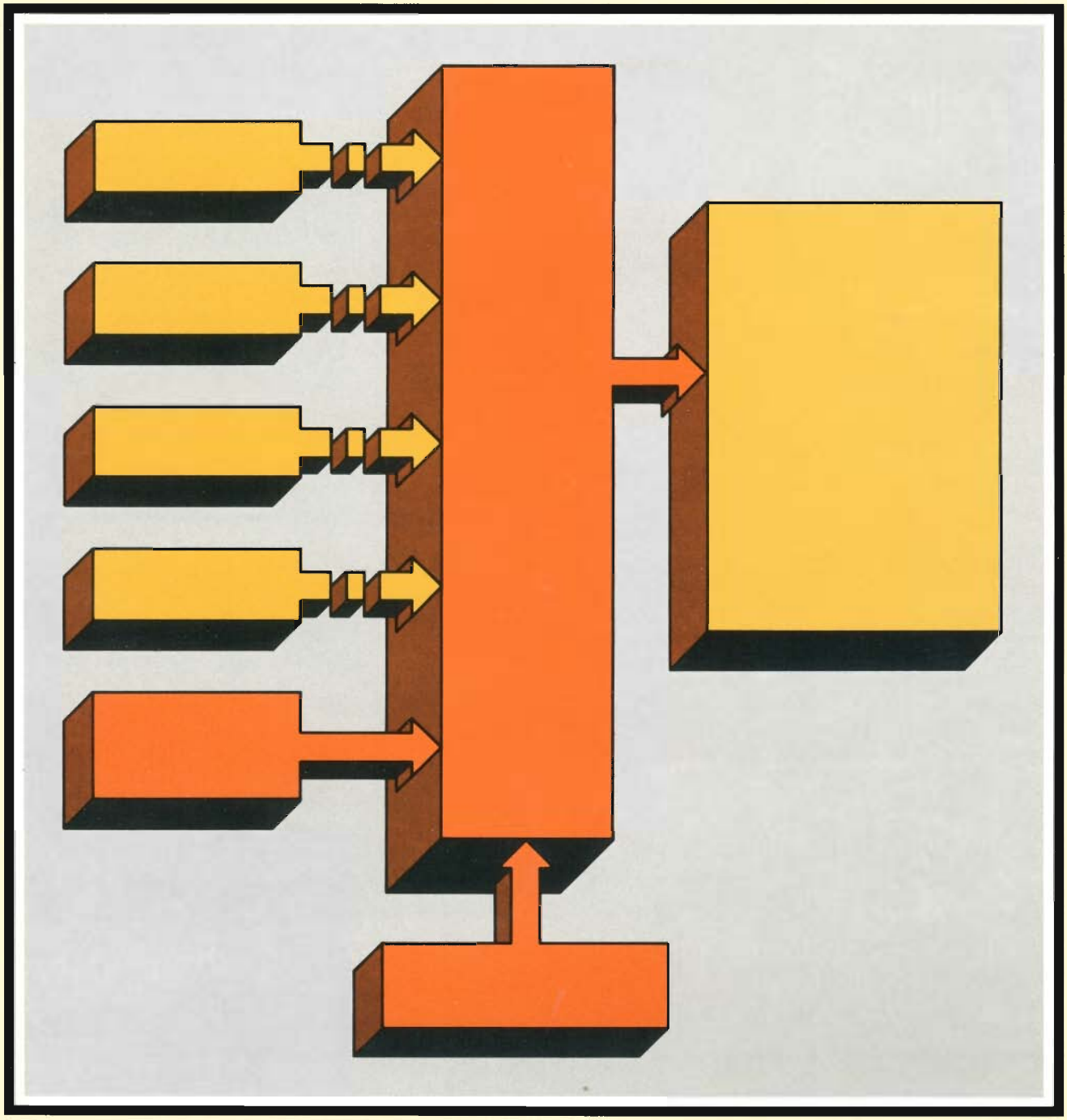
HEWLETT PACKARD

QUERY User's Guide

for the HP 9000 Model 20



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

QUERY User's Guide

for the HP 9000 Model 20

Part No. 97053-90001

Hewlett-Packard Engineering Systems Division
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition.

Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a note at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered updated.

Reprints incorporate all updates since the last edition.

The printing date changes on reprints and new editions.

First Edition...December 1982
Update 1...February 1983

HEWLETT PACKARD

Manual Update

Update 1...February 1983

This update goes with:

QUERY User's Guide
97053-90001
First Edition...December 1982

THE PURPOSE OF THIS MANUAL UPDATE

is to provide new information for your manual to bring it up to date. This is important because it ensures that your manual accurately documents the current version of the product.

THIS UPDATE CONSISTS OF

this cover sheet, a printing history page, all replacement pages and write-in instuctions (if any). Replacement pages are identified by the update number at the bottom of the page. A vertical line (change bar) in the margin indicates new or changed text material. The change bar is not used for typographical or editorial changes that do not affect the text. New pages to be added do not contain change bars.

TO UPDATE YOUR MANUAL

identify the latest Update (if any) already contained in your manual by referring to the Printing History Page (page ii of your manual). Incorporate only the Updates from this packet not already included in your manual. Following the instructions on the back of this page, replace existing pages with the Update pages and insert new pages as indicated. Destroy all replaced pages. If "write-in" instructions are included, they are listed on the back of this page.

Hewlett-Packard Engineering Systems Division
3404 E. Harmony Road, Fort Collins, CO 80525

P/N 97053-90001
U 0283
Printed in U.S.A.

Replace the following existing pages with the Update page and add new pages as indicated.

Update 1: Replace
pages iii and viii
Add
page ix
Replace
pages 3 thru 6
pages 9 and 12
pages 15 thru 18
pages 28 thru 37
page 51
pages 64 thru 70
pages 80, 139 and 140
Add
pages 141 thru 152

Table of Contents

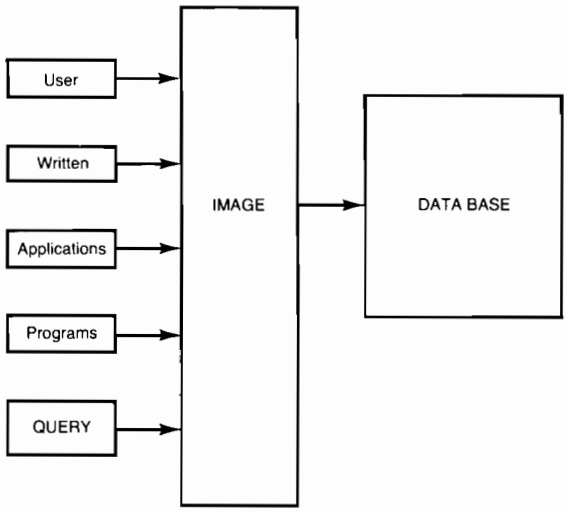
Chapter 8: Defining a Data Base

Chapter 9: Linking Other Subprograms to QUERY/9000

Chapter 1
General Information

What is QUERY/9000?

The HP 9000 Data Base Management System enables you to set up, access and maintain a data base. A **data base** is a group of logically related files containing data and information about how that data is interrelated.



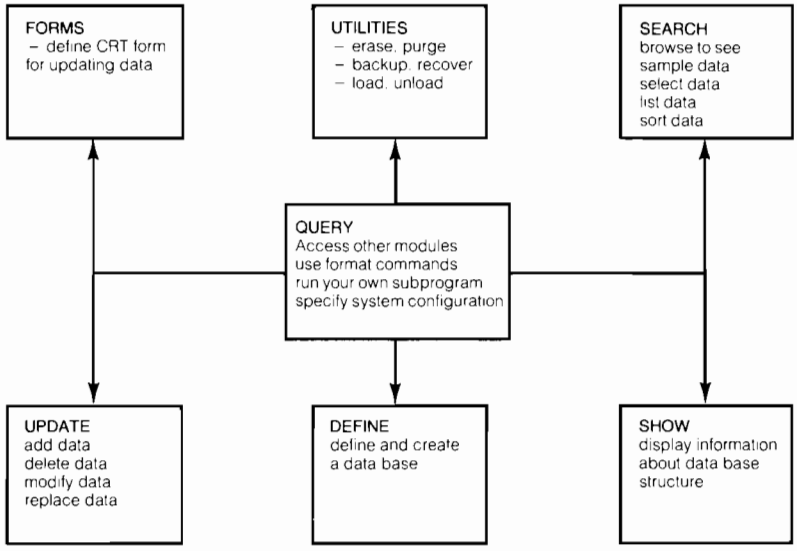
QUERY/9000 is part of the Data Base Management System that enables you to perform data base management operations without having to write a program. IMAGE/9000 integrates the data files for an application and relieves you of the task of data organization and management.

QUERY/9000 is a BASIC language program which enables you to define a data base, put information into the data base and query or ask for information from the data base, thus saving the time it takes to write and debug a program. It also contains utilities to backup and recover data bases.

QUERY/9000 consists of six subsystems — SHOW, SEARCH, UPDATE, FORMS, DEFINE and UTILITIES — which are linked together and accessed through a central subsystem called QUERY.

QUERY is shipped as a set of files contained on three discs. The QUERY program cannot be run from these discs. You must load and run the BUILD_QUERY program to make a useable QUERY product. BUILD_QUERY copies the files from the three master discs onto a single large disc such as the internal disc or an HP 7908. QUERY is then in a useable form. The three master discs should be saved as backup.

The following diagram represents QUERY/9000 and gives a brief description of each subsystem.



How QUERY/9000 Works

The various features of QUERY/9000 are accessed by pressing **softkeys** (the Special Function Keys 0 through 31). At any particular point in the program, the available options (called a **menu**) are displayed and are accessed by pressing the corresponding key. The EXIT softkey is used to return to the previous menu.

Throughout this manual, the following assumptions are made for all examples:

1. QUERY files and the sample LIBRARY data base have been placed on the internal disc drive at :CS80,7. Both the QUERY directory and the LIBRARY directory are at the root directory in the SDF directory structure.
2. This disc has a volume label "MY_VOLUME". (Yours may have a different volume label).

Data Base Management Steps

There are six basic procedures involved in using QUERY/9000 for data base management.

1. Become Familiar with QUERY/9000

Chapter 2 contains step-by-step example operations to familiarize you with QUERY/9000.

2. Understand Data Base Concepts and Terms

It is essential to understand the basic concepts and terms in order to use QUERY/9000 effectively. Chapter 3 describes these concepts.

3. Design the Data Base

If you are going to use a data base of your own, rather than using the data base of someone else in your organization, you must convert your application needs into a design that QUERY/9000 can understand. This is the most important step in using your data base management system. The Data Base Design Kit provides procedures and tools to help you do this.

4. Define and Create the Data Base

Once you have designed your data base, the DEFINE subsystem is used to describe the data base to QUERY/9000. The DEFINE subsystem is also used to create the data base once it is defined. Chapter 8 covers the DEFINE subsystem. Alternately, the SCHEMA processor can be used to define the data base. The SCHEMA processor is described in the IMAGE Data Base Programming Techniques manual.

5. Access the Data Base

Once the data base is created, data can be stored, retrieved and modified. The UPDATE and SEARCH subsystems are used to perform these operations. If you are unfamiliar with a data base, the SHOW subsystem provides information about the structure of the data base. These subsystems are introduced in Chapter 2 and are covered in more detail in Chapters 4, 5 and 6. In addition, CRT forms can be used to enhance data entry and updating. The FORMS subsystem is covered in Chapter 7.

6. Maintain the Data Base

Backing up your data base should be a regular part of your data base operations. This information can be found in Chapter 11. Also covered are the utilities to LOAD and UNLOAD a data base. If a data base needs to be altered for different passwords, set names or capacities, then the DEFINE module (described in Chapter 8) allows you to make these changes without going through an UNLOAD/LOAD process.

