type CONTAINS AB END

5 RECORDS SELECTED

ACTION
:        EXTRACT Name END

5 RECORDS SELECTED


FIELD 1 TAKES 5 VALUES.

John Smith   Howard Levin   George Garth
Fred Henny   Frank Martel
```

Exhibit A

4.4 The DISPLAY Command

This command is used when the user wishes to type out the information obtained by the previous SELECT command. The user writes

```
        DISPLAY(List of field names or numbers) END
   or   DISPLAY ALL
   also DISPLAY NUMBER
        DISPLAY LIST
        DISPLAY (Record number)
```

(Note Exhibit B)

In many cases, however, the typing of the information in this form is not practical, either because it is too long, or because several copies are needed or because the extraction done through DIRAC is only one step in a more complicated editing task. To solve this problem the user writes

```
        DISPLAY WYLBUR (List of fields) END
   or   DISPLAY WYLBUR ALL
```

WYLBUR is the name of the interactive text editor developed at Stanford(*)

(*) see: "WYLBUR on the IBM 36-/67: A Time Sharing, Fast Remote Batch, Text Editing and Job-Shop System", by Rod Fredrickson. Available from Information Services, Stanford University Computation Center. (Note Exhibit C)

5. CONCLUSION

An interactive retrieval language suitable for a widerange of business, research and library applications has been proposed. A prototype implementation for a particular computer (the IBM 360/67) is currently the object of experiments by the Information Systems group at Stanford University. This non-procedural language is original in two respects: first, it gives the user an opportunity to drive the file creation and file update phases from the text editor. Extended to the query phase, this concept leads to catalogued interrogations and complex report generation. Thus, DIRAC represents a departure from those retrieval languages that attempt to combine both the text editing and the file management features within a single package. We believe the approach taken here leads to greater flexibility and easier application to real-life processing situations.

Second, it provides a computational interface with the user's own code, at the same time avoiding the problems of the "host-language" systems. DIRAC is utilized at Stanford to build a data-base on which file structures of increasing complexity can be tested in a concrete, quantitative manner.

DIRAC COMMANDS

```
ACTION
:          RETAIN

ACTION
:          Name EXISTS

64 RECORDS SELECTED

ACTION
:          Age < 20 AND Sex CONTAINS Male END

3 RECORDS SELECTED

ACTION
:          DISPLAY Name Age Sex Type END

________________________________
   18
Name                John Smith
Age                 19
Sex                 Male
Type                AB
________________________________
   43
Name                George Farmer
Age                 18
Sex                 Male
Type                AB
________________________________
   55
Name                Harold Price
Age                 18
Sex                 Male
Type                0
________________________________
            3 RECORDS SELECTED
ACTION
:           DISPLAY WYLBUR Name Age Sex Type
:           END
________________________________
            3 RECORDS SELECTED
```

Exhibit B

WYLBUR DATA SET

