Development and Optimization of Data-Base System for Welding Research†

Katsunori INOUE* and Hideo TAKECHI**

Abstract

The operation of data-base has been made possible by the development of sophisticated softwares which in turn were the results of the development of smaller and faster memory units. Only about ten years has passed since the notion of data-base, but it has revolutionized the way we look at data. The present study, a trial of creating and using data-base for engineering data analysis, has demonstrated the direction to be taken in the future in the field of engineering data-base management. A FORTRAN application through DML (Data Manipulation Language) is also carried out. The ease and versatility in using the data-base, as demonstrated by these advanced application techniques, suggests a proper direction to be taken hereafter in this field. The end-user support utility prepared by NEC and installed in ACOS-6/SYSTEM 1000 called INQ (Information Query), is used. Details of INQ for data-base operations are explained here and demonstrated with examples, of which source data in this case, is the Transactions of JWRI.

KEY WORDS: (Data-Base) (Information Query) (Data Manipulation Language) (File Description Language) (Integrated Data Store)

1. Introduction

The notion of data-base has been introduced and developed for the past ten years in line with the development of smaller and faster memory units, which in turn has given us access to more and more sophisticated software. There are quite a number of Data-Base Management Systems commercially available throughout the nation. They are mainly used for information retrieval in scientific research and business information processings.

The greatest merit of these systems is that once a data-base file is prepared in the computer system and provided telex access to the computer is obtainable, one can use the data in the file in various ways without having much knowledge of computer programming[1]. What is more, simple commands to the computer can do the various operations, which were previously performed only through supplying the computer with a series of complicated instructions written in the programming language such as FORTRAN or COBOL. Before the introduction of data-base management systems, the data were prepared by a programmer who was well-versed only in the programming languages for a particular job he had in mind. In other words, the data had to be accordance with a specific format which would fit the program prepared by the programmer. The programmer, or the program, determined the kind of data to be prepared. What is more, only the programmer could use and manipulate the data. But now, thanks to the data-base management system, a stock of data can be used and manipulated for different purposes on different occasions by different users. Even users who are not familiar with programming languages can have access to the data and use them, provided that they know simple commands for data search.

As mentioned above, there are many data-base systems available throughout the nation. Therefore it seems that it is no longer necessary for us to build our own system, saving a great deal of our labor. Most of the data-base services around us are normally supplied by computer manufacturers that provide us with manuals explaining how to use it. These commercial data-base are operated and updated regularly by professional engineers at computer centers and they are fairly reliable at any time for almost all users.

The ACOS-6/SYSTEM 1000 prepares the end user support utility for developing his own data-base.

† Received on October 21, 1983

* Professor

** Co-operative Researcher (Research Associate, Anan Technical College)

Transactions of JWRI is published by Welding Research Institute of Osaka University, Ibaraki, Osaka 567, Japan

2. Data Set Texture and Organization

The fragments of data set kept in data-base are consisting from the indexes of the Transactions of JWRI. Their titles, authors, keywords and abstracts are the key items set forth to identify each record. By specifying a value to these key items, the search goes through the key items and extracts the records having the same at specified key field. The design of FD (File Description) is to prepare a catalogue for the data set, in this case which is of the indexes and the abstract of the Transactions of JWRI. The structure of the catalogue has a simple tree top configuration and a model was invented. **Figure 1** is showing a schematic diagram of record structure in

RECORD	
INDEXES	
LEVEL ONE	REGISTRATION NO. / CLASSIFICATION CODE
LEVEL TWO	TITLE/AUTHER/JOURNAL/KEYWORD/ABSTRACT
LEVEL THREE	PHRASE
LEVEL FOUR	WORD

Fig. 1 A Simulated File Description for Data-Base Record

data-base file. An alpha-numeric code is assigned to the first root item, which corresponds to the primary key of INQ system. This code represents the record and is absolutely unique among the data set. The second key item is of an attribute made up of strings of words. It can be such as the title of a journal. These two root items are reckoned as the privileged indexes of record.