Advanced Operations

QUERY/9000 enables you to perform more advanced operations such as linking in your own subprogram using the RUN USER PROGRAM feature. If you are very familiar with various QUERY/9000 operations, you can use the FORMAL COMMAND mode to bypass the softkeys and type in the operations directly. These are covered in Chapters 9 and 10.

The Sample Data Base

Throughout this manual, a sample data base is used in the examples. This sample data base (called LIBRARY) was shipped with your Data Base Management System and enables you to become familiar with and perform the QUERY/9000 operations before you create your own data base.

The Scenario

The NOP Manufacturing Company has eight plants. Each plant has a small library of reference books. The books may be checked out by any employee from any plant and kept as long as needed.

NOP would like to use an HP 9000 to keep the following information:

- Information about each book (title, author, call number[1], publication date, publisher, cost, and subject).
- A list of every book, identifying which plant owns it.
- Whether or not a book has been checked out, and when.
- Which employee has borrowed a book (including name, employee identification number, department and phone number).
- General information about each library branch (plant name and address, librarian and phone number).
- Classification of each book by subject.

All of this information has been arranged logically and stored on a disc to become your LIBRARY Data Base. Books can be found by the author, subject, title, or call number. The call number can be used to determine whether or not a book is checked out (and to whom) and if there are any additional copies available. Additionally, all books on a particular subject can be found. Exercises using this sample data base are presented in Chapter 2.

The detailed information about the LIBRARY data base is:

data base name: LIBRARY
passwords: LIBRMGR (READ/WRITE access to all sets)
ENGINEER (READ-ONLY access to all sets)
maintenance word: BOOKS

1 A simple numbering scheme is used for the call numbers that identify each unique author-title combination in the library. This scheme is not related to any particular library numbering system.

Getting Started

Before you can use QUERY/9000, there are three procedures you must follow:

1. Load the IMAGE_DBM and Mass Storage Options, as described in Appendix B of your HP 9000 BASIC Programming Techniques manual.
2. Transfer the QUERY/9000 file from the 5¼" discs to your computer's internal disc or other large disc.
3. Create a copy of the sample LIBRARY data base.

Steps 2 and 3 are discussed in detail in the following section.

Transferring QUERY/9000 from 5¼" Discs to your Disc

QUERY/9000 comes to you on three 5¼" discs, but it must be installed on a single large disc before it can be used.

The BUILD_QUERY program is provided to transfer it for you.

Before transferring QUERY/9000, ensure that your external mass storage device (if any) is properly connected to your HP 9000 and that they are both turned on. Then follow these steps as the program prompts you.

1. Insert the disc labeled: 'QUERY Disc 1 of 3 (English version)' into the internal disc drive.
2. Type `LOAD "BUILD_QUERY:INTERNAL"` and press [EXECUTE].
3. When the program is loaded (the run light in the lower right of the CRT goes out), press [RUN].
4. Enter the media specifier (without quotes) (:CS80,7 is an example for the internal Winchester disc drive.) of the disc drive on which you are storing QUERY/9000 , then press [RETURN].
5. Enter the directory path on which to place the QUERY files, then press [RETURN]. The program suggests "/QUERY", but you can enter a different one.
6. All discs referenced by QUERY must have a volume label. If the target disc does not have a label, you are asked to supply one. An example response would be MY_VOLUME.

Note

The program checks now to see if there is already a version of QUERY/9000 on the disc. If there is, you are asked to purge all files and store QUERY again. Answer 'Y' and press [RETURN].

7. The files on Disc #1 are then transferred to the target disc. When the transfer is complete, you are asked to insert QUERY Disc #2 into the internal disc drive and press [CONT].
8. When this information is transferred, you are asked to insert QUERY Disc #3 into the internal disc drive and press [CONT].

9. When all of QUERY/9000 is transferred, you are asked whether you wish to create a copy of the sample LIBRARY data base. Enter 'Y' and press RETURN.

Note

If the LIBRARY Data Base already exists, enter 'N' and press RETURN.

10. Enter a media specifier without quotes. Place the 5¼" disc labelled sample LIBRARY data base into the internal drive and press RETURN. Then enter a directory path (it must already exist) and press RETURN.
11. When everything is copied, the message 'BUILD_QUERY finished' is displayed.

Running QUERY/9000

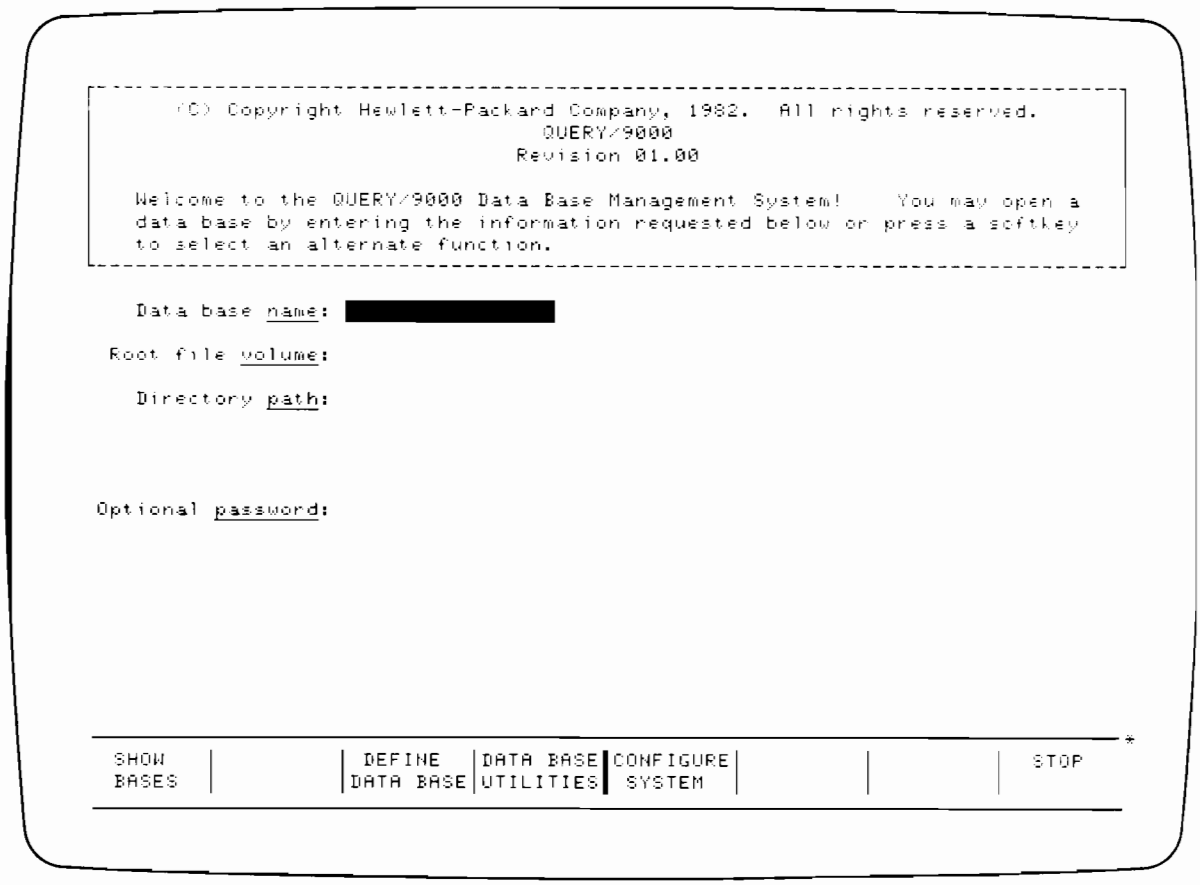
To use QUERY/9000, you need to load the program into the HP 9000 (after transferring it) and run it. To load QUERY/9000, type the following and press EXECUTE:

```
MSI "QUERY:msus"
```

(MSI is used instead of MASS STORAGE IS) then type:

```
LOAD "QUERY"
```

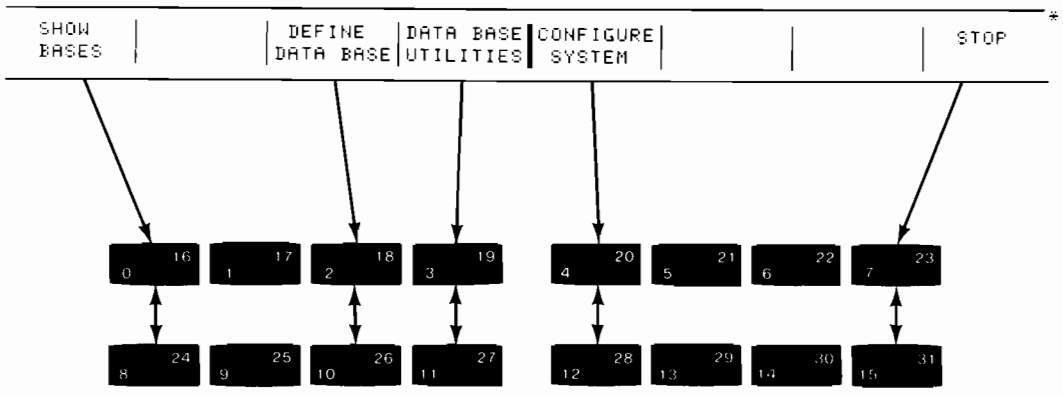
and press EXECUTE. After QUERY/9000 is loaded, press the RUN key. The program is ready when you see the following displayed on the CRT.



The Softkeys

The softkeys are used to access the features of QUERY/9000 and are located at the bottom of the CRT and relate to the Special Function Keys on the keyboard, as shown here:

Softkeys



Special Function Keys

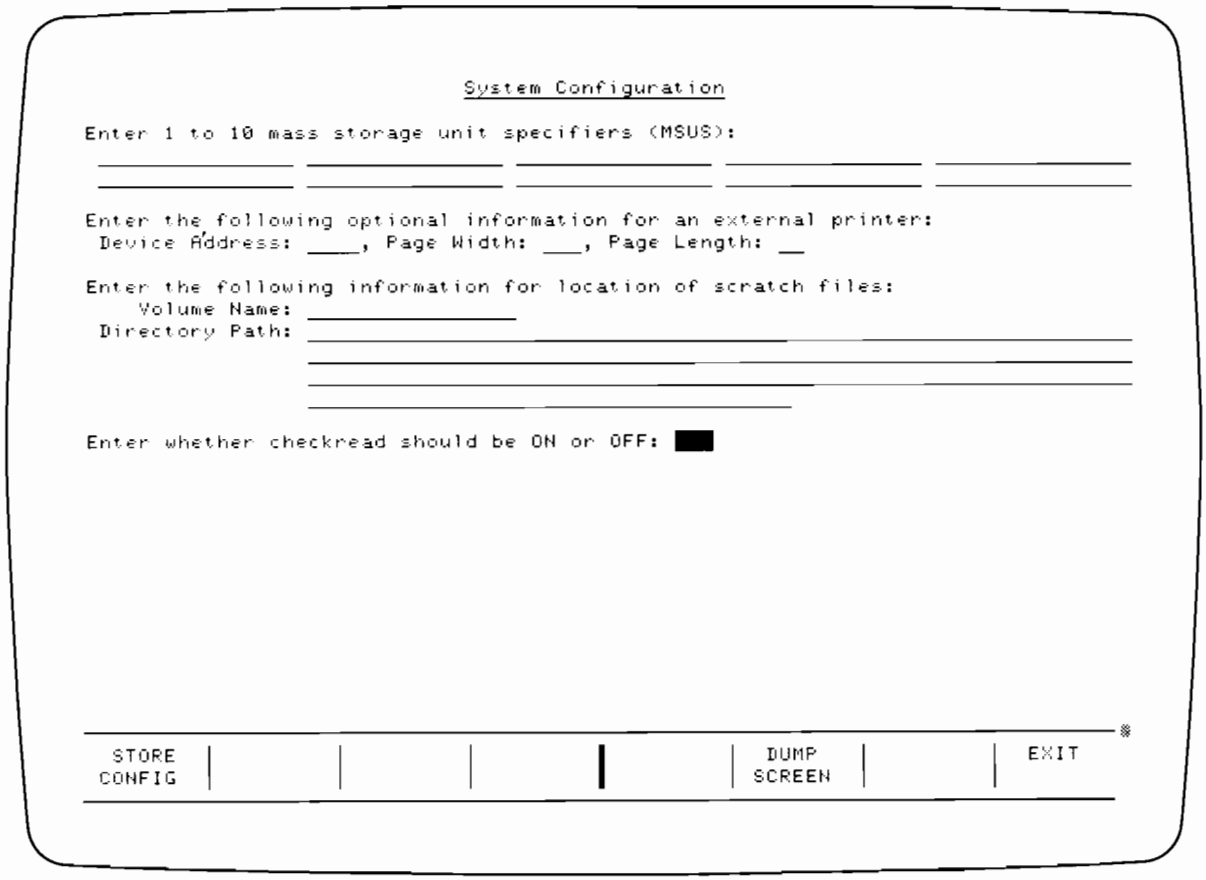
The top row of softkeys have the same functions as the bottom row, as shown above.

