Discussed is an experimental, specialized data-base system developed for users who do not require the sophisticated resources of the large data-base systems but do require many of the capabilities they provide. Data manipulation and retrieval within a data base are made available for the nonprogrammer user. The function and design attributes of the system are described including the reasons for basing the system on APL functions.

A user-oriented data-base retrieval system

by A. U. Jones

Potential users of data-base systems range from those with small quantities of tabular data that is organized in a simple manner to those with large amounts of highly organized, complexly structured data and related data files. The current and available data-base systems tend to be oriented more to the user with requirements for complex systems rather than for simple ones. Those users with the lesser requirements in terms of system sophistication, while still requiring many of the capabilities of a larger system, need them on a smaller, more immediate, and direct scale. They require the capabilities without having to face such complexities as data description languages, file definition requirements, space utilization and allocation considerations, among others.

To these users, the data-base definition and manipulation capabilities of the system are more important than the structure and efficiency of the underlying techniques with which the system organizes and manages the data base. They do not usually have the time or the expertise to undertake the task of subsetting, modifying, or adapting the more general-purpose data-base systems for their specific applications. As a result, they are often faced with the necessity of designing their own systems in order to achieve the required simplicity, flexibility, and applicability. It was for this type of user, who needs a general-purpose system on a smaller, more user-oriented scale, that a specialized system called the Data Base Retrieval System was created.

The Data Base Retrieval System (DBRS) is an experimental, interactive system[1] that provides facilities for using a terminal to maintain information in a data base and to generate reports from it. DBRS is comprised of APL (A Programming Language) functions, and is designed to be used by people who have little knowledge of APL.[2] DBRS offers a straightforward, command-oriented interface that appears to the user to be tailored specifically to each unique data base that is established. Major facilities include: data-base definition; data entry, update, and listing; arithmetic computation; selective retrieval; and formatted report generation. This paper describes DBRS and the rationale for various design decisions and implementation considerations that occurred during its development. In addition, the paper discusses some of the benefits as well as limitations of using APL[3,4] as the base system and implementation language, and presents, from the designer's viewpoint, the characteristics of a user-oriented data-base system. The appendix contains a sample terminal session illustrating the primary DBRS functions and should be referred to when it may be necessary to clarify any questions on the actual facilities provided by the system.

System objectives and user functional requirements

DBRS was originally conceived as a system that would provide logical search and selection capabilities for data maintained in an interactive, on-line terminal system environment. It was intended to be a generalized tool, although developed as a specialized data-base system, that would replace any similar existing systems, or would eliminate the need to create additional special-purpose, application-oriented systems. It was expected that many of the users would be nonprogrammers. Therefore, DBRS was designed to provide the following functional capabilities:

- Establish the format of the data and the size and structure of the data base.
- Add and delete data and modify data-base content.
- Retrieve data on a selective basis and display the results.
- Generate reports utilizing user-specified formats.
- Provide facilities for data analysis and computation.

The manner in which these capabilities were to be presented to the user was also bounded by the following usability objectives:

- Completely interactive with all facilities available at the terminals.
- Simple and easy to use, including easy-to-remember command syntax with standard defaults and suitable error messages.
- Not oriented or limited to specific types of applications.

- A minimal investment in terms of system installation, education, and implementation effort.

General design considerations

The overall DBRS design philosophy was to provide a system to help complete a job in a manner suitable to the user. This approach is preferred over one that forces the user to do the job in the manner determined by the system. A very important consideration in the design was the desire to provide a user interface that would be suitable for nondata-processing users, including secretaries and administrative personnel, as well as programmers and APL users.

implementation language

APL was chosen as the implementation language because of its interactive capabilities and the ease with which commands can be defined by the implementor and expressed by the user in an "English-sentence-like" fashion.

User communication with the system is based on words and phrases that appear to be directions and commands for data-base manipulation. In reality, these statements specify APL function calls and related parameters. The obvious advantage is that the user need not know the APL language nor be a programmer to use the system. Nevertheless, the more experienced user can take advantage of APL knowledge to extend the basic DBRS functions or add additional functions as appropriate to the needs of specific applications. Thus, the system can be a useful tool for both the APL expert and the less-experienced terminal user, without appearing to be too simple or too complex for either. This duality is accomplished, in general, by providing a comprehensive set of commands (APL functions) that allow the user to maintain complete control over the data base without realizing that APL is being used.