The first level of the record texture is provided solely for the use of librarians or for those who administrates the data-base. The two items consist of this level one. The first item is the registration number, i.e. a unique and serial number given to each record, and with which one can locate the required record on selves or in store. The librarian may be able to put in coded information to this ten digits, i.e. such as the year of their receipt, storage address where the record is kept and so on. The second item at level one is of the classification, which is also a coded alpha-numeric derived from the category belonging. This code of classification could have many abbreviated form of information associated with record. The string of this ID code have a maximum of 54 bytes and it is not normally identical to UDC. The coding for classification is preferred to be universal with other data-base systems, but so far any generalized coding convention is not available. Therefore the actual practice of coding for ID shall be confined reluctantly within our system.

The two items of level one are exclusively for the use of librarian and not intended for that of ordinary users. These two coded items are deviced especially for those who need quick reference and fast identification of record. As mentioned earlier, two of these items are unique among the data-base so that any source of information can be spotted instantaneously by specifying one of the two values. This seems unnecessary for computer data search, but is required only to retain the uniqueness of each record.

The second level of the texture is prepared for the use of end-users who attempt to retrieve records by these items listed at this level. They are the very common key items shared with other data-bases. Since one might wish to find out a record by the name of title, author, journal or keywords. There are no sub-structures within this level two, i.e. the five key items are independent to each other so that the search goes through in any order of these items.

The third level is a sub-structure of level two. The group of strings defined at level two is split into phrases and stored at this level three. There are quite a few possibilities in making up a rule with regard to the order of phrases. In other words, the phrase has different attribute depending on their field position, i.e. such as the second phrase found in the item of JOURNAL indicates the year of publication and so on. The usage of this routine for staging phrases at right position should be prescribed carefully before preparing inputs for data-base.

The record has a triangular shape of tree structure which has two roots of registration number and classification ID, five branches of Title, Author, Journal, Keyword and Abstruct, and plenty of leaves of key-data. Incidentally our system of providing indexes for record is more or less similar to that of the library,[2] but once the indexes are translated into computer code, fast and reliable search is obtainable absolutely for anyone wishing particular records of his interests.

3. Storing Data in Data-Base Files

3.1 Creating data-base files

In order to store the data in data-base files, it is necessary to secure the file spaces in the user's area of the computer system, and this activity is performed by the system utility program FILSYS. The FILSYS activity, in general terms, creates, modifies and releases files in user's file space. This first activity is to create an IDS (Integrated Data Store) file, i.e. the data-base file in the user's file space.

3.2 Initializing data-base files

The INQ utility program called QATI is provided by

the system to initialize the INQ files before physical loading of the input data. This activity requires one directive card to set the first and the last page numbers which correspond to the numbers found in the RNG phrase of previous FILSYS activity.

3.3 FDL descriptions

In order to store the data in the INQ files, the nature of the data, namely their names and types, should be defined and registered in the respective files.[3] This is accomplished by another INQ utility program, FDL (File Description Language). FDL is a language similar to FD (File Description) of COBOL, by which we define the names and the types of the data to be stored in the data-base file. Only after defining them, we can store the data in the data-base file. FDL performs this recording of FDL descriptions onto the data-base file.

3.4 Generating INQ sections

FDL descriptions discussed above are associated with the data-base file itself and defines its file structure. In order for a programmer or an end-user to see or search for data in the data-base file, one also has to define a method of viewing the file, and this is again performed by one of the INQ utility programs, i.e. INQ Section Generator. INQ Section descriptions depend upon the FDL descriptions, but to some extent they are independent. The users can specify only one part of the FDL descriptions if they want to see only a part of data. They can also connect up to four different data-base files in the INQ section and establish a very large virtual data-base file.[4] In principle, the INQ Sections are quite flexible, but in order to load the data into a real data sector and an index sector, the system must refer to the INQ Section, which is actually the repetition of FDL descriptions written either in COBOL or FORTRAN type language.

3.5 FORTRAN program for data preparation

The original raw data were punched on to the ordinary 80 column computer cards in more or less the same way as one types. Obviously these raw data cannot be used directly as an input to the data-base files since their structure is not in accordance with the FDL descriptions. So one has to reform these raw data according to the FDL descriptions and the requirements of LOADER ONE, which loads the edited data into a real data sector of the data-base file.

3.6 Loading the data into data-base files

The INQ utility program called INQ LOADER ONE is used to read the input data stored in the user's permanent file and to load them into a real data sector of the data-base file. It also prepares an intermediate file of the sorted data under each key item, i.e. the input data for INQ LOADER TWO, which will generate an index sector of the data-base file.