```
?list
  0.001  ________________________________
  0.002  Name            John Smith
  0.003  Age             19
  0.004  Sex             Male
  0.005  Type            AB
  0.006  ________________________________
  0.007  Name            George Farme
  0.008  Age             18
  0.009  Sex             Male
  0.01   Type            AB
  0.011  ________________________________
  0.012  Name            Harold Pric
  0.013  Age             18
  0.014  Sex             Male
  0.014  Type            0
```

Exhibit C

D - DATE
I - INTEGER
R - REAL
A - ALPHANUMERIC
C - CODED

FIRST VERSION

PRELIMINARY USER'S GUIDE

TABLE OF CONTENTS

1. INTRODUCTION

The language described here is the first prototype in a family of information oriented languages developed by the Stanford Computation Center. The objective of the project is to expand the services currently offered by the Campus Facility in application areas that demand flexible interaction with large files. The language is called DIRAC. It is non-procedural and demands no previous computer experience on the part of the user. It allows creation, updating, bookkeeping operations, and the querying of data files in conversational mode. It interfaces with the Stanford text editor, WYLBUR, and with the user's own FORTRAN code when complex computations on the contents of the files are required.

2. THE DIRAC SYSTEM

DIRAC (Date, Integer, Real, Alphanumeric, and Coded) is an information retrieval language which provides the user the ability to operate under four modes: CREATE, UPDATE, QUERY and STATUS.

(1) The CREATE mode allows the user to completely define the terminology and structure of his own file.

(2) The UPDATE mode allows such operations as adding, deleting or replacing records.

(3) The QUERY mode of DIRAC allows the user to obtain information about SELECTed subsets of his file at any level of the record structure. The different commands through which a file may be queried are described in this section.

(4) The STATUS mode is the fourth execution mode in DIRAC. It provides the user with an up-to-date status report for his particular file. Field identification, description of the fields, statistics and validation information are displayed in a standard report form.

3. FILE STRUCTURES FOR DIRAC

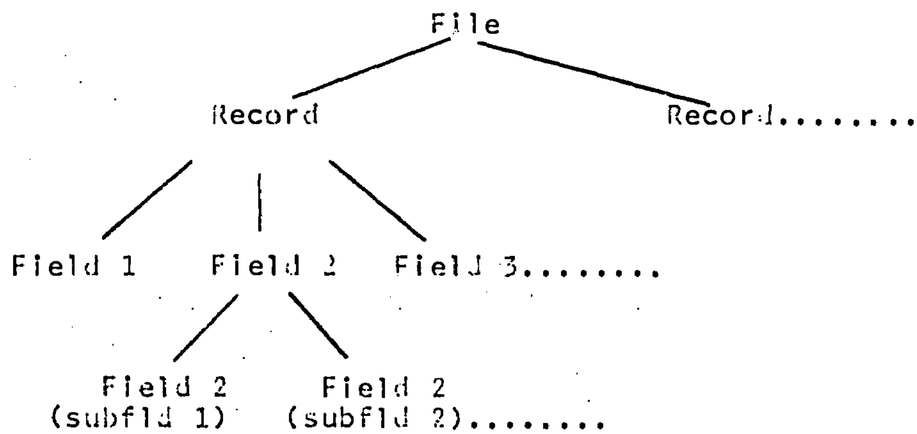
3.1 Files and Records

A file is defined here as a collection of related records containing data needed for subsequent processing. This need may arise in the regular course of a routine utilization of the data. Alternatively, it may be necessary to answer unpredictable queries about a file, and the latter situation causes many difficulties under standard, procedural languages. DIRAC addresses itself to the need of facilitating data retrieval in response to inquiries and requests for special analysis.

3.2 Fields and Subfields