In contrast, the experienced APL user might find it more expedient to use APL facilities directly for some of the data-base manipulation; the user is not prevented nor constrained by DBRS from so doing. The ease with which this user-oriented capability was accomplished reflects well on both the structure and design of APL and the implementation of DBRS.

The data-base environment

One of the characteristics that distinguishes DBRS from other data-base systems is the maintenance of data in a form directly

representative of the manner in which the user defined the data base. While there is nothing inherent in the DBRS interface nor in APL that would prevent a more complex data storage scheme, the choice was made to use the most simple available data structure. Had this not been done, it would have been much more difficult for the user who is familiar with APL to access the data directly or to expand on DBRS capabilities with additional functions. It would also have made the implementation of DBRS itself much more complex. A more sophisticated data structure would probably have necessitated a secondary level of functions within DBRS that did the actual data-base management. In its current format, all DBRS functions reference the data directly; there are no "pointers" or definition items contained within the data.

data-base structure

The data-base structure supported by DBRS is a two-dimensional array (or table) containing character data. A single user-defined format consisting of fixed-length fields is imposed over each row of the array through the use of global control variables constructed when the user defines the characteristics of the data. A field is merely a set of contiguous columns and may contain any data chosen by the user. All data is maintained in character form; therefore, the user does not have to distinguish between numeric and character (or alphanumeric) data—a number is always a number, at least as the user views it. This choice of data format was not necessarily the most efficient in terms of storage usage (numeric data can usually be expressed in less space) or implementation strategy (DBRS must convert to internal numeric form for arithmetic processing). However, it was decided that this format was better for DBRS because (1) the user's data can be kept in a single array, (2) the user does not have to explicitly declare the format of the fields, and (3) the user is not restricted to the type of data that can be put into any field.

The decision to restrict data bases to fixed-length fields was made in the interests of keeping both the user interface and the design of DBRS straightforward and consistent. Hierarchical structures or fields with multiple elements can be handled by entering the same item repeatedly, with appropriate variations in field content. Variable-length, text-like fields can be developed by defining a field for the maximum length of data expected, or by having a field in the data base that contains a note or reference to text that is kept elsewhere. Thus, what might appear to be a lack of facilities or a limitation of the system can be overcome in various ways without compromising the objectives of DBRS. Special functions for handling fields with variable content and of varying length (data storage and text-processing functions) are not necessary; the existing functions can be applied equally to all fields.

The data-base command language

Design of the DBRS command syntax was intended to be as "natural" as possible from the user's viewpoint. In the case of one-word commands (with or without parameters), the word is always spelled out. For example, PRINT is used instead of an abbreviation like PRNT. Commands requiring two parameters are structured with extra functions so that the resulting phrase appears to be an imperative command rather than an APL function with left and right arguments. Examples of this are:

```
FIND COST EQUAL 10      (rather than COST FIND 10)
MOVE 25 AFTER 3         (rather than 25 MOVE 3)
```

prompting

In cases where more than two parameters are required, or where the command might be repeated with some of the parameters changed, the functions are designed to "prompt" for input. It is tempting from the designer's standpoint to have only one function doing all the work of some area, i.e., data-base modification, and to let the user's replies determine the specific work to be done, thereby "tailoring" the function to each request made. However, replying to multiple prompts can be tedious from the user's standpoint, particularly if the command involves a fairly straightforward operation. The prompts merely move the user to the proper part of the function, possibly after executing some portion common to all operations of the function. Therefore, it was decided to provide different functions in DBRS for each specific operation, repeating code or using common subroutines as necessary. The system also accomplishes the prompting in an order that appears most obvious to the user, rather than in a way that was easiest for the implementor.

In all cases where inputs are requested, standard default values or actions are predefined. Usually, all the user has to do is reply with a carriage return to indicate use of the default value, or have no action or change take place. Functions that actually change the data base are designed to allow display of the current content of the data base before changes are made. This "second chance" is especially important when the user requests that portions of the data base be deleted, since there is no way to recover data if the request was in error (unless the user has saved a backup copy).