3.7 Generating an index sector

The last step in creating data-base files is to store the sorted data in the intermediate files, prepared by LOADER ONE, into a key index sector of the data-base files. This index is prepared for every record set in consecutive order and is, in turn, made accessible by data search verbs of EUL (End User Language) or DML (Data Manipulation Language). This activity is performed by another INQ utility program, LOADER TWO. If these activities are successfully completed, the data-base files are established and are ready to be reached by EUL or DML, as will be mentioned below.

4. JCL (Job Control Language) Cards for the Above Activities

In order to execute a series of the above activities, one must inform the system of the kind of activities to be executed, the kinds of files and the kinds of data to be used, etc. This is done by supplying the system with a series of JCL cards and the data cards. A complete list of JCL cards needed for generating and accessing the data-base file is shown in **Figure 2**.

The entire job stream is made up of several activities, the beginning of which are identified by the system call statements indicated by "A (Activity)" which is supplied by the system at the beginning of the lines.

The first FILSYS activity can be ignored since this is to release the data-base file. This, however, is necessary to run the entire job a number of times to correct execution errors.

The second FILSYS activity is to create a data-base file in the user's file space. The third activity is to initialize the data-base file. One must specify the catalogue/file name of his data-base file, which is created in the previous activity and will be initialized by the present activity. This is done by the PRMFL (Permanent File) statement. After PRMFL, one needs to insert a parameter card to inform the system about the range of pages to be initialized.

The fourth activity FDL is done by calling the FDL processor which is kept in the file INQ under the name FDL and access is gained by the first two PRMFL cards. The data-base file which has already been initialized and will be structured in the present activity, should be

```
¥¥    S966T ENTERED *S1000 AT 13:08:37 ON 10/07/83 TSS/S   3-08-43
0001  ¥       SNUMB    S966T
0002  ¥       COMMENT  6080333636        TSS CARDIN
0003  ¥¥      USERID   60803336364
      ¥C      JOB      60803336364 ,B,HIDE,R
0004  ¥*      CPROC    A/B,,6080333636,6080333636¥ ,R,B,,C,HOLD
0005# ¥       IDENT    6080333636,3636B1007,R,B                          1007831308
0006# ¥       JOBDEF   DEST=,CLASS=C,OPTION=HOLD
0007# ¥¥      USERID   60803336364
0008# ¥       LIMITS   25,,,20000
0009# ¥       LOWLOAD
0010# ¥       OPTION   FORTRAN,RELMEM
0011  ¥       LIMITS   25,,,3000
0012 A¥       FILSYS
0013  ¥       IF       ABORT,ENDJOB
0014 A¥       FILSYS
0015  ¥       IF       ABORT,ENDJOB
0016 A¥       PROGRAM  QATI
0017  ¥¥      PRMFL    A1,W,R,6080333636/WRIDATA
0018  ¥       IF       ABORT,ENDJOB
0019 A¥       PROGRAM  FDL
0020  ¥       LIMITS   25,70K
0021  ¥¥      PRMFL    **,R,R,INQ/FDL
0022  ¥¥      PRMFL    H*,R,R,INQ/FDL
0023  ¥¥      PRMFL    DB,W,R,6080333636/WRIDATA
0024  ¥       FILE     S1,X1D,50R
0025  ¥       FILE     S2,X2D,50R
0026  ¥       FILE     S1,X1D,50R
0027  ¥       FILE     S3,X3D,50R
0028  ¥       FILE     I1,X4D,50R
0029  ¥       FILE     I2,X5D,50R
0030  ¥       SYSOUT   LP,ORG
0031  ¥       DATA     CD
0032  ¥       IF       ABORT,ENDJOB
0033 A¥       PROGRAM  INQGEN
0034  ¥       LIMITS   25,60K
0035  ¥¥      PRMFL    **,R,R,INQ/INQGEN
0036  ¥¥      PRMFL    H*,R,R,INQ/INQGEN
0037  ¥¥      PRMFL    DB,R,R,6080333636/WRIDATA
0038  ¥¥      PRMFL    CP,W,S,6080333636/WRILIB
0039  ¥       SYSOUT   LP,ORG
0040  ¥       DATA     CD
0041  ¥       LOWLOAD
0042  ¥       OPTION   FORTRAN
0043 A¥       FORTRAN  LSTIN,BIN
0044  ¥       LIMITS   25,40K,-2K
0045  ¥¥      SELECT   6080333636/WRI/WRISOUB2
0046  ¥       IF       ABORT,ENDJOB
0047 A¥       EXECUTE
0048  ¥       LIMITS   25,120K,-6K,1500
0049  ¥       SYSOUT   06,ORG
0050  ¥¥      PRMFL    08,R,S,6080333636/WRI/DATAA
0051  ¥       FILE     09,A1S,100L
0052  ¥       IF       ABORT,ENDJOB
0053 A¥       PROGRAM  INQLD1
0054  ¥       LIMITS   25,80K,,1000
0055  ¥¥      PRMFL    **,R,R,INQ/INQLD1
0056  ¥¥      PRMFL    H*,R,R,INQ/INQLD1
0057  ¥¥      PRMFL    DB,W,R,6080333636/WRIDATA
0058  ¥       SYSOUT   LP,ORG
0059  ¥¥      PRMFL    IQ,R,S,6080333636/WRILIB
0060  ¥       FILE     S1,S1D,50R


0061  ¥       FILE     S2,S2D,50R
0062  ¥       FILE     S3,S3D,50R
0063  ¥       FILE     W1,W1D,100L
0064  ¥       FILE     W2,W2D,100L
0065  ¥       FILE     W3,W3D,100L
0066  ¥       FILE     W4,W4D,100L
0067  ¥       FILE     WK,K1D,100L
0068  ¥       FILE     IN,A1D
0069  ¥       FILE     OT,A3S,100L
0070  ¥       DATA     CD
0071  ¥       IF       ABORT,ENDJOB
0072 A¥       PROGRAM  INQLD2
0073  ¥       LIMITS   25,120K
0074  ¥¥      PRMFL    **,R,R,INQ/INQLD2
0075  ¥¥      PRMFL    H*,R,R,INQ/INQLD2
0076  ¥       SYSOUT   LP,ORG
0077  ¥       FILE     IN,A3D
0078  ¥¥      PRMFL    IQ,R,S,6080333636/WRILIB
0079  ¥¥      PRMFL    DB,W,R,6080333636/WRIDATA
0080  ¥       DATA     CD
0081  ¥       ENDJOB
  TOTAL CARD COUNT THIS JOB = 000260
```