Entering Information

While you are using QUERY/9000, press the RETURN or the TAB key to enter information into the system as it is requested.

Configuring Your System

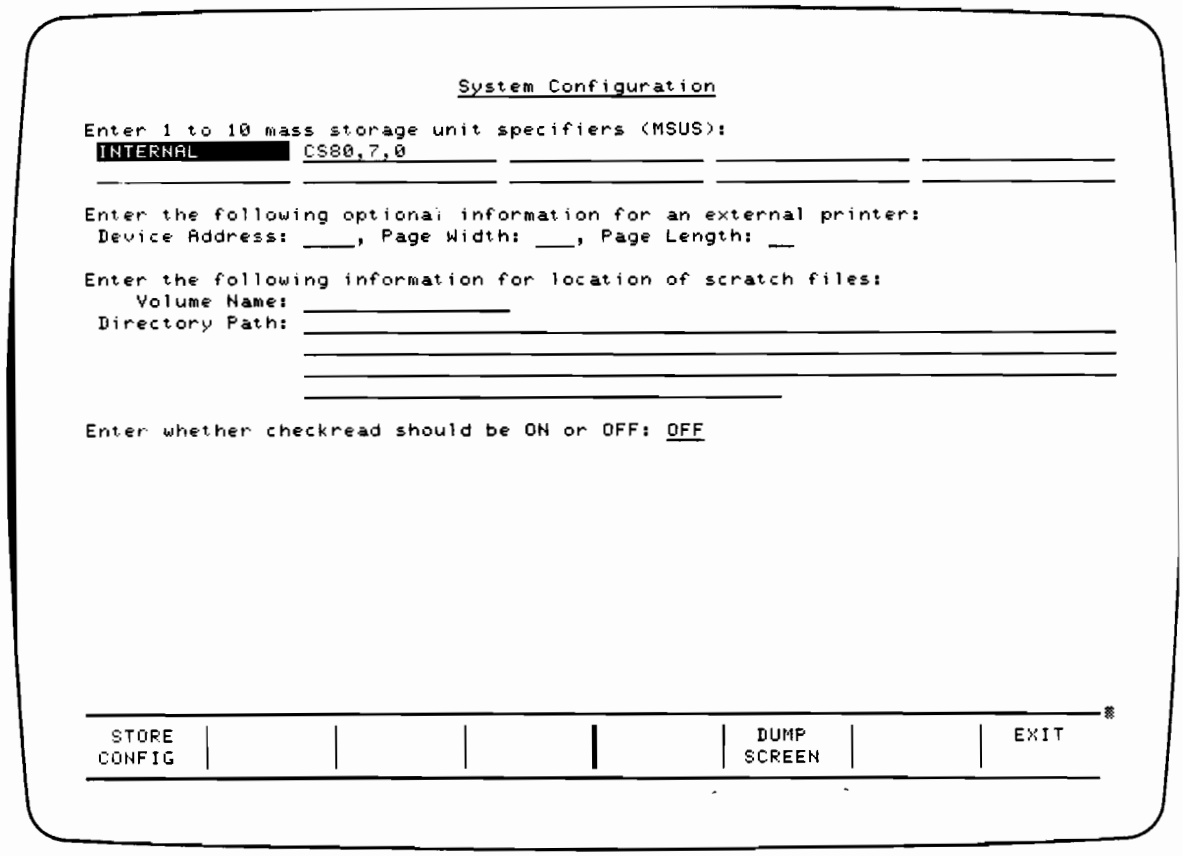
If you are running QUERY/9000 for the first time, you must specify the configuration of your system, such as the location of the optional printer. This should also be done any time you change the configuration. To do this, press the CONFIGURE SYSTEM softkey. The CRT now looks like:



The first step is to indicate the mass storage devices that you are using with your system, including the disc which contains the sample data base. The table shows the device type for five of the more common devices.

HP Device	Device Type (msus)
7908P 16.5 Mbyte Winchester Disc	CS80
7911P 28.1 Mbyte Winchester Disc	CS80
7912P 65.6 Mbyte Winchester Disc	CS80
7933H 404 Mbyte Winchester Disc	CS80
82901M/S 5-¼ inch Flexible Disc (dual)	HP 82901 or HP 8290X
82902M/S 5-¼ inch Flexible Disc (single)	HP 82902 or HP 8290X
9130K 5-¼ inch Internal Flexible Disc	CS80,7,1 or INTERNAL
97093A 10 Mbyte Internal Flexible Disc	CS80,7,0
9885M/S 8 inch Flexible Disc	HP 9885
9895A 8 inch Flexible Disc	HP 9895

Next, you can specify an optional printer for your output, other than the internal printer. For its page length, specify the total number of lines per page. For example, enter 66 for 6 lines per inch on 11-inch paper. This includes top and bottom margins. Specify the volume name and directory path where you want QUERY to create its scratch files. The default is the location of the data base. If there is insufficient room on that disc or you have a faster media available, specify a new location for scratch files; otherwise, there is no need to specify a different location. The last line specifies error-checking on disc accesses and should be "ON" if you are using flexible discs, as shown in the example.



When all the information is entered, record it by pressing the STORE CONFIG softkey.

The System Configuration information needs to be entered only once unless you add or remove mass storage devices or change any other information on this display. When you need to edit the System Configuration screen, the CONFIGURE SYSTEM operation can be performed at the beginning of the QUERY/9000 program.

Chapter 2
Getting Acquainted with QUERY/9000

Introduction

The purpose of this chapter is to enable you to become familiar with the major capabilities of QUERY/9000. This is done using a series of exercises which consist of simple operations. Important data base management terms are also introduced as needed. After performing these exercises, you should be familiar enough with QUERY/9000 to use it for the majority of data base operations. Once you are familiar with the information in this chapter, you can refer to the remaining chapters for more detailed information about how QUERY/9000 works.

Performing the Exercises

Each exercise contains step-by-step instructions. In general, when you are asked to enter the specified data, you should use the [RETURN] key to enter it. Any special instructions are included with the exercise.

Using the Sample Data Base

The exercises in this chapter use the sample data base, LIBRARY, which is provided with your Data Base Management System package. These exercises enable you to perform data base operations using actual data. To perform the exercises correctly, the LIBRARY data base must be in exactly the same form as it was when it was shipped to you. If other people have performed these exercises or the LIBRARY data base contents have been changed in any way, follow the steps (outlined in Chapter 1) to restore the LIBRARY data base to its original state.

Loading QUERY/9000

Before you can access a data base, you must load QUERY/9000 into the HP 9000. (If you've already done this, skip to the next section.) To load QUERY/9000 type:

```
MASS STORAGE IS ":QUERY:msus"
```

(MSI may be used in place of MASS STORAGE IS) and press [EXECUTE].

Now type:

```
LOAD "QUERY"
```

and press [EXECUTE]. Press [RUN] after the run light goes out.

Accessing a Data Base

In using QUERY/9000 to access a data base the first step is to open it. To open the LIBRARY data base, type `LIBRARY` for the data base name, the volume name for your medium, (e.g. MY_VOLUME), the directory path you entered when you copied the LIBRARY data base, and `LIBRMGR` for the password, pressing RETURN after each one. Notice that when you typed the password, it was not printed on the screen. This is to ensure that only authorized people know the password of the data base. Now that the LIBRARY data base is open, the CRT displays:

```
                                QUERY/9000

Data Base: LIBRARY
Number of updates since last backup: 0
Size of search result: 0

              Please select one of the softkey functions below.

                                                                        *
 SHOW      | BROWSE | SEARCH | UPDATE | CONFIGURE | FORMAL  | MORE    | CLOSE
BASE INFO  |        |        |        |  SYSTEM   | COMMAND | KEYS    |DATA BASE
```

Finding Out What the Data Base Looks Like

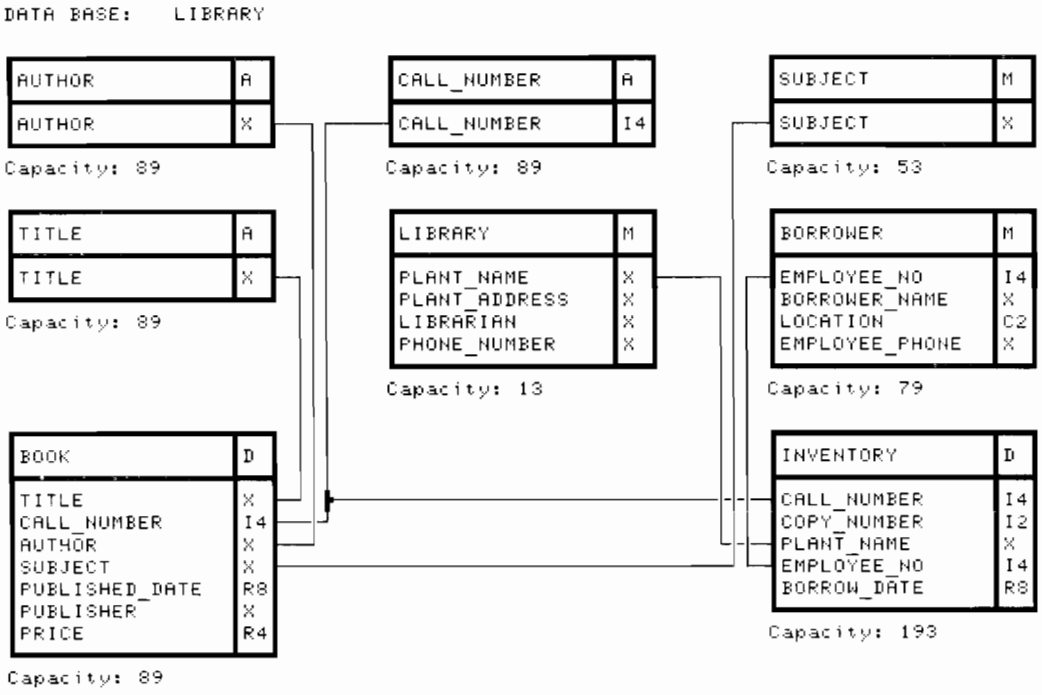
When you encounter a data base for the first time, you might want to know the structure of the data base and look at a few entries to see samples of the data base contents. By pressing the SHOW BASE INFO softkey you are able to display the data base in four different ways. Only two of the ways are discussed in this chapter, the remaining ways are discussed in Chapter 4.