Data-base definition and manipulation facilities

One of the primary purposes of DBRS is to provide facilities for the user to manage data in an APL work space by using user-oriented commands rather than APL expressions. The first step in this process occurs during the user's definition of the data-base structure.

field name

After specifying the number of fields in the data base and the maximum size to be allowed for each one, the user is asked to supply a name for each field. It is up to the user to select appropriate names that have meaning related to the content of the data or reflect the way the data is visualized. These names are used directly in all DBRS commands to reference fields (sets of columns) within the data base. DBRS converts each field name to an APL function that produces, as an explicit result, a number representing the relative order of this field within the set of fields defined for the data base. The "field number" is used by DBRS functions as an index to other tables of control information about the data base, such as the table containing the character representation of the field name. The field number is used primarily, along with other control information, to generate a vector of column numbers that becomes the second subscript of any reference to the data-base array. Entries in the data base are referenced by their line/row number (the first subscript of the array reference), which is always the relative position of the entry within the data-base array. Thus, the user can address the data base both "horizontally" by entry, and/or "vertically" by defined field name, depending on the desired data-base subset.

In addition to data entry and deletion capabilities, DBRS also provides functions that allow data modification in specific fields in specific lines, as well as multiple fields in multiple lines of the data base. Since all modification functions (replace, duplicate, and text edit) ask the user to specify the field names and entry/line number to be changed, and since all data is maintained in character form, only one set of functions is needed for data-base maintenance. As a result, the DBRS functions (commands) appear to be tailored to the user's own set of field names, when, in fact, they are for general-purpose use from the viewpoint of the implementor as well as the user.

Data retrieval and arithmetic computations

The use of field names as a means of "customizing" commands for the user was an important consideration in the design of both the retrieval and computational facilities. Retrieval, in the case of DBRS, means the ability to address, or select, portions of the data base as a result of searching for specific values in the data. Logical operations such as "equal," "less than," and "greater than" form the basis for these selections, which result from comparisons of data-base field content to literal values specified in the command. Additional operators allow searching for specific character strings within a field. The general command format is:

FIND fieldname operator 'literal value'

Examples of actual use are:

```
FIND UNITPRICE LT '29' or FIND SUBJECT CONTAINS 'SYSTEM'
```

In reality, the actual search of the data base is done by the "operator" function (LT or CONTAINS in the example), and the field name and literal value are its left and right arguments. The FIND function merely converts the binary vector that is the explicit output of the operator into a numeric vector containing a list of line numbers that met the specified selection criteria.

A further application of this same procedure allows the user to combine search criteria using the logical connectives AND and OR. An example is:

```
FIND (NAME EQ 'SMITH') AND CLASS EQ 'SOPH'
```

which would find all sophomores with the name of Smith in a student data base. In this case, the AND function expects binary vectors as input and produces a single binary vector as output, which is what the FIND function requires as input. Note that parentheses are required in order to cause APL to execute the functions in the desired manner.

arithmetic computation

The commands to execute the arithmetic computation facilities provided by DBRS are structured in nearly the same manner as the commands for retrieval. The full statement format is:

```
COMPUTE fieldname AS fieldname operation fieldname FOR lines
```

A specific example is:

```
COMPUTE COST AS PRICE TIMES QUANTITY FOR ALL
```

which generates entries in the COST field for ALL lines in the data base by multiplying the value in the PRICE field by the value in the QUANTITY fields. Since the "operation" functions are designed to allow literal values as well as field names as arguments, it is possible to perform multiple operations within the same command. For example:

```
COMPUTE C AS A PLUS (D MINUS ('3.5' TIMES F)) FOR LIST
```

will insert a computed value [A + (D − 3.5* F)] in field C for all the lines contained in the vector LIST. The FOR, AS, and COMPUTE functions perform special operations related to selecting entries and storing results in the data base, setting switches and global variables as required. In fact, the purpose of COMPUTE is primarily to provide a verb for the command, although it does do some cleanup work at the end of the computation sequence. In any case, the overall effect is to provide the user a customized command for doing arithmetic operations. The necessary indexing and character/numeric conversion are done automatically.