Fig. 2 JCL Cards for Generating and Accessing Data-Base File

designated as DB file by the third PRMFL card. An optional operand "ORG" in the SYSOUT (System Output) statement, which assigns a local output device to the system output, is necessary when one runs this job through the remote batch mode. Otherwise, the system report of FDL compilation will not be printed out through LP at the remote batch station. After DATA statement indicating that cards are the input media, the series of cards are to be inserted here.

The next step is to generate INQ section. INQ Section Generator is stored in the file INQ under the name INQGEN as specified in the first two PRMFL cards. Again one needs to specify the catalogue/file name of the data-base file in the third PRMFL statement in order for the system to read FDL descriptions. The catalogue/file name of a permanent file into which the INQ section object is copied must be specified. In this case, a permanent file, 6802033059/WRILIB is assigned for that purpose and access to that file is gained by the fourth PRMFL card. Again after the DATA card, the COBOL INQ section descriptions together with various descriptive cards should be inserted.

The eighth activity is to execute the program LOADER ONE, which is stored in the file INQ with the name INQLD1, as specified in the first two PRMFL's. Data-base file WRIDATA which has already been initialized and structured in the previous activities should be designated as DB file by the third PRMFL card. The fourth PRMFL card is necessary for access to the INQ section file WRILIB, which has been prepared by the INQ Section Generator. The consecutive cards to assign work files are required for LOADER ONE to sort the data. In the present case, 4 work files are assigned and this number appears on the WORK card of this activity. IN and OT files are compulsory for LOADER ONE. IN file is the permanent file in which the input data prepared by the FORTRAN program are stored. OT stores the output of LOADER ONE, which in turn will be used as an IN file for the next activity. After the DATA card, the parameter cards for LOADER ONE are to be inserted.