1. Press the SHOW BASE INFO softkey (after the data base is opened).

Structure

First let's look at the structure of the LIBRARY data base. The softkey labeled GRAPHIC STRUCTURE prints a diagram of the data base on the printer. This diagram is useful during other data base operations.

1. Press the GRAPHIC STRUCTURE softkey and the LIBRARY data base is drawn as:



The diagram of the data base shows the data sets within the data base. A **data set** contains information about something, such as a BOOK or BORROWER. Each data set contains **data items** which describe the thing; TITLE and AUTHOR are two of the data items used to describe a book. The diagram of the data base also shows the **data paths** which connect the data sets to each other.

This diagram also shows the maximum number of entries that can be entered for each set, called **capacity**. For example, the BOOK data set can contain, at most, 89 books.

Data Sets

If you just want to know what data sets are in the LIBRARY data base, the SET INFO softkey gives you this information.

1. Type T to indicate that you want the set information listed on the internal printer.
2. Press the SET INFO softkey.

This is what is listed:

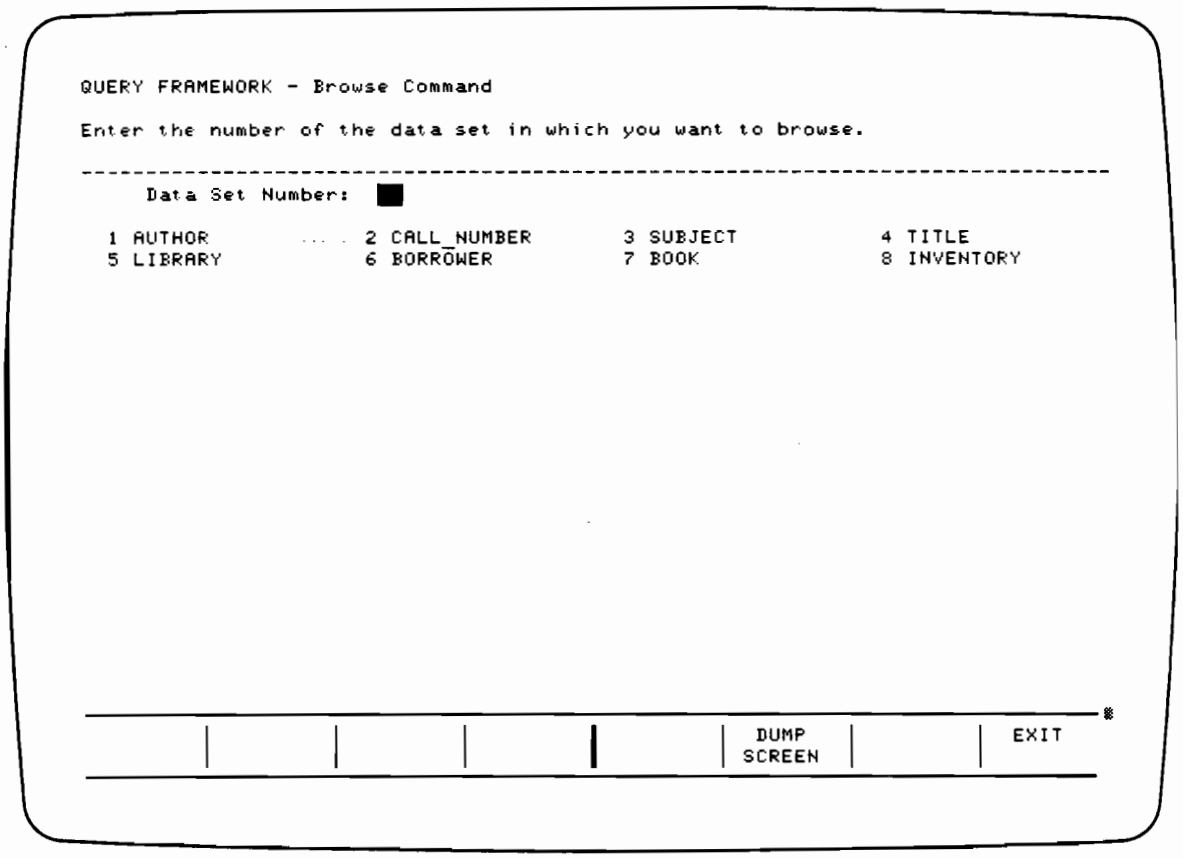
```
 QUERY/LIBRARY:LABEL QUERY

SET NAME          TYPE               ENTRIES     SET          VOLUME
                                                 CAPACITY     LABEL
================  ================   =========   ==========   ================

AUTHOR            AUTOMATIC MASTER          67           89
CALL_NUMBER       AUTOMATIC MASTER          70           89
SUBJECT           MANUAL MASTER             44           53
TITLE             AUTOMATIC MASTER          68           89
LIBRARY           MANUAL MASTER              8           13
BORROWER          MANUAL MASTER             61           79
BOOK              DETAIL                    68           89
INVENTORY         DETAIL                   148          193
```

3. Press the EXIT softkey to return to these softkeys:

```
                                                                          *
 SHOW      | BROWSE | SEARCH | UPDATE | CONFIGURE | FORMAL  | MORE  | CLOSE
BASE INFO  |        |        |        |  SYSTEM   | COMMAND | KEYS  |DATA BASE
```

Browsing Through the Data

The BROWSE softkey enables you to view all entries in a particular data set one at a time. The primary purpose of BROWSE is to help you become familiar with the data in an unfamiliar data base. The BROWSE softkey is displayed after the data base is opened. After pressing it this screen is displayed:



You can now specify the set to be examined by entering its corresponding number. After you enter the appropriate set number and press RETURN, entries from that set are displayed on the screen. If the entries do not fit on one screen, you can see the rest of the entries by pressing the NEXT PAGE softkey. Press the ABORT softkey when you have finished looking at the entries.

Exercise #1: Browsing Through the BORROWER Data Set

To browse through the BORROWER data set, you first must open the LIBRARY data base (as discussed previously), which displays these softkeys:

SHOW BASE INFO	BROWSE	SEARCH	UPDATE	CONFIGURE SYSTEM	FORMAL COMMAND	MORE KEYS	CLOSE DATA BASE

1. Now press the BROWSE softkey.
2. Specify the BORROWER data set by typing the number 6 and press RETURN.
3. As entries appear on the screen, press the NEXT PAGE softkey to advance through the entries.

The first two entries displayed are:

```
Browsing in set BORROWER.  Entry number 1 of 61.

        EMPLOYEE_NO: 4191
      BORROWER_NAME: ATKINSON,MAGGIE
           LOCATION: PERSONNEL
     EMPLOYEE_PHONE: (408) 257-7000


Browsing in set BORROWER.  Entry number 2 of 61.

        EMPLOYEE_NO: 7822
      BORROWER_NAME: AYLER,SUSAN
           LOCATION: PERSONNEL
     EMPLOYEE_PHONE: (714) 487-4100
```

While browsing the BORROWER data set, you might notice that JAN HARRISON's name is in the data set and therefore, she has checked out a book. If you want to find out the particular books checked out by Jan, then the SEARCH subsystem is used as explained in the following section.

When you reach the eighth entry in the data set, press the ABORT softkey to leave the BORROWER data set. Now that you have seen the contents of the BORROWER set, BROWSE some other data sets. Use the DUMP SCREEN softkey to get a copy of the screen printed on the internal printer. When you are ready to get back to the beginning of QUERY/9000, press the EXIT softkey.

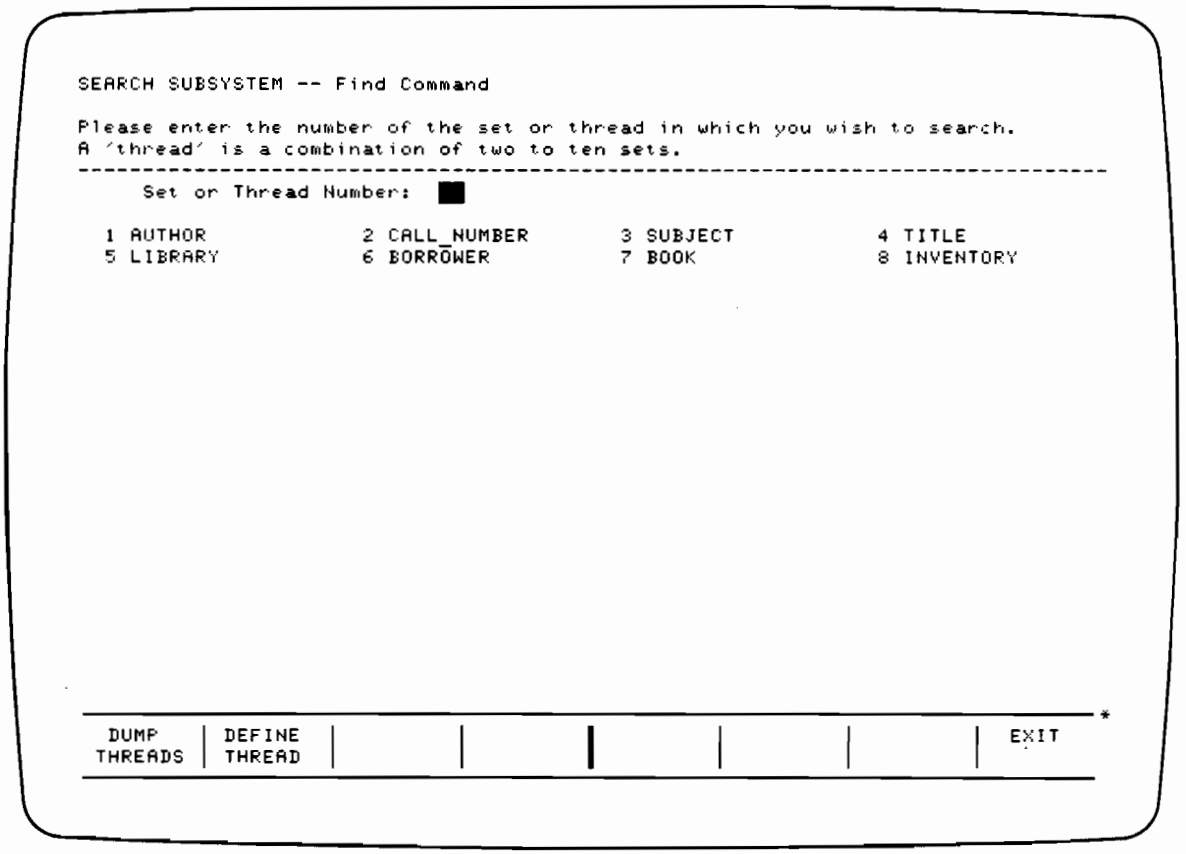
Retrieving Data Meeting Certain Criteria

Once you have used BROWSE to become familiar with the data in the data base, you can use the SEARCH softkey to retrieve selected data entries. A **search expression** specifies the criteria for selecting the entries. For example, you could retrieve all the books by the author William M. Newman by creating this search expression:

AUTHOR = WILLIAM M. NEWMAN

After selected data is retrieved, the SEARCH subsystem also enables you to sort and print this data.

The SEARCH softkey is displayed after the data base is opened. After pressing it, this screen is displayed:



You can now specify the data set you want to search. After you type the number that corresponds to a set, the screen is cleared and you can type the search expression. A search expression can be specified in many different ways; these are explained in Chapter 5.