Within a DIRAC record every attribute is identified as an individual Field: a patient's name in a hospital record, a social security number, a charge account number 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 type of retrieval response from the file. Fields of a record can be numeric integer such as a charge number, numeric real such as purchases within that charge account (xx.xx), alphabetic such as name or address; they can also be dates or codes.

A record consists of fields which may themselves be formed from two or more subfields. This process of subdivision (tree structure) can theoretically be continued.



However, in the first version of DIRAC representations will not be supported beyond the subfield level. Such data structures will be introduced beginning with DIRAC2 when a suitable data base has been constructed. (full compatibility between the two languages being preserved)

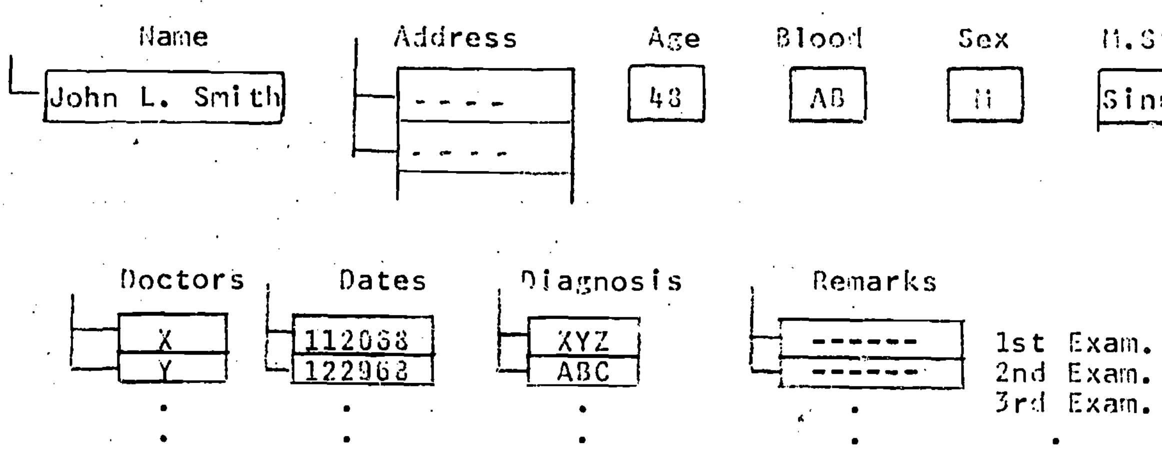
3.3 Setting up a File Under DIRAC

DIRAC provides the user with the opportunity to completely specify his own file organization. Thus, the user does not have to be concerned about using a fixed field or fixed word type of format. The user is not bound by a set of rigid rules pertaining to record size, length, etc., and these parameters are not even apparent to him.

The user should first compile a working list of all fields which he wants contained in a record, specifying whether or not a field is singular or multiple (subfields). Example: Suppose that we were to create a DIRAC file of patients for a hospital; we have determined that we wanted to include the following entries (fields) in a patient's record:

Patient's Name
Home Address
Age
Blood Type
Sex
Marital Status
Doctor(s)
Date(s) of Examination
Diagnosis
Remarks or Impressions

A typical Patient Record would have the structure:



Note that the fields Address, Doctor, Date, Diagnosis, and Remarks are multiple. In other words a given patient might have seen several doctors over the past year(s); some of the doctors possibly appearing several times in the list. In each examination, which took place on a given date, a diagnosis was made and some remarks were recorded by the doctor.

The user must also determine the type of each field which he includes as part of a record. For example, patient's name would be alphanumeric (ALPHA), whereas age probably would be integer Blood type and sex could be either alpha or coded in the example given above.

After determining the type of each field and whether or not that field is singular or multiple, the fields can be numbered as follows:

FIELD	NAME	DESCRIPTION
1	Name	Patient's Name
2	Address	Patient's Home Address
3	Age	---
4	Type	Blood Type
5	Sex	---
6	Status	Marital Status
7	Doctors	Doctors Seen by Patient
8	Date	Date(s) Seen
9	Diagnosis	---
10	Impression	General Remarks by Doctor

A delimeter should now be picked from the set -- % $: # ; @ ¢ --. This delimeter will now be used to define a field in DIRAC. (The user should pick any delimeter out of the list which is convenient to him)

DIRAC will prompt the user for Type and Multiplicity of the fields within a record. In our example the following information would be given to DIRAC by the user: (the underlined lines are the prompts of DIRAC)

TYPE AND MULTIPLICITY