LOADER TWO activity is performed by calling the program INQLD2 stored in the file INQ as indicated by the first two PRMFL's. The input file IN is transferred from the output file OT prepared in LOADER ONE activity and is specified to be disposed of after completion of LOADER TWO activity. INQ section file WRILIB and the data-base file WRIDATA are also necessary and access is gained by the following two PRMFL cards. The parameter card LOAD should also be inserted after the DATA card.

Of course, it is not necessary to run the entire job at one setting. Each activity can be performed separately on condition that the required previous steps have been successfully completed. So after establishing the data-base file, only the last EUL activity remains to be performed. Whenever access to the file is required, prepare the parameter cards and command cards depending upon the kind of work necessary and execute the EUL program.

Provided that one is permitted access, data-base files already established by other programmers may be used by simply performing the EUL activity.

5. Conversational EQL (End User Query Language)

Once the data-base file is created as described in earlier parts, there are two ways to obtain access to this data-base file: One is a contemporary method of TSS (Time Sharing System) using conversational EQL; the other is batch method called DML (Data Manipulation Language) for advanced application programs. These two methods of access can perform essentially the similar functions. The INQ through batch mode, however, is more versatile than EQL, especially in its linkage ability with other system programs.

Conversational EQL seems more friendly to ordinary users since it automatically requests the inputs for the next activity and the response of the system appears immediately on the terminal. The results of retrieval can be seen more quickly and it is easier to correct minor errors. Access is easily gained to this EQL program from TSS terminals by typing in "INQ" after "SYSTEM" query.

For those who do not have access to TSS, batch EUL utility makes it possible to input the command parameters from the cards, not from the TSS terminals. Command cards are prepared in exactly the same way as you type them on the terminal. In order to begin data search, the operator must inform the system what data-base file and what INQ section will be used. The EQL/EUL program acknowledges these parameters by listing the tables of data-base files and INQ sections with the messary "INQ DATA BASE RETRIEVE START". The operator then inputs the various command parameters in order depending upon the kind of information he wants from the data-base files.

As shown in **Figure 3**, there are 17 commands in total, such as SORT, GRAPH, SAVE, COPY, AND, OR, NOT, etc., by which various operations may be performed. A selected set of records can be saved in up to 30 SAVE-files by using SAVE command. Records may also be put into the permanent file by the COPY command and the permanent file may be used in different, independently written, programs. Logical operations like AND or OR may be performed in between of two selected SAVE-files.

```
*syst inq

    INQ EQL/JIPS VERSION 9.1-00     15:26'47"      09/27/83

OPTION FILE ?
SYSTEM ?inq

    INQ EQL/JIPS VERSION 9.1-00     15:27'16"      09/27/83

OPTION FILE ? /WRI/OPTION2
**  15:27'29"111  09/27/83  PROC TIME 00'00"303.12

  ------------------------------------------------------
  : FILE NO : FILE NAME     : RECORD CNT : DATABASE NAME :
  ------------------------------------------------------
  :    01   : WRIDATA       :      337   : WRIDATA       :
  ------------------------------------------------------

  ----------------------------------------
  : INQSECTION NAME : TYPE : INQ FILE NO  :
  ----------------------------------------
  :   INQWRI        :  1   :  01          :
  ----------------------------------------


     INQ DATA BASE RETRIEVE START

**  15:27'30"098  09/27/83  PROC TIME 00'00"342.12
? ?
**  15:28'01"928  09/27/83  PROC TIME 00'00"342.48
        INQ  EQL  COMMAND
   1. RETRIEVE  CONDITION EXPRESSION/ -N-
      RETRIEVE  TABLE (M,N)
   2. SORT  ITEM-NAME ...  -DES(ASC)- / -N-
   3. DISPLAY  ITEM-NAME (VARIABLE-NAME)  ... / -N- -LM-
   4. GRAPH  ITEM-NAME (VALUE)  ... X(ITEM-NAME) / -P(B)- -N-
   5. SAVE N
   6. AND M,N
   7. OR M,N
   8. NOT N
   9. COPY N,FILE-NAME
      COPY FILE-NAME,N
  10. TABLE  ITEM-NAME-1 BET(EQ) VALUE ...
             ITEM-NAME-2 BET(EQ) VALUE ... / -N- -TOTAL-
  11. KEYLIST  ITEM-NAME-(VALUE)- ...
  12. FIELD   INQ-FILE-NAME(ITEM-NAME ... )
  13. CHANGE INQ-SECTION-NAME
  14. LET VARIABLE-NAME = VALUE
      LET VARIABLE-NAME = ITEM-NAME-1
          OP ITEM-NAME-2 (VALVE)  ...
  15. CALL MACRO-NAME,VARIABLE-NAME = VALUE ...
  16. ?
  17. DONE
  18. MOVE ON FILE-NAME ITEM-NAME ... / -N-
  19. FIND ITEM-NAME (VALUE)  ... / -N-
  20. THESAURUS  THESAURUS-WORD
  21. SEARCH  FROM FILE-NAME ITEM-NAME (VALUE) ...
  22. INFORM
  23. SKIP  -TOP(NO,NN)-
  24. GDISPLAY  ITEM-NAME ... T(ITEM-NAME) X(ITEM-NAME)
   ?     COMMAND  END
**  15:28'02"253  09/27/83  PROC TIME 00'00"356.14
```