Exercise #2: Searching for Books on Computer Programming

To search for books on computer programming, the LIBARY data base must be opened and these softkeys displayed:

1. Press the SEARCH softkey.
2. Enter 7 to indicate that BOOK is the data set to be searched.
3. Press the PROMPT softkey
4. Type SUBJECT for the item name and press RETURN.
5. Type 1 to indicate the equal sign (=) and press RETURN.
6. Type COMPUTER PROGRAMMING for the value and press RETURN.

   The search expression now looks like:

   SUBJECT = COMPUTER PROGRAMMING

7. Type 3 to indicate that the search expression is complete and press RETURN.

When the search is complete, the screen displays:

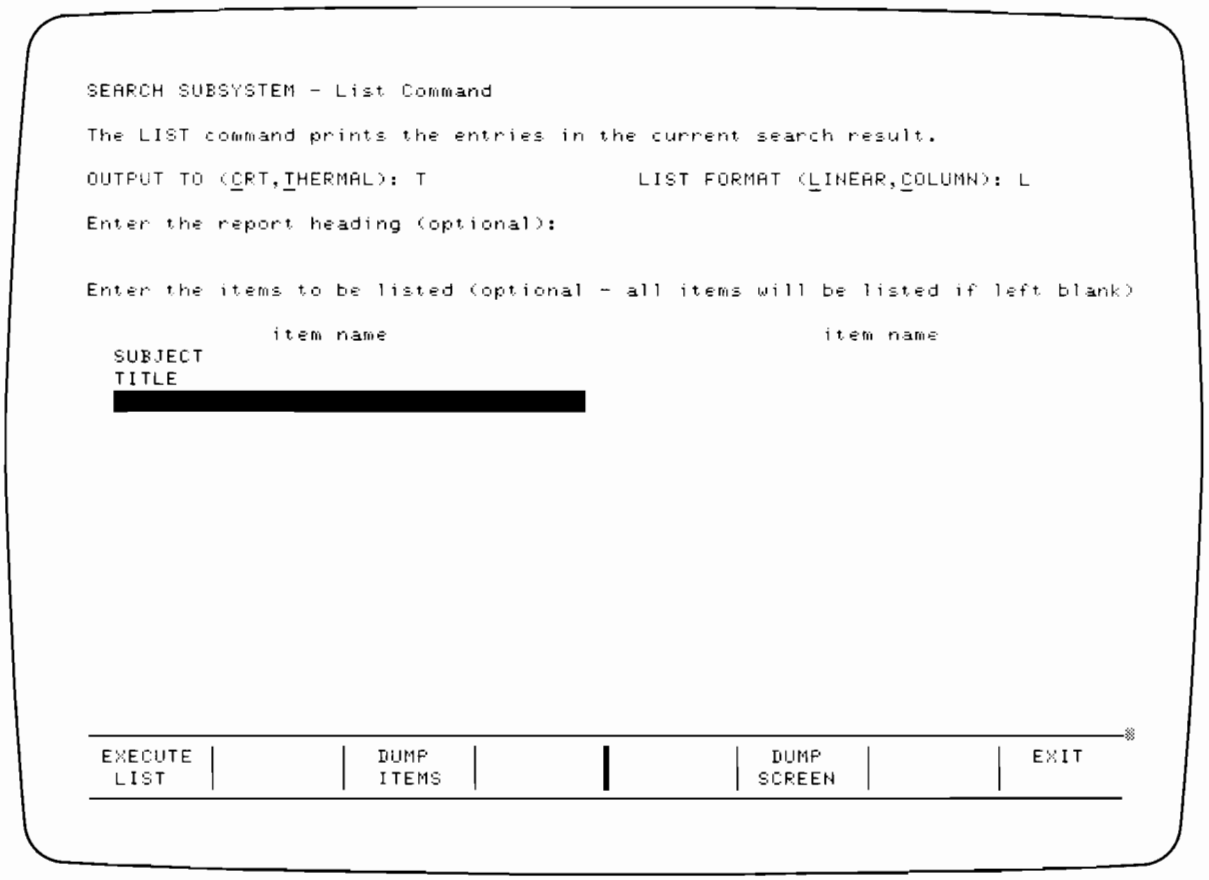
Size of search result: 12

This indicates that 12 data entries were found with the SUBJECT = COMPUTER PROGRAMMING. Now that a search has been performed, these softkeys are displayed:

Exercise #3: Listing the Books on Computer Programming

In order to see the titles of the books found in the previous exercise:

1. Press the LIST RESULT softkey.
2. Fill in the LIST COMMAND screen as shown next. This causes the subjects and titles of the selected books to be printed.
3. Press the EXECUTE LIST softkey.
4. Press EXIT to get out of the LIST command.



How many of the titles, out of the 12 data entries found, actually had the word COMPUTER in them? You should find four:

FUN AND GAMES WITH THE COMPUTER
TECHNIQUES IN COMPUTER PROGRAMMING
STANDARDIZED DEVELOPMENT OF COMPUTER SOFTWARE
THE PSYCHOLOGY OF COMPUTER PROGRAMMING

Listing the Data Entries

The LIST RESULT softkey enables you to examine all or selected entries found by the search. You can print the values in linear or columnar format with a optional heading printed at the top of the listing.

Exercise #4: Listing a Search Result

In this exercise, you will search the INVENTORY data set and find the entries indicating books checked out by employees with employee number 9905 or 2843. The resulting information is printed in columnar format.

1. Press the NEW SEARCH softkey.
2. Type `8` to indicate the INVENTORY data set.
3. For the search expression, type:

   ```
   EMPLOYEE_NO = 9905 OR EMPLOYEE_NO = 2843
   ```

   and press RETURN.
4. When the search result is found (4 entries), press the LIST RESULT softkey to examine these entries.
5. Type `C` to indicate the listing to be on the CRT, and type `C` to indicate columnar format.
6. For the heading to be printed at the top of the pages, type the following in the HEADING field.

   ```
   THE DATES THAT EMPLOYEES 9905 AND 2843 CHECKED OUT BOOKS
   ```
7. Now type `EMPLOYEE_NO` and `BORROW_DATE` as the item values that are to be printed out.
8. Press the EXECUTE LIST softkey.

The listing looks like:

```
            THE DATES THAT EMPLOYEES 9905 AND 2843 CHECKED OUT BOOKS

EMPLOYEE_NO BORROW_DATE
----------- ---------------
       2843 13 Jun 1977
       9905 10 Jul 1979
       9905 10 Jun 1978
       2843 4 Feb 1977
```

9. Press the EXIT softkey to get out of the LIST command.

Sorting the Selected Data Entries

After a search has been performed, you may also sort the resulting entries into a certain order based on a particular item. The sort is performed in ascending order unless you specify descending order. Additionally, you can sort using more than one item. For example, the books found in Exercise #2 on computer programming could be sorted alphabetically by author and chronologically by publication date. If two or more books have the same author, these books would be sorted by publication date.

To perform a sort, press the SORT RESULT softkey which appears after a search has been performed. You are asked to enter the data items on which you want to sort. Remember that sorting works only on data entries that were found by the search.

Exercise #5: Sorting the Books in the BOOK Data Set

Now let's search for all the books in the BOOK data set that have a price less than $10.00, then sort these books by SUBJECT and AUTHOR in ascending order and print the result on the internal printer.

1. Press the NEW SEARCH softkey to indicate you want to perform a new search.

2. Type `7` to specify that the BOOK data set is to be searched.
3. To find all the books with prices less than $10.00, type the search expression as:

   ```
   PRICE < 10.00
   ```

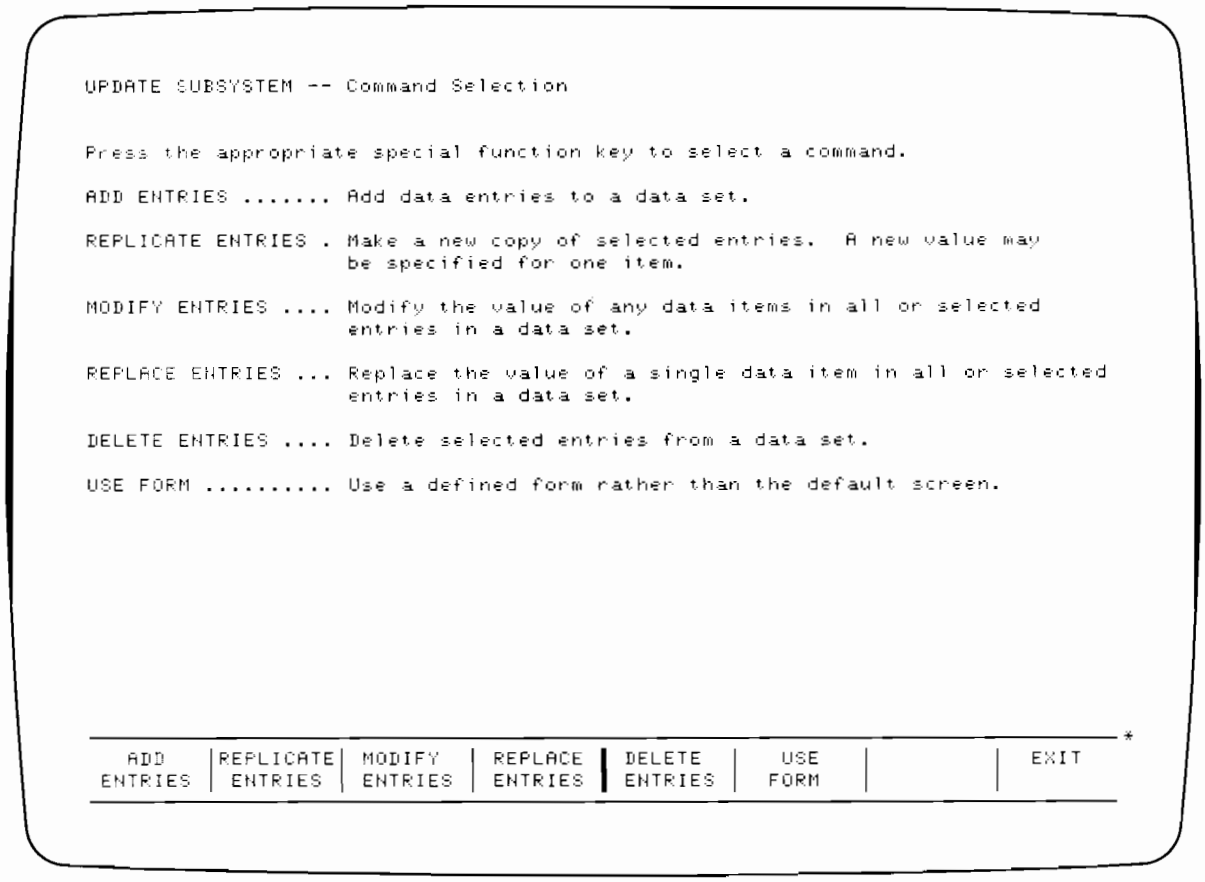
   and press RETURN.
4. When the search is complete the screen displays that 8 entries are in the search result.
5. Now press the SORT RESULT softkey to sort the search result
6. To indicate that you want the result sorted by SUBJECT and arranged in ascending order, type `SUBJECT` and press RETURN twice to bypass the ORDER field. Now type `AUTHOR` and press the EXECUTE SORT softkey. Remember, unless you specify descending order the sort is done in ascending order.
7. To print the sorted data, press the LIST RESULT softkey.
8. Type `T` to specify output to the thermal printer and type `L` to specify linear format.
9. Press the EXECUTE LIST softkey.

After the listing is done, notice the order of the books. All the subjects are in ascending alphabetical order and where the subject is the same for two books, the author is arranged in ascending order.