Data display

Since data is in character form and is stored line by line as rows of an array, it would be easy enough for the user to display the content of the data base simply by typing the array name, using subscripts if desired. However, listing the data with spacing between fields, or generating a formatted report using only basic APL array referencing, is quite a bit more complex. Therefore, DBRS provides two means of displaying data-base content. The PRINT function produces a simple listing of the lines that the user specifies, printing the fields in the order they appear in the data base, with a single space between fields. This spacing is easily achieved because the data base is allocated by DBRS with one more column than the user defines. Since it always contains a blank, spacing can be inserted in a printed line by using a vector as the second subscript of the array reference and including the number of the extra column whenever a space is required. No matter how the user has defined the fields, formatting (spacing) can be achieved without additional complexity in the function implementation.

Facilities are also provided by DBRS that allow the user to print selective fields, in any order desired, with variable spacing between them. As in printing, the actual formatting is achieved by constructing a vector referencing the data-base columns (including the blank column for spacing purposes) in the desired order. The REPORT function also allows the user to specify titles, column headings, and page sizes, as well as providing a facility for saving these definitions and referencing them when needed. In effect, by using combinations of the FIND, COMPUTE, and/or REPORT functions, the user can create a "program" within the confines of the data-base system.

Development experience—the APL environment

Preceding sections of this paper have suggested that APL provides a favorable environment in which to develop a user-oriented interactive system. The primary advantage was in permitting an interface to be developed that did not require the user to become knowledgeable about APL. At the same time, the DBRS designers were able to use all of the capabilities and features of APL, with the need to conserve space being the only major restriction, since all data and functions reside in the user's work space. For example, in order to avoid "WS FULL" situations, the logical search functions check the amount of working area available and do element-by-element comparisons rather than using the more appropriate inner product array operations if sufficient space is not available.

APL capability

Users with APL expertise can easily modify DBRS functions to suit their own needs or write additional functions within the framework of DBRS, since there is only one level of functions to deal with. All DBRS functions are designed to be executed directly, and any function can reference the data base and use the global control variables; there is no higher-level monitor or lower-level access method surrounding the functions.

This type of design does have some drawbacks. Perhaps the most significant drawback is that each function, or combination of functions such as the COMPUTE sequence, is independent of the others and is expected to reach normal completion each time it is executed. If a function should inadvertently be interrupted, the data base might be left in an undesirable state since it is not possible in APL to have a supervisor-like program to intercept control on error situations or when functions do not complete normally. For example, the occurrence of an interruption while lines are being moved around within the data base could cause some lines to be lost entirely if they were stored in temporary variables during the move operation. While it would have been possible to include additional checking or a special function to be executed when an interruption occurs, it was decided not to do this since the user will know the interruption occurred, and will have to take corrective action, if only to clear the State Indicator.

Actually, users need some minimal knowledge of APL, since system commands such as)LOAD and)SAVE must be executed directly by the user. Also, since lack of working area can become a problem if the data base has been defined near the maximum calculated by DBRS, knowledge of)ERASE and)COPY commands is helpful. DBRS function groups are set up to be deleted if space is needed and are copied back in when required. It is the user's responsibility to monitor the work-space area, since APL does not permit system commands to be executed within functions.

Experience with the system indicates that most users tend to use DBRS "as is" even when they have some knowledge of or experience in writing APL functions. The most common change, if any, has been the implementation of user-defined functions that combine selection (FIND commands) and reporting in a single function with predefined search criteria. There have been instances, however, where DBRS has been combined with other systems either as a "front-end" processor for ease of data entry, or as an output processor because of the capabilities for selection and formatted report generation. Since the system was designed to be open-ended and modifiable, such uses have been encouraged.

Summary

This paper has described the function and design attributes of a user-oriented data manipulation and retrieval system intended for the nonprogrammer user. The acceptance of the system, by those who have used it, is sufficient to demonstrate its feasibility to maintain small-to-moderate-size data bases for users who do not require the resources of larger data-base systems for routine tasks. The considerations and techniques given here relate specifically to DBRS, but should be applicable in principle to similar systems. The prime advantages of this approach have been:

- Ability of the user to use the system without need for assistance from system or programmer personnel.
- Immediate, on-line data-base definition, data entry, and reporting capabilities.
- Ease of change and new function definition within the system, both for the users and the designers.