Fig. 3 INQ Commands for Retrieval

There are also useful functions like SUM, AVE, MAX, MIN, DEV, etc. available in the system. What is more, by defining new variables in LET or CALL commands, it is possible to virtually build new items in the data-base file. Thus even without knowing a complicated programming language, the data-base file may be used in an infinite number of ways.

6. INQ Commands for Retrieval

6.1 RETRIEVE and DISPLAY

The RETRIEVE command is to select a set of records according to the retrieval conditions. After retrieval, the number of records which satisfy the searching conditions is typed out. In order to actually see the retrieved records,

```
**  11:39'27"835  09/10/83  PROC TIME 00'00"859.62
? RETR RGSTNB > MIN AND RECID = AUTH AND PHRSNO > MIN AND KWORD = INOUE
**  11:40'28"282  09/10/83  PROC TIME 00'00"860.04
  12      RECORDS FOUND
**  11:40'30"311  09/10/83  PROC TIME 00'00"958.26

? DISP RGSTNB TEXT
**  11:40'45"142  09/10/83  PROC TIME 00'00"958.60

 RGSTNB       TEXT

 1981-3-31    INVESTIGATION ON WELDING ARC SOUND (REPORT 4) -VIBRATION ANALYSIS
              OF BASE METAL DURING WELDING-;
              VIBRATION OF BASE METAL AT CO2 ARC WELDING WAS MEASURED  IN ORDER
              TO CLARIFY ITS  CHARACTERISTICS AS WELL AS ITS  CORRELATION  WITH
              THE BEHAVIOR OF MOLTEN POOL AND THE WELDING ARC SOUND.
              VIBRATION,ESPECIALLY THAT IN LOW FREQUENCY REGION CAN BE DETECTED
              FAIRLY  ACCURATELY WITH A  PIEZOELECTRIC  ACCELEROMETER  OF THE
              SENSIBILITY OF 100 MV/G CLASS, AND  THE  VIBRATION  HAS A GOOD
              COMPETENCE AS THE INFORMATION SIGNAL TO  EXHIBIT THE  BEHAVIOR OF
              MOLTEN POOL.;
              ACOUSTICS;ARC WELDING;PROCESS PANAMETER;ENVIRONMENT;
              ARATA Y;INOUE K;FUTAMATA M;TOH T;
              TRANSACTION OF JWRI;3;31;1981;39;46;
```

Fig. 4 EQL Activities Showing RETRIEVE and DISPLAY Commands

one uses the DISPLAY command, which prints out the retrieved set of records with the required variables one wishes. **Figure 4** is of the activities showing RETRIEVE and DISPLAY commands.

6.2 KEYLIST and FIELD

KEY LIST prints out the value and the record count of the specified items. FIELD prints out the field descriptions of the specified INQ file. This command is only used to assist the data search, but it is especially useful to confirm the file descriptions when one uses the file prepared by other programmers.

6.3 SORT and THESAURUS

SORT command performs the task of rearranging a selected set of records either in ascending order or in descending order according to the value of the specified data item. THESAURUS prints out the thesaurus information of the specified entry word in the Thesaurus File.