10. Press the EXIT softkey to get back to the SEARCH subsystem.
11. Now press the EXIT softkey to exit from the SEARCH subsystem and to return to the beginning of QUERY/9000.

Changing the Data in the Data Base

The UPDATE softkey enables you to change the data in a data set by adding complete entries, deleting complete entries, or changing data item values. When you press the UPDATE softkey after you open the data base, this screen is displayed:



Adding Data Entries

Pressing the ADD ENTRIES softkey enables you to add entries to a manual master or a detail data set. You need to specify the data set to which you want to add entries. After a set is selected, the item names in that set are displayed, and now you may enter the values. After you have typed a value, press RETURN, TAB or the ↓ (down arrow key). If you need to move the cursor to a previous field, press the SHIFT key and the TAB key together, or press ↑ (up arrow key). After all the values have been entered, press the ADD ENTRY softkey to store the entry. Once the entry is stored, the screen is cleared and another entry can be added.

Exercise #6: Adding a New Book to the BOOK Data Set

In this exercise, you can add a new book to the BOOK data set. This can be done after the UPDATE softkey is pressed.

1. Press the ADD ENTRIES softkey.
2. Enter `4` to indicate the BOOK data set is the one to which data is to be added.
3. Type the information about the book as described next, pressing RETURN after each data item value.

   Title: `THE MYTHICAL MAN-MONTH`
   Call number: `744714`
   Author: `FREDERICK BROOKS`
   Subject: `COMPUTER PROGRAMMING`
   Date Published: `22 Aug 1975`
   Publisher: `ADDISON-WESLEY`
   Price: `315.30`
4. Press the ADD ENTRY softkey to store the entry into the data base.
5. If you wish, you may add another book. If not, press the OPTION SELECTION softkey to return to the beginning of the UPDATE subsystem.

Deleting Data Entries

Pressing the DELETE ENTRIES softkey enables you to select and delete entries from a data set. Only complete data entries can be deleted; you cannot delete individual data item values. For example, all the data about a book has to be deleted, not just the publisher or the price.

To delete an entry, you must first specify the data set from which you want to delete entries. Next, specify which entries are to be deleted by entering a search expression. You can either delete all the entries found or delete only certain entries.

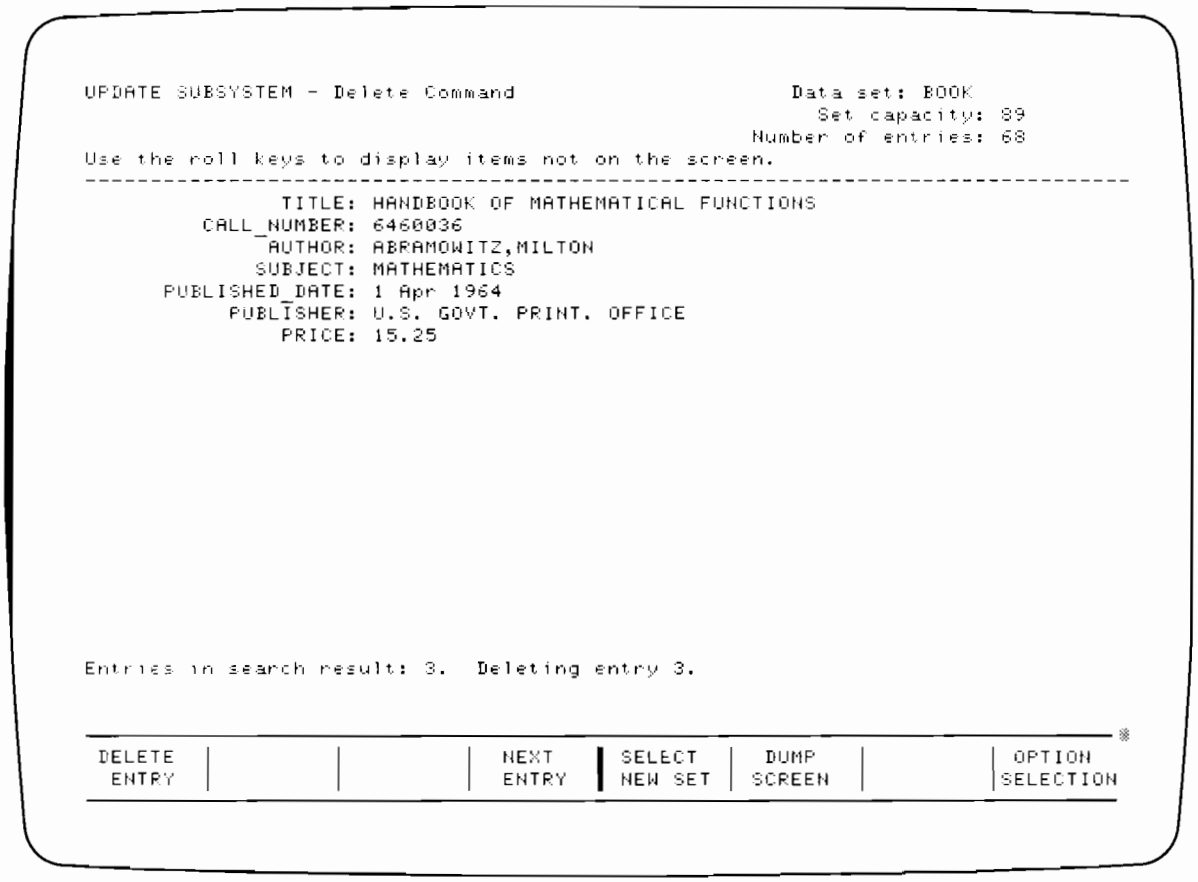
Exercise #7: Deleting a Book from the BOOK Data Set

In this exercise, you can delete the entry in the BOOK data set which has the subject MATHEMATICS and price 15.25.

1. Press the DELETE ENTRIES softkey.
2. Type `4` to indicate the BOOK data set.
3. Type `SUBJECT = MATHEMATICS` for the search expression and press RETURN.
4. Press the PAUSE DELETE softkey to view each of the three entries in order to find the exact entry.

If the entry displayed on the CRT is not the one you want to delete, press the NEXT ENTRY softkey to view the next entry in the search result.

5. Press the DELETE ENTRY softkey when the following entry is displayed on the CRT.



Note

Another way to find the above entry is to use:

SUBJECT = MATHEMATICS AND PRICE = 15.25

as the search expression. This returns one entry in the search result and then you can press the DELETE ALL softkey. Using this search expression saves you time since you don't have to examine each entry looking for the particular entry you want to delete.

6. Press the OPTION SELECTION softkey to return the program to the beginning of the UPDATE subsystem.

Other Capabilities of QUERY/9000

As explained earlier in this chapter, QUERY/9000 enables you to BROWSE through the data, SEARCH for and list specific data, and UPDATE data that is found in a data base. There are several other features associated with QUERY/9000, but they are more complex than BROWSE, SEARCH, and UPDATE. This section briefly discusses these additional features in general terms so that you are aware that they exist. Detailed explanations of each feature are found in the following chapters.

Using Forms with Your Data Base

A **form** is an image on the CRT used while adding or modifying entries. You can define a form for any data set except an automatic master data set (since data in automatic masters is maintained automatically by the IMAGE/DBM System). By defining your own form, you can display additional information about the data to be entered. For example, you can clarify the CALL_NUMBER item by also specifying that it is a number with a maximum of 9 digits. Forms are explained in Chapter 7.

Defining a Data Base

When creating a new data base, you must first design it (draw the data base diagram) and then interactively communicate the definition to QUERY/9000. You can use QUERY/9000 to interactively define or redefine a data base. The Data Base Design Kit helps you to develop a data base diagram; Chapter 8 explains how to define and create a data base using QUERY/9000.

Using the Formal Command Language

QUERY/9000 enables you to use the formal command language to perform QUERY/9000 operations rather than using the softkeys. Formal commands can be used when you are familiar with an operation, thus saving the time taken by using softkeys. After you open a data base, there is a softkey labeled FORMAL COMMAND; pressing it enables you to type and execute the formal commands. For example, you could type:

```
FIND SUBJECT = COMPUTER PROGRAMMING
    and
LIST TITLE, AUTHOR
```

instead of using the softkeys. The command language is explained in greater detail in Chapter 10.

Linking Other Programs to QUERY/9000

Another feature of QUERY/9000 enables you to execute subprograms which are written to extend the feature of QUERY/9000. For example, if you wrote a program which produces formatted reports, QUERY/9000 could use this program to output data in a data base. You would open the data base, enter the SEARCH subsystem, and establish a search result. Then, by pressing the RUN USER PROGRAM softkey, QUERY/9000 calls your reporting subprogram and the search result is used as the data for reporting.

There are two places within QUERY/9000 that a subprogram can be linked. One is at the beginning of QUERY/9000, after you have opened the data base, and the other is in the SEARCH subsystem. Extensions to QUERY/9000 should be written by experienced programmers. Anyone can use the extension once it is written. The details about RUN USER PROGRAM are found in Chapter 9.

UTILITIES

QUERY/9000 has a subsystem which allows you to backup a data base so that there is a "snapshot" of the information in your data base. In the event of a system failure which could corrupt information on a disc, this "snapshot" can be RECOVERED so that not all the information is lost. BACKUP should be done on a regular basis. The UTILITIES subsystem also allows you to ERASE, PURGE, LOAD and UNLOAD data bases. These features are described in Chapter 11.

Chapter 3
Data Base Concepts

If this is the first time you have used the IMAGE/DBM System, you may encounter many unfamiliar terms. This chapter defines these terms and concepts as they apply to QUERY/9000.

Data Base

A **data base** is a collection of related information that has been stored in a logical manner so that data can be easily accessed. For example, you can think of a company's library as being a data base with information about the books, who borrowed the books, and some information about each company's branch library.

Data Item

A **data item** is the smallest accessible data element in the data base. Each data item is given a value. In the library data base, examples of data items are the title, price, and author of a book.

Data Item	Data Item Value
CALL_NUMBER	5712902
TITLE	ENGINEERING ELECTROMAGNETICS
AUTHOR	WILLIAM HAYT
SUBJECT	ELECTROMAGNETICS THEORY
PUBLISHED_DATE	DECEMBER 26, 1956
PUBLISHER	MCGRAW-HILL
PRICE	8.50

Each data item has a name associated with it. For example, the data item containing the title of a book has the data item name TITLE. The format for data item names is found in Chapter 8.

Pseudonyms

When referring to data items, QUERY/9000 enables you to use pseudonyms for the item name. A **pseudonym** is a name that can be used interchangeably with the item name as a memory aid or a typing aid, such as one or two letter abbreviations for a long item name. For example, the item name TITLE has the pseudonyms, HEADING and NAME_OF_BOOK. A pseudonym can be used instead of the item name TITLE while using QUERY/9000. Up to five pseudonyms may be defined for each data item.

Types of Data Items

There are six types of data items used with QUERY/9000. QUERY provides the FORMAT specifier so that strings may be interpreted as NAMES, and R8 items may be interpreted as DATE, TIME or TIMEDATE formats.

Character string (X) – a string of ASCII characters. A table of ASCII characters is found in the Appendix. A maximum length must be specified for each string.

Integer (I2) – a whole number in the range –32 768 through 32 767.

Double (I4) – a whole number in the range –2 147 483 647 through +2 147 483 647.

Short Numeric (R4) – a short precision number with seven significant digits and an exponent in the range –38 and +38.
(Actual range: –3.402E+38 through 1.175E–38,0, +1.175E–38 to +3.402E+38.) and absolute value from 1.175E–38 through 3.402E+38

Long Numeric (R8) – a full precision real number with sixteen significant digits and an exponent in the range –308 and +308.
(Actual range: +1.797 693 134 862 315E+308 through –2.222E–308,0, +2.222E–3308 through +1.797E+308. and absolute value from 2.222E–308 through 1.797E+308.)