The following guidelines were found to be especially helpful and are suggested as considerations for designers of user-oriented systems:

1. Try to envision the user's perspective when designing or modifying systems—the design that is easiest to implement will probably not be the easiest to use.
2. Try it yourself—what appears to be a superior design on paper doesn't always turn out that way when you try to apply it to a real-life problem.

ACKNOWLEDGMENTS

The author acknowledges the contributions of C. E. Mahood of the Endicott laboratory of the System Products Division as co-developer, with the author, of the Data Base Retrieval System described in this article. Also acknowledged is the management support and direction provided by J. W. Owens.

CITED REFERENCES AND FOOTNOTE

1. DBRS is a system developed by IBM for internal use and is not available for distribution outside of IBM.
2. C. E. Mahood, "Data Base Retrieval System (DBRS), A personalized data base system for the APL user," *APL 76*, Ottawa, Canada (September 20–22, 1976).
3. *APL Language*, No. GC26-3847, IBM Corporation, Data Processing Division, White Plains, New York.
4. *APL Shared Variables User's Guide*, No. SH20-1460, IBM Corporation, Data Processing Division, White Plains, New York.

Appendix: Example of a DBRS terminal session

Initialization

```
      NEWΔΔ
ENTER, IN SEQUENTIAL ORDER, THE MAXIMUM NUMBER OF CHARACTERS FOR EACH FIELD.
  (TYPE RESPONSE ON ONE LINE WITH A BLANK BETWEEN EACH ENTRY)
□:
      12 8 8 20 4 6 8
THIS DATA BASE WILL HAVE 66 CHARACTERS PER ITEM AND CAN HAVE A MAXIMUM
OF 564 ITEMS OF 7 FIELDS EACH. ENTER DESIRED MAXIMUM NUMBER OF ITEMS.
□:
      100
ENTER THE NAME OF EACH FIELD IN SEQUENTIAL ORDER,
ONE NAME PER LINE FOLLOWED BY A CARRIAGE RETURN(CR)
 (NAMES USED MUST BE UNIQUE WITHIN THE WORKSPACE)
NAME
PHONE
DATE
STATUS
AMT1
AMT2
BALANCE
INITIALIZATION COMPLETE
```

Data entry

```
      ENTER
FIELD NAMES:

NNNNNNNNNNNNPPPPPPPPDDDDDDDDSSSSSSSSSSSSSSSSSSSSAAAA******BBBBBBBB
MAHOOD,C.E. 789-342210/23/75ON-TIME             4.22 43.12
RUSH,A.E.   788-009812/13/74LATE                 .98  1.01
JONES,E.E.  555-000101/12/76                    4.56


      USED
3 ITEMS OUT OF 100



      PROMPT
FIELD NAMES:
NAME
DATE

NAME   :ARBIT,E.R.
DATE   :11/01/74
NAME   :BALL,S.
DATE   :
NAME   :


      INSERT
AFTER WHICH LINE?0
FIELD NAMES:

NNNNNNNNNNNNPPPPPPPPDDDDDDDDSSSSSSSSSSSSSSSSSSSSAAAA******BBBBBBBB
JONES,A.U.  888-098701/01/76OK
OPALD,S.                    OK                  5.44

DONE


      PRINT ALL
ITEM NAME         PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
1    JONES,A.U.   888-0987 01/01/76 OK
2    OPALD,S.                      OK                  5.44
3    MAHOOD,C.E.  +789-342 210/23/7 5ON-TIME             4.2 2 43.1 2
4    RUSH,A.E.    788-0098 12/13/74 LATE                .98   1.01
5    JONES,E.E.   555-0001 01/12/76                     4.56
6    ARBIT,E.R.            11/01/74
7    BALL,S.
```

```
      REENTER
STARTING WITH WHICH LINE?3
FIELD NAMES:

NNNNNNNNNNNPPPPPPPPDDDDDDDDSSSSSSSSSSSSSSSSSSSSAAAA******BBBBBBBB
MAHOOD,C.E. 789-342210/23/75ON-TIME             4.22 43.12


      PRINT 3 4
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
3    MAHOOD,C.E.   789-3422 10/23/75 ON-TIME             4.22  43.12
4    RUSH,A.E.     788-0098 12/13/74 LATE                .98   1.01


      MOD
LINE:7
FIELD:DATE
IS:
DATE   :09/09/74
FIELD:AMT1
IS:
AMT1   :5.00
FIELD:
LINE:


      PRINT 7
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
7    BALL,S.                09/09/74                     5.00


      REPLACE
LINES:2 3
FIELD NAMES:
STATUS
AMT2

SSSSSSSSSSSSSSSSSSAAAAAA
ITEM NO.2
OK
NOW LATE              -2.33
ITEM NO.3
ON-TIME               43.12
ALSO LATE             44.56
DONE
LINES:


      PRINT 2 3
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
2    OPALD,S.                        NOW LATE            5.44  -2.33
3    MAHOOD,C.E.   789-3422 10/23/75 ALSO LATE           4.22  44.56


      EDIT
LINES:6
FIELD:NAME
ITEM NO. 6 FIELD IS:ARBIT,E.R.
OLD PHRASE:AR
NEW PHRASE:BAR
DONE
LINES:


      PRINT 6
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
6    BARBIT,E.R.            11/01/74
```