6.4 CHANGE

Data-base files are viewed through INQ sections registered in INQ section files. If there is more than one INQ file in the data-base, the number of registered INQ sections should be equal to, or greater than, the number of INQ files. Whenever one wants to change the INQ files to be used, or if more than one INQ file is going to be used at the same time, the INQ section to be referred to should also be changed. This is done by the CHANGE command, which changes the INQ section so far used to that which is to be used in the succeeding commands.

6.5 DONE

The last command card should be "DONE" to inform the system to stop execution. If this command is performed, the execution terminates with the message "INQ DATA BASE RETRIEVE END".

7. Application Program by Means of DML

In principle, the retrieval of data from the INQ files is easily performed by EQL, but sometimes EQL is not satisfactory. If it happens, then one has to write one's own program using the compiler languages. INQ accepts either FORTRAN or COBOL as a host language. DML is a language by which one can retrieve or add data from INQ files. It is inserted into a program written in FORTRAN or COBOL and is used in the form of subroutines.

8. FORTRAN INQ Section

In order for the application program, or the programmer, to use the data in INQ files. INQ sections should be written referring to the FDL descriptions of the files to be used. The language used for describing INQ sections depends on the compiler language.

FORTRAN INQ section is generated by INQ Section Generator and the generated INQ section object is stored in a SELECT file, which should be created beforehand by FILSYS utility.

9. DML

DML prepares 27 INQ verbs with which one can manipulate the data in INQ files. These verbs are inserted in the application program written in FORTRAN as an argument of subroutines. But before using INQ files in the application program, the files must be opened. This is performed by the INQ verb OPEN, which defines the INQ files and INQ sections to be used in the program. As the INQ verbs are treated as arguments of INQ subroutines, they are referred to in CALL statements. OPEN should always be associated with another INQ verb CLOSE, which declares the end of the opened files.

10. INQ Linker

Before executing the application program, one has to prepare a proper set of loader control statements depending the INQ verbs used in the program. One of the utilities call INQ LINKER prepares this set of loader control statements and stores it into another SELECT file, which in turn is referenced by the system when loading the program.

11. Discussion

The data-base is now very practical means of manipulating and processing of mass data. In the past, just a few scientists in a particular field of engineering concerned to its development and maintenance. Latest computers have equipped with system utilities which provide the capability of data base manipulations. Most of these main frames are interfaced with quite a few TSS terminals from which down-end users can enter his input data or commands to retrieve a group of data for his specific desire of interest.

In a series of data manipulations involved in data-base operation, TSS knowledge is essential and released data processings through peripherals and mass data storage device should have to be exercised extensively in telex communication. Therefore the data-base performance depends heavily on its main frame efficiency.[5] The method of access to data-base on a particular machine is also subjected inevitably to their commands which are normally exclusive to other systems.

There are three types of computers currently in use, i.e. the first and the largest is multi-processing super computer, whose size is of our ACOS-6/SYSTEM 1000. The second so-called mini computer, which are very common to our department offices and automated laboratories. The third is handy micro computer which is now very familiar to almost all kind of engineers. These are the machines, with which we are to develop and operate our data-base system. Consequently the efficiency and the reliability of data-base depends heavily on their OS (Operating System). Because the OS acts like an active interface between us and real data on disks, and if in the absence of OS, we do not have any means to communicate with computer. It seems that there are not fundamental differences in the architecture of these machines and not much differences in the principle of operation. However, tremendous dissimilarities are predominating in their real usage. As a result, the same data-base responds inevitably in different ways at each machines, but we do have to compromise an optimum efficiency somehow to obtain the maximum performance.

In this sense, micro and mini are not the proper device to pursuit the entire job required for data-base manipulations. Their CPU capacity and mass data storage are too small to accommodate more than a few 10 thousand indexes, of which number is normally expected as minimal capacity. Their DBMS (Data Base Manipulation System) is generally not comprehensive so as to compile application programs for end user support. However, these two, especially the micro has very easy interfaces to many kind of data logging devices such as digital sensors for displacement and temperature measurement. Millions of data can be acquired instantaneously by running interfaced micro and can be stored in their mini floppy disks. In this connection, the micro is prospective to data acquisition and processing of analog to digital conversions.

These data on the micro can be transferred later to the mini computer for further computations and graphical presentations of numerical results.