Code (C2) – enables you to establish a set of 32 allowable 16 character code values for a data item.

These types are explained in greater detail in Chapter 8.

NAME Format

The **NAME Format** is used to store personal names in a consistent format. Names can be entered in one of two ways:

"last name, first name [middle name] [,suffix]"
or
"first name [middle name] last name [,suffix]"

No matter which way a name is entered, it is stored by QUERY/9000 using the first format shown above. The part of the name enclosed in square brackets is optional. Some examples of how QUERY/9000 converts names are:

Sam Smith	is converted to	SMITH,SAM
SAM SMITH, JR.	is converted to	SMITH,SAM,JR.
SAM E. SMITH, JR.	is converted to	SMITH,SAM E.,JR.
MR. & MRS. SAM E. SMITH	is converted to	SMITH,MR. & MRS. SAM E.

If a person's last name has two or more parts, such as Von Grendy, the last name should be entered with the underscore character (_) between the parts of the name, such as Von_Grendy.

DATE, TIME and TIMEDATE

The BOOK data set has a data item (PUBLISHED_DATE) of **type R8** and the **DATE** format. This enables you to enter dates for this item in the standard BASIC format DD MMM YYYY and HH:MM:SS. QUERY uses the TIMEDATE format in BASIC to convert these back and forth from the R8 format. The advantage of using the Date type is that the dates can be conveniently searched and sorted in chronological order.

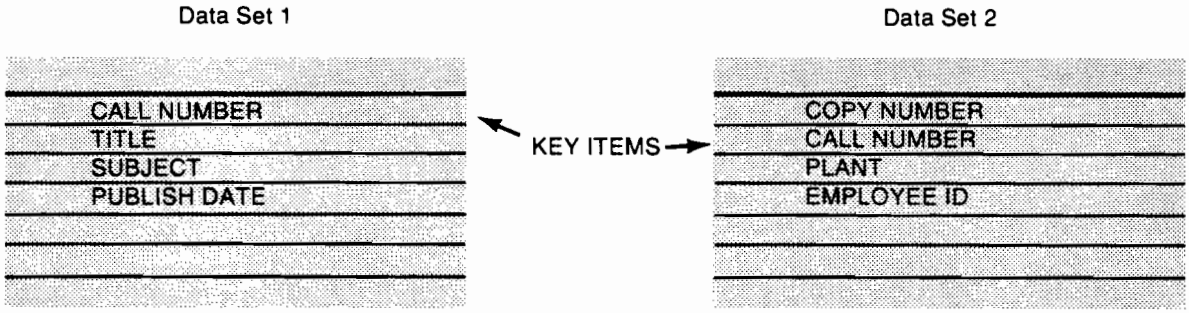
Arrays

In the LIBRARY data set there is an item named PLANT_ADDRESS. This item is defined with DIMENSION: 1:3. The 3 indicates that this item is a three element array. QUERY can handle up to six dimensions when arrays are specified.

PLANT_ADDRESS has three elements, each one being 40 characters long. One is for the address and street, the second is for city and state, and the third is for the zip code. An array item is used when you want to perform a search using just one part of an item. For example, to find the plant in Fort Collins, CO, you can search on ADDRESS(2) which contains the plant's city and state. The 2 in parentheses indicates the second element.

Key Item

A **key item** is used to **identify** data entries for quick **retrieval**. For example, a book can be identified by its call number, so this is defined to be a key item. Additionally, since one use of the LIBRARY data base is to retrieve all books on a particular subject, SUBJECT is defined to be a key item. Key items also link data sets together as illustrated next. A key item cannot be an array or a string longer than 256 characters.



Ranges

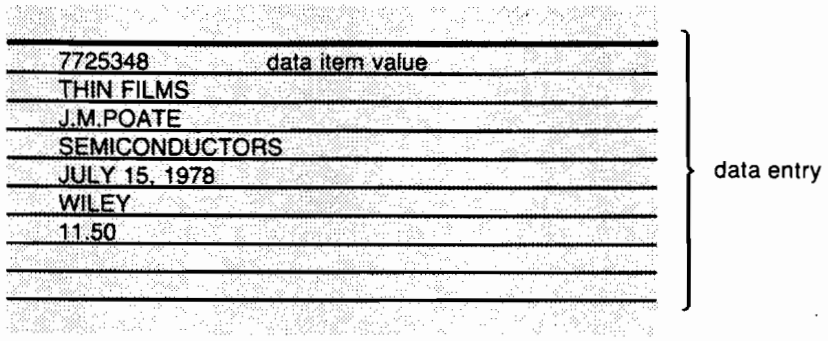
Each numeric or date data item can be given a **range** which indicates the upper and lower bounds of allowable values. For example, the data item EMPLOYEE_NO should always be a positive number, so the range is defined as 0 to 999 999 999 instead of the default range. When a data item value is entered, its value is checked to see if it is within the range. If the value isn't within the range, an error message is displayed and a new value must be entered.

Null Values

When a value is not entered for a data item, a **null value** is stored. The default null value is blank for strings and is 0 for numbers. The data base designer may explicitly specify a null value in SCHEMA or DEFINE. This may be any numeric value.

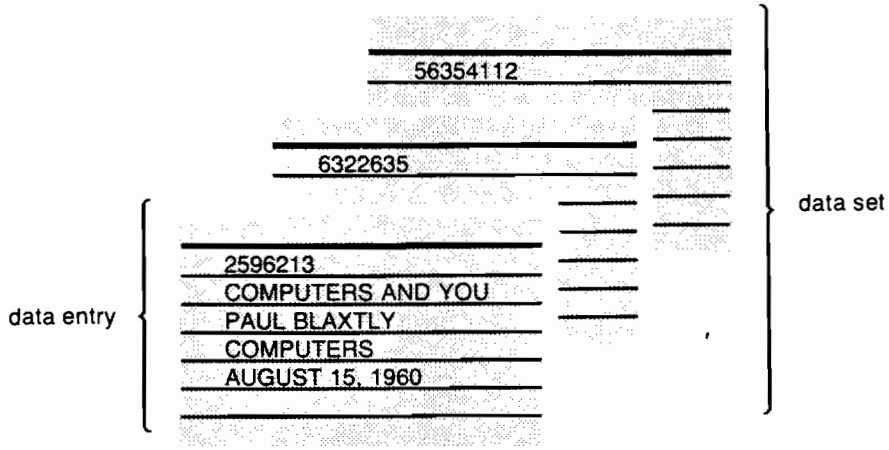
Data Entry

A collection of related data item values is called a **data entry**. An entry can be thought of as a card found in the card catalog at the library. For example, here is a data entry for the BOOK data set:



Data Set

A collection of similar data entries is called a **data set**. The next example contains three data entries which together make up a particular data set, the BOOK data set.



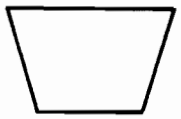
Data Set Types

There are two types of data sets used by QUERY/9000: **detail sets** and **master sets**. **Detail sets** contain the bulk of the information. A **master set** is generally used as an index for fast access to the information in detail sets. There are two types of master data sets: **automatic** and **manual**.

Each master data set contains one key item. Each detail data set can contain up to 31 key items and other non-key items. Each of the key items in a detail set is linked to the key item in one of the master data sets. Key items can be used to link together the information in two or more data sets. A detail data set need not contain any key items if you do not wish to index or link it.

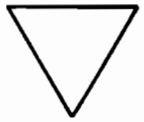
Detail Data Sets

A **detail data set** generally contains data items which describe one "thing" about which information is kept. For example, the BOOK data set contains data about the books, but nothing about the library branches or borrowers.



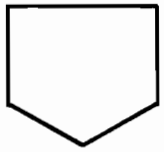
Automatic Master Data Sets

An **automatic master data set** contains one data item, the key item. (Remember that a key item is an item common to two or more sets to "link" master and detail sets together.) When a new value is added for the same key item in a detail data set, the value is automatically added to the automatic master data set. Deletions from the automatic master data set are made when the last data entry with the particular key item value is deleted from the detail data set. Automatic masters are used when key item values are too numerous or unpredictable, making manual entry unreasonable.



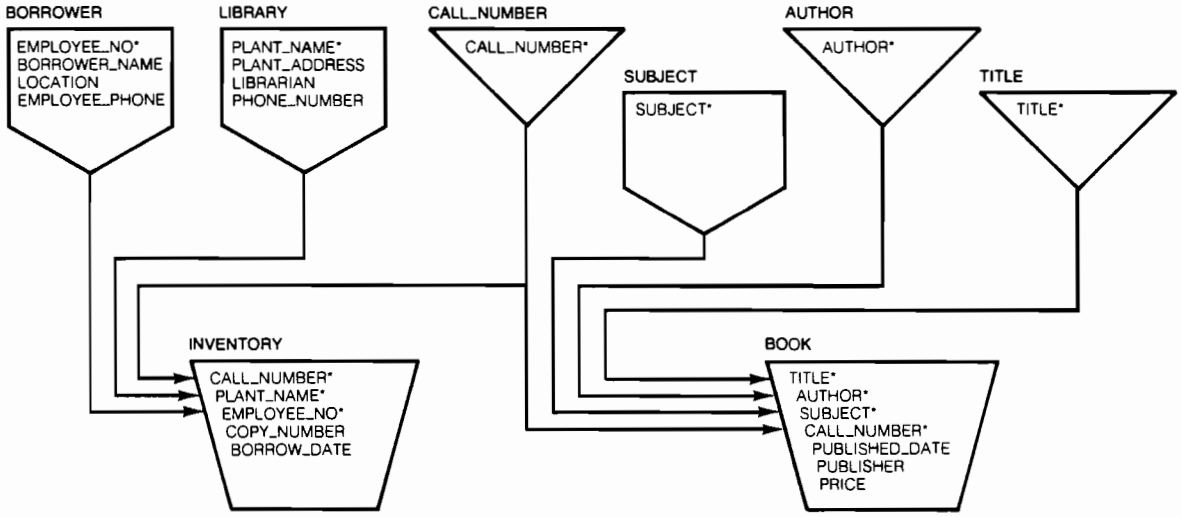
Manual Master Data Sets

A **manual master data set** can contain one or more data items in addition to the one key item. Before a new key item value can be added to a detail data set that is linked to a manual master data set, the value must first be added to the manual master data set. This enables you to control what data is entered into a manual master data set since you have to enter the values. Typing errors in a manual master data entry won't be automatically entered as they would if you were using a automatic master data set.



The LIBRARY Diagram

The following diagram illustrates the LIBRARY data base using the "boxes" defined for each data set. It is similar to the diagram produced by the Data Base Design Kit and shows all of the different sets and their relationships.