```
INTEGER SINGLE @3
ALPHA SINGLE @1 @2 @4 @6 @5
ALPHA MULTIPLE @7 @9 @8 @10
```

The user should note that field specifications can be input in any order. Also note that the delimeter "@" was used to specify fields. "Integer Single" means that the value to be stored in field 3 will be a single integer number. "Alpha Multiple" means that there EXISTS a multiple field in which alphanumeric information is stored. From the example we note that fields @7 - @10 are multiple. Thus, when reference is made to @7(1) -- the name of a doctor -- the date, diagnosis, and impression for that visit are contained in @8(1), @9(1), @10(1), respectively.

3.4 Actual Input into a DIRAC File

Once the file structure has been specified by the user to DIRAC, the user will want to input information (records) into the DIRAC file. DIRAC will prompt the user with "NEW". The user can now input information into the DIRAC file under the following rules:

(1) Fields can be listed in any order.
(2) Empty fields need not be listed.
(3) In the "multiple" case subfields can be listed in any order and empty subfields need not be defined.
(4) Alpha values must be enclosed in " if the string CONTAINS the following symbols: Blank, *, (,), <, >, =, /, ?.

EXAMPLE:

NEW

```
@1 "John Smith"
@2 "1426 So. Magnolia St., San Francisco, Calif."
@3 28
@5 M
@4 A
@10(2) "Prescribed long rest in bed"
@10(3) "Quarentined for one month"
@7(1) "Dr. Jones"
@7(2) "Dr. Paul Woodward"
@7(3) "Dr. William Lowell"
@9(2) "Minor Cold"
@9(3) Measles
@9(1) Flu
@8(2) "March 2, 1968"
@8(3) "April 3, 1969"
@8(1) "Feb. 4, 1968"
```

One record has now been generated and input into the DIRAC file. To start a new record the user must type the word NEW (All commands to DIRAC must be capitalized. The information that goes into the file, however, may contain any character, in upper or lower case, from the terminal character set, with the exception that the character " may not appear within a string). All following records are treated in a similar manner. In the above example John Smith visited Dr. Jones on Feb. 4, 1968. It was diagnosed that he had the flu and no remarks were made!

4. DIRAC "QUERY" MODE

There are five fundamental commands utilized by the DIRAC query mode.

(1) SELECT - Initializes the definition of a sequence of SELECTion rules that define a subset of the data file.

(2) EXTRACT - Used to transmit specific field information from a record through a computational interface with FORTRAN. As a default, this command will generate cross-tabulations among the extracted fields.

(3) RETAIN - Used after the Select command has been executed to save the current subset. The resulting records are usually processed again by further SELECTion until the search has been narrowed to the desired information.

(4) DISPLAY - Used to print out information obtained through Select commands. If the volume of information is large then printing can be done offline on high speed printer.

(5) RELEASE - In contrast to the RETAIN command, this re-initializes the search to the entire data file.

4.1 The "SELECT" Command

This command will probably be the most used by the user. The SELECT command permits the user to interrogate a set of specified fields by the following SELECTion rules. The user may write:

```
(Field Name or Number) DOES NOT CONTAIN (value)

(Field Name or Number) CONTAINS (Value)   for alpha, coded
                                          or real fields

(Field Name or Number) =,<,>,<=,>= (Value) ⎫
                                           ⎬ for any field
(Field Name or Number) EXISTS              ⎪
                                           ⎪
(Field Name or Number) DOES NOT EXIST      ⎭
```

where "Value" is real, integer, or alpha, depending on the mode of the operand. The above SELECTion rules can also be combined into a logical expression of any length and complexity.

EXAMPLE:

```
ACTION
        SELECT
SELECTION RULES
:       @7<19691126 END
```

Field 7 (@7) is tested and all records where field 7 EXISTS and has a value less than 19691126 are SELECTed.

EXAMPLE:

```
ACTION
:       SELECT
SELECTION RULES
:       @7<691126 AND @7 >= 1961115 END
```

All records whose field 7 is less than 691126 and greater than or equal to 1691115 are SELECTed.

EXAMPLE:

```
ACTION
:       SELECT
SELECTION RULES
:       @3<35 AND @3 >=25
:       AND (@7(1) CONTAINS "Jones" OR @9(1) CONTAINS "Flu") END
```

All records whose field 3 is less than 35 and greater than or equal whose field 9, subfield 1, CONTAINS the word "Flu" are SELECTed.

EXAMPLE:

```
ACTION
:        SELECT
SELECTION RULES
:        @3 (<35 AND >=25) AND (@7(1) CONTAINS "Jones"
:        OR @9(1) CONTAINS "Flu")
:        AND @10 EXISTS
:        AND @2 CONTAINS "Calif." END
```

All records whose field 3 is less than 35 and greater than or equal to 25 <u>AND</u> whose field 7, subfield 1, CONTAINS the word "Jones= <u>OR</u> whose field 9, subfield 1, CONTAINS the word "Flu" <u>AND</u> whose field 10 EXISTS and whose field 2 CONTAINS the word "Calif." are SELECTed.

The need to type the command SELECT after the prompt ACTION has been eliminated. <u>DIRAC</u> assumes that anything that does not begin with a command at this point must be a SELECTion rule. If an error is encountered, it is then diagnosed as an error in a SELECTion rule and recovery proceeds accordingly.

The Selections can be applied to record fields under the following rules:

(1) For any "Alpha", "Real", or "Coded" field -- CONTAIN or DOES NOT CONTAIN can be used.

(2) For any field -- EXISTS or DOES NOT EXIST can be used.

(3) Inequalities apply to all fields.

EXAMPLE:

```
ACTION
:        SELECT
SELECTION RULES
:        @9 CONTAINS .5 END
```

In every record where it EXISTS, field number 9 will be scanned to determine whether it CONTAINS a period followed by the digit 5. (This rule may appear obscure in a strictly numerical sense. In some library or medical applications, however, the digits of a real number may have individual meaning and may be susceptible to SELECTion as such)

4.2 The EXTRACT Command

In some cases the user wishes to access <u>DIRAC</u> records only as a preliminary step in a more complex computational program. Such a computational interface EXISTS in <u>DIRAC</u> and functions as follows. The user writes

EXTRACT(List of fields) END

EXAMPLE: (the following examples are drawn from an astronomy file on supernovae. The field names and descriptions are described in Appendix E. Knowledge of astronomy is not necessary in order to understand the following concepts)

ACTION
: Vs EXISTS AND Morphology EXISTS
: AND Cluster CONTAINS Virgo END

23 RECORDS SELECTED

ACTION
: EXTRACT Morphology END

All records are SELECTed for which Vs (@10 - Recession Velocity in km/s) AND Morphology (@8 - Morphology of Parent) exist AND Cluster (@11 - Cluster Membership of Parent) CONTAINS the word "Virgo". 23 records were found to satisfy this logical expression. From these 23 records Morphology was extracted. (Exhibit A)

4.3 The RETAIN Command

The RETAIN command allows the user to keep (RETAIN) those records which have just been SELECTed and apply another SELECT command to that set. The user can thus narrow down a given set of records until the desired set is obtained by using the RETAIN command.

EXAMPLE:

ACTION
: SELECT
SELECTION RULES
: @11 CONTAINS Virgo END

24 RECORDS SELECTED

ACTION
: RETAIN
ACTION
: @1 CONTAINS S END

5 RECORDS SELECTED

ACTION
: @10 <999 END

1 RECORD SELECTED

The text stored in field 11 is scanned for the word "Virgo". 24 records are found to exist with this word. These 24 records are now RETAINed. From these 24 records now, field 1 is tested for an "S". 5 records are found to exist with the letter S in

Example of EXTRACT Command

```
ACTION
:        SELECT
SELECTION RULES
:        Vs EXISTS AND Morphology EXISTS
:        AND Cluster CONTAINS Virgo END
____________________________
   23 RECORDS SELECTED

ACTION
:        EXTRACT Morphology END

____________________________
   23 RECORDS SELECTED


______________________________________
 FIELD    8 TAKES    23 VALUES.

  pec.  Sb    Sb    Sb    E0    SB    Sb    E0    E5    Sc
  Sb    E1    SBc   Sb    SBc   S0    E6    Sb    SBc   S0
  Sb    1     E0
```

Exhibit A

field 1. Field 10 for these 5 records is now tested for values less than 999. One record is found. Note that the RETAIN command need only be exercised once to successively RETAIN following SELECTed records. It serves essentially to define a "filter" over the file while giving the user an interactive browsing facility.

4.4 The DISPLAY Command

This command is used when the user wishes to type out the information obtained by the previous SELECT command. The user writes

```
        DISPLAY(List of field names or numbers) END
   or   DISPLAY ALL
   also DISPLAY NUMBER
        DISPLAY LIST
        DISPLAY (Record number)
```

(Note Exhibit B)

In some cases, however, the listing of the information in this form is not practical, either because it is too long, or because several copies are needed or because the extraction done through DIRAC is only one step in a more complicated editing task. To solve this problem the user writes

```
        DISPLAY WYLBUR (List of fields) END
   or   DISPLAY WYLBUR ALL
```

(Note Exhibit C)

4.5 The RELEASE Command

The RELEASE command allows the user to address his SELECTion rules to the whole file again after working under the RETAIN command for a while.

EXAMPLE:

```
ACTION
:       SELECT
SELECTION RULES
:       @11 CONTAINS Virgo END

24 RECORDS SELECTED

ACTION
:       RETAIN

ACTION
:       @1 CONTAINS S END
```

DIRAC COMMANDS

```
ACTION
:        RETAIN

ACTION
:        Morphology DOES NOT CONTAIN Sb END

    14 RECORDS SELECTED

ACTION
:        Vs (>1000 AND <= 1500) END

     3 RECORDS SELECTED

ACTION
:        DISPLAY SN Vs CLUSTER Morphology END

   18
SN                  1919a
Vs                  1261
Cluster             Virgo
Morphology          EO

   89
SN                  1960f
Vs                  1240
Cluster             Virgo
Morphology          SBc

  246
SN                  s1922 alpha
Vs                  1243
Cluster             Virgo
Morphology          EO

    3 RECORDS SELECTED
ACTION
:        DISPLAY WYLBUR SN Vs Cluster Morphology
:        END

    3 RECORDS SELECTED
```

Exhibit B

WYLBUR DATA SET

```
?list
  0.001
  0.002  SN                1919a
  0.003  Vs                1261
  0.004  Cluster           Virgo
  0.005  Morphology        EO
  0.006
  0.007  SN                1960f
  0.008  Vs                1240
  0.009  Cluster           Virgo
  0.01   Morphology        SBc
  0.011
  0.012  SN                s1922 alpha
  0.013  Vs                1243
  0.014  Cluster           Virgo
  0.015  Morphology        EO
```

Exhibit C

1 RECORD SELECTED

ACTION
:   RELEASE

ACTION
:   @1 CONTAINS S END

65 RECORDS SELECTED

There are 65 records in this file where field 1 CONTAINS the letter S, but only one such record was found among these records where field 11 contained the word "Virgo". The user typed the command RELEASE to reinitialize the search to the entire file.

5. OPERATION OF DIRAC

The following examples demonstrate the four execution modes of DIRAC. The user should note how each mode is initiated. DIRAC allows the user to exit from an execution mode either by initiating a new mode -- responding to a prompt from DIRAC -- or by typing the word "END".

5.1 CREATE Mode

```
? use diracl clear load
1 UNRESOLVED REFERENCES

? enter

                    D I R A C   V E R S I O N   1

NAME OF USER
:        Smith
PLEASE TYPE EXECUTION MODE
:        CREATE
FILE IDENTIFICATION
:        L020
                        CUMUL.TERMINAL TIME :  0.42 MIN
                             CUMUL.CPU TIME :  0.10 MIN

KEY FOR THIS MODE
:        Q
FILE NAME
:        Supernova
FILE "DESCRIPTION"
:        "Preliminary Catalogue of Supernovae"
DISPOSITION (PUBLIC/PRIVATE)
:        PRIVATE
TYPE "LIST OF QUERY USERS"
:        "Smith Jones Johnson"
GIVE NOTATION FOR RECORD AND FIELD
LEFT RECORD NUMBER DELIMITER
:        $
```

```
RIGHT RECORD NUMBER DELIMITER
:       $
LEFT FIELD NUMBER DELIMITER
:       @
RIGHT FIELD NUMBER DELIMITER
:       NONE
RECORD LENGTH
:       256
SUPPLY NAME AND "DESCRIPTION" OF ALL FIELDS
@1?
:       SN "Supernova Number"
@2?
:       zI "Zwicky I System"
@3?
        .
        .
        .
:       NONE
SUPPLY DATA TYPE AND MULTIPLICITY
:       ALPHA SINGLE @1 @2 @3 @4 @9 @11 @25 @26 .......
:       INTEGER SINGLE @6 @7 @8 @36 @37 .......
:       ALPHA MULTIPLE @5 @21 ......
:       INTEGER MULTIPLE @22 @23 ......
:       REAL SINGLE @39 @40 .......
:       REAL MULTIPLE @15 @16 .......
DEFINE "RECORD LOCATOR"
:       ""
DEFINE RECORD STRUCTURE
:       NONE
VALIDATION SPECIFICATIONS
:       @1 NECESSARY
:       @3 NECESSARY
:       @5 NECESSARY
:      NONE
THE FILE HAS NOW BEEN CREATED


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

5.2 UPDATE Mode

The UPDATE mode is utilized to fill a newly created file with information or to alter the contents of a previously updated file. The user should remember that during the CREATE mode an 'empty' file was created, and that during the UPDATE mode that file's contents are either supplied or altered.

```
? use diracl clear load
1 UNRESOLVED REFERENCES

? enter

                 D I R A C   V E R S I O N   1

NAME OF USER
:       Smith
PLEASE TYPE EXECUTION MODE
:       UPDATE
FILE IDENTIFICATION
:       L020
                 CUMUL.TERMINAL TIME :  41.18 MIN
                      CUMUL.CPU TIME :   0.26 MIN

SPECIAL INPUT INTERFACE  ?
:       ¢***                (press ¢, then attn key)
DO YOU WANT YOUR PROGRAM? no
SESSION BREAK, ATTENTION AT 71C240
? use Supernova
? CONTINUE
INCORRECT STATEMENT.  PLEASE RETYPE  : UPDA...
:       WYLBUR
UPDATE COMPLETED ; MAX.RECORD LENGTH =   140
THE FILE CONTAINS     10 RECORDS


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

The above UPDATE procedure could also be simplified by the following procedure:

```
? use diracl clear load
1 UNRESOLVED REFERENCES
use Supernova
enter
.
.
.
```

This eliminates the procedure of breaking out of <u>DIRAC</u> control in order to fetch the Supernova records for input into the file. It eliminates the statements between "SPECIAL INPUT INTERFACE ?" and "WYLBUR" in the first example of the UPDATE mode.

5.3 QUERY Mode

The QUERY execution mode has been sufficiently examined in Section 4 so that no further example will be given here at this time.

5.4 STATUS Mode

The user answers the prompt:

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

or

PLEASE TYPE EXECUTION MODE

with the word STATUS. He then receives the following information (Exhibit D). This status report is taken from the Supernova Catalogue.

STANFORD UNIVERSITY
COMPUTATION CENTER

STATUS REPORT FOR FILE A010
23/JAN/1970

LANGUAGE: DIRAC1

DESCRIPTION : Preliminary Catalogue of Supernovae

CREATION DATE : 23/JAN/1970
RECORD NOTATION : N
FIELD NOTATION : @F
DISPOSITION : PUBLIC
NO. OF RECORDS : 259

1 = TYPE
2 = MULTIPLICITY
3 = INDEXING
4 = CODE RESIDENCE
5 = CODE TYPE

FILE NAME : Supernova
FILE CREATED BY : Vallee
RECORD LENGTH : 1024
NO.OF FIELDS : 23
LATEST UPDATE ON : 23/JAN/1970

F I E L D I D E N T I F I C A T I O N , S T A T I S T I C S A N D V A L I D A T I O N I N F O R M A T I O N

FLD	NAME	DESCRIPTION	STORAGE 1	2	3	4	5	VALIDATIONS NEC.	SIZE	SUB.	STATISTICS SIZE	DEC.	SUB.	EXISTENCE REC.	PCT
1	SN	Supernova Number	A	S	0			YES	35	0	11	0	0	259	100.00%
2	zI	Zwicky I System	A	S	0				0	0	3	0	0	108	41.70%
3	zII	Zwicky II System	A	S	0				0	0	3	0	0	204	78.76%
4	Designation	Other Designation	A	S	0				0	0	70	0	0	47	18.15%
5	Galaxy	PArent Galaxy	A	S	0				0	0	79	0	0	157	60.62%
6	Alpha	Right Ascension 1950.0	A	S	0				0	0	7	0	0	234	90.35%
7	Delta	Declination 1950.0	A	S	0				0	0	6	0	0	235	90.73%
8	Morphology	Morphology of Parent	A	S	0				0	0	8	0	0	241	93.05%
9	Mp	Photographic magnitude of Parent	A	S	0				0	0	11	0	0	224	86.49%
10	Vs	Recession Velocity in km /s	I	S	0				0	0	64	0	0	107	41.31%
11	Cluster	Cluster Membership of parent	A	S	0				0	0	13	0	0	92	35.52%
12	l2	Galactic longitude	A	S	0				20	0	9	0	0	258	99.61%
13	b2	Galactic latitude	A	S	0				20	0	7	0	0	257	99.23%
14	maxdate	Date of Maximum	A	S	0				0	0	14	0	0	258	99.61%
15	Discovery	Date of discovery	A	S	0				0	0	4	0	0	255	98.46%
16	Magnitude	Maximum photographic magnitude	A	S	0				0	0	10	0	0	245	94.59%
17	Position	Position in galaxy	A	S	0				0	0	27	0	0	240	92.66%
18	Type	Type of supernova	A	S	0				0	0	10	0	0	101	39.00%
19	Discoverer	Name of discoverer	A	S	0				0	0	72	0	0	243	93.82%
20	Remarks	Remarks	A	S	0				0	0	71	0	0	45	17.37%
21	Authors	Authors	A	M	0				100	0	0	0	0	0	0.0 %
22	Sources	Bibliographic references	A	M	0				0	150	0	0	0	0	0.0 %
23	Information	Information in article	A	M	0				0	0	0	0	0	0	0.0 %