Search and retrieval

```
      FIND AMT1 LE '3.22'
3 ITEMS


      PRINT LIST
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
1    JONES,A.U.    888-0987 01/01/76 OK
4    RUSH,A.E.     788-0098 12/13/74 LATE                 .98  1.01
6    BARBIT,E.R.            11/01/74


      FIND (NAME BEGINS 'J') AND AMT1 LE '0.44'
1 ITEMS


      PRINT LIST
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
1    JONES,A.U.    888-0987 01/01/76 OK

      PROMPT
FIELD NAMES:
NAME

NAME   :**TOTAL
NAME   :
```

Computations

```
      COMPUTE  LASTENTRY AS SUM AMT1, AMT2 FOR 1 THRU LASTENTRY-1

      COMPUTE BALANCE AS AMT1 PLUS AMT2 FOR ALL

      PRINT ALL
ITEM NAME          PHONE    DATE     STATUS              AMT1 AMT2   BALANCE
1    JONES,A.U.    888-0987 01/01/76 OK                                  .00
2    OPALD,S.               NOW LATE                    5.44 -2.33     3.11
3    MAHOOD,C.E.   789-3422 10/23/75 ALSO LATE          4.22 44.56    48.78
4    RUSH,A.E.     788-0098 12/13/74 LATE                .98  1.01     1.99
5    JONES,E.E.    555-0001 01/12/76                    4.56           4.56
6    BARBIT,E.R.            11/01/74                                   .00
7    BALL,S.                09/09/74                    5.00           5.00
8    **TOTAL                                            20.2  43.24   63.44
```

Defined reports

```
      DEFREPORT
TITLES CAN BE UP TO 120 CHARS.
REPORT NO. 1
TITLE:SAMPLE BALANCE SHEET
HEADER LINE 1
NAME OF USER  NORMAL  EXTRA    BALANCE     DATE
HEADER LINE 2
              HOURS   HOURS    IN TOTAL
FIELD NAMES:
NAME
AMT1
AMT2
BALANCE
DATE

FIELD SPACING:3 2 4
REPORT NO. 2
TITLE:
```

```
     REPORT 1
ENTER LINE NOS.:ALL
PRINT LINE NOS.?N
NO. OF LINES /PAGE?56
POSITION PAPER, THEN HIT CR



SAMPLE BALANCE SHEET          15.41.53 07/20/76  PAGE 1

NAME OF USER   NORMAL   EXTRA     BALANCE      DATE
               HOURS    HOURS     IN TOTAL

JONES,A.U.                            .00    01/01/76
OPALD,S.         5.44   -2.33        3.11
MAHOOD,C.E.      4.22   44.56       48.78    10/23/75
RUSH,A.E.         .98    1.01        1.99    12/13/74
JONES,E.E.       4.56                4.56    01/12/76
BARBIT,E.R.                           .00    11/01/74
BALL,S.          5.00                5.00    09/09/74
**TOTAL          20.2   43.24       63.44
```

Reprint Order No. G321-5042.