Then we are obliged to make use of computers to establish data-base system. There are two choices for us with regard to the site to install the data-base. We can design the system with one particular machine or in between of more than two computers by means of so called computer network.[6] As mentioned earlier, none of the three seems self-contained in itself to pursuit the entire job stream of data-base operation. To depend exclusively on one particular machine for running data-base is often very risky because it unexpectedly hits the limit of capacity at data storage. Usually it happens much earlier than we expect. Once the overflow occurred, recovery of system by modification is rather impossible even though eliminating unnecessary spaces among CPU and disks can not do much help. So the second choice is promising. Dispersing discrete tasks to various machines seems to be the most optimum configuration of the system, i.e. data acquisition by micro, manufacturing by mini, housekeeping by super. The series line connection of three computers can be clustered into two groups, data input/output is dealt with by micro/mini, and daily bookkeeping by mini/super. The data transfer is actually taking place in both direction of full duplex line between mini and super, whereas the micro is only supporting the mini for its data transmission.

As of the softwares involved in micro and mini, the micro has not much intelligence to support data-base operations. The mini has so called DBMS utilities, which handle just a few functions necessary for data-base operation. The DBMS is entirely dependent to their compiler languages, with which we need to write our own program to suffice the necessaries of data-base manipulation. So far the computer manufacturer does not make any provision of their sub-systems, but they do supply only a bunch of subroutines accessible from compiler language.

The DBCF (Data Base Control Facility) in HATAC E800 has two method of access to their data-base. The first way of searching is by key items affixed to each record, of which key data is assigned to a unique value for their identity. The second method is by the record number given in consecutive order by the system to each records. This way of access works in sequential or random order depending on the requirements of searching. However, the access is obtainable only by satisfying complete match of the keyfield to the key-data specified for retrieval. The maximum number of key field available for one record is fixed to five. This type of search is very similar to that of indexed sequential record with multi-indexes, which is typical in COBOL compiler. Searching by record number is another alternative and which works like batch line editor for source library updates. Expanding DBMS capacity seems rather discouraging even if obtainable to some extent by writing elaborate application programs of his own. Otherwise, the DBCF subroutines in the system behave like library macro's, which are very flexible and versatile in general, but too discrete to pursuit a task of data-base job.

12. Concluding Remarks

As shown in the previous sections, the entire job stream for data-base file seems a bit complicated. The application programs seem even more difficult. But once the procedure is established, it becomes very easy since one only has to follow more or less the same routine if he can in any way possible obtain access to, and has permission to use the computer system in which the data-base is created. Even without any knowledge of computer languages, he can retrieve the data by using EQL commands as demonstrated above, or if he is versed in languages like FORTRAN or COBOL, he can write his own application program to manipulate the data in the file.

Once the FDL structure is designated, most of the job procedures are supplied by the system. Exceptions are COBOL INQ section descriptions and the parameter cards for LOADER ONE, which can be prepared easily according to the FDL descriptions. The most difficult part may be the FORTRAN program to convert the original raw data into inputs for LOADER ONE. Anyone who knows any of the compiler languages should not find much difficulties in writing his own program.

Application programs have to be prepared for advanced ways of using data-base file. The data can be prepared in different ways depending upon the type of data search to be conducted. These and augmentations for current data-base system will be taken up and published in due course.

References

1) Y. Isomoto, T. Matsuda and N. Tanaka: Guidance System for Structuring or Restructuring of a Data-Base in Multiple Data-Base Management Systems, Journal of Information Processing, Vol. 5 (1982), No. 3, 182–187.

2) Project Report 1982, Association of National University Libraries in Tokyo (Network Division), June (1983), (in Japanese).

3) H. Takechi and J. Nakamura: Advanced Application of a Data-Base Management to the Analysis of English Texts, Journal of Cultural and Social Sciences, University of Tokushima, Vol. 19 (1983).

4) H. Takechi and J. Nakamura: Application of a Data-Base Management System to the Analysis of English Text, Journal of Cultural and Social Sciences, University of Tokushima, Vol. 18 (1982), No. 18, 95–123.

5) B. Shneiderman: Improving the Human Factors Aspect of Data Base Interactions, ACM Trans. Syst., Vol. 3 (1978), No. 4, 417–439.

6) D. Mcleod and D. Heimbigner: A Federated Architecture for Data Base Systems, National Computer Conference (1980), 283–289.