Chapter 4
Basic Operations

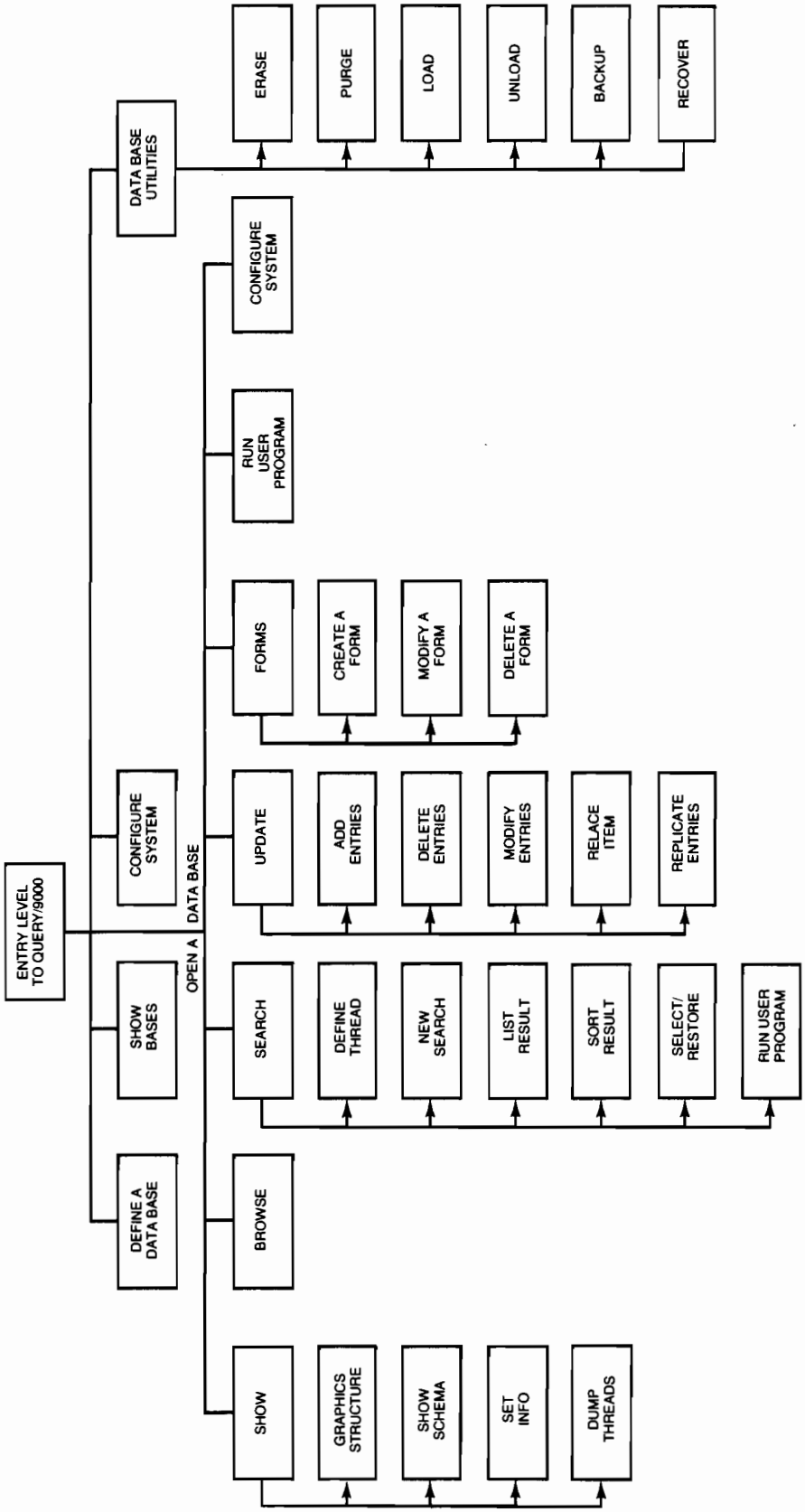
Introduction

Chapters 4 through 7 provide detailed explanation of the QUERY/9000 features and capabilities. An overview of these features was presented by the exercises in Chapter 2. The following chapters contain helpful hints and information necessary for the use of QUERY/9000. This chapter contains:

- an overview of QUERY/9000.
- a review of general features of QUERY/9000.
- an explanation of basic data base operations: opening and closing the data base and looking at the structure and contents of the data base. (SHOW BASE INFO and BROWSE)

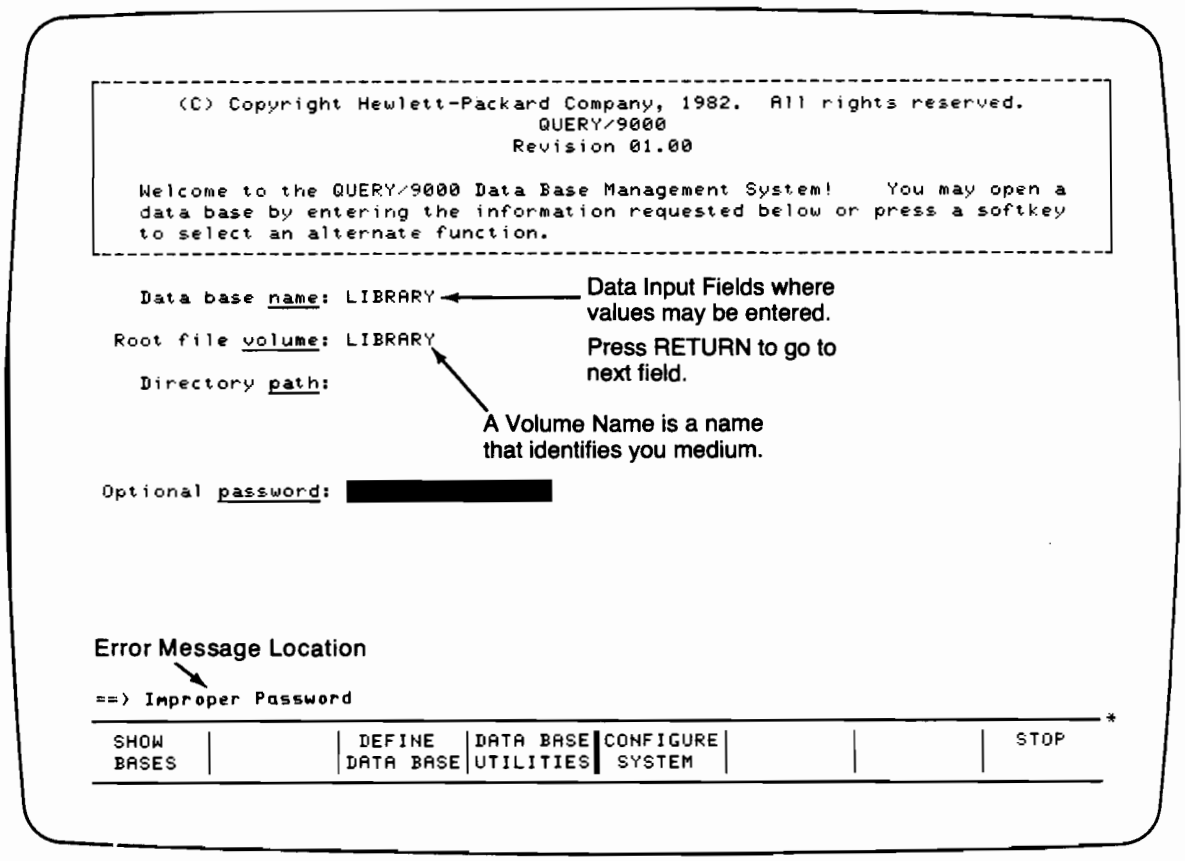
Overview of QUERY/9000

QUERY/9000 is organized as a **tree structure** enabling you to branch to subsequent levels, then return to access other features. The top of the tree enables you to open a data base, define a data base, configure your system, or run utilities such as BACKUP and RECOVER. If you choose to open a data base, you then can branch down to perform different operations on the data base. When you want to exit from the program, you have to exit back up through the levels of the tree structure to close the data base and stop the program. Following is a general diagram of QUERY/9000 showing the levels of the tree structure:


ENTRY LEVEL TO QUERY/9000
OPEN A DATA BASE
DEFINE A DATA BASE
CONFIGURE SYSTEM
SHOW BASES
DATA BASE UTILITIES
ERASE
PURGE
LOAD
UNLOAD
BACKUP
RECOVER
CONFIGURE SYSTEM
RUN USER PROGRAM
FORMS
CREATE A FORM
MODIFY A FORM
DELETE A FORM
UPDATE
ADD ENTRIES
DELETE ENTRIES
MODIFY ENTRIES
RELACE ITEM
REPLICATE ENTRIES
SEARCH
DEFINE THREAD
NEW SEARCH
LIST RESULT
SORT RESULT
SELECT/ RESTORE
RUN USER PROGRAM
BROWSE
SHOW
GRAPHICS STRUCTURE
SHOW SCHEMA
SET INFO
DUMP THREADS

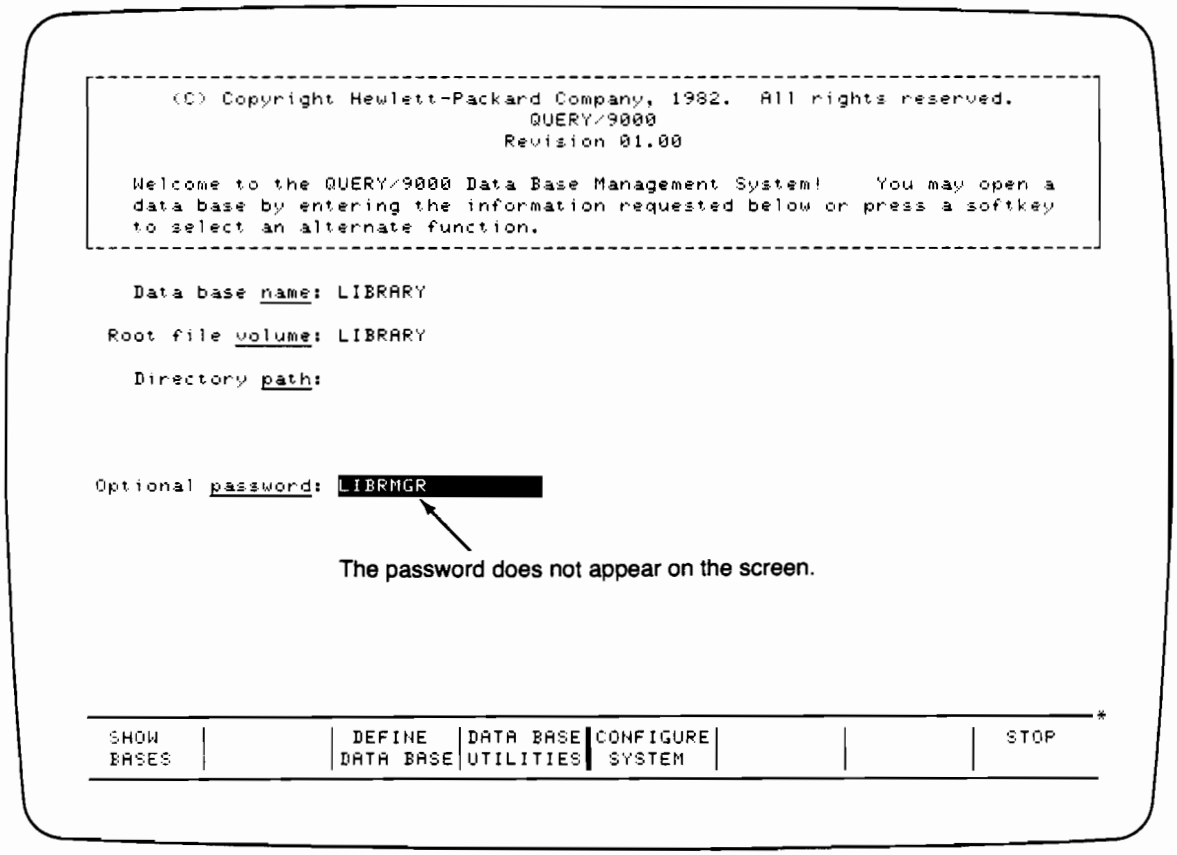
General QUERY/9000 Features

QUERY/9000, has a number of features designed to make using the program as simple as possible. These features (softkeys, fields, error message, location and volume names) are described in Chapters 1 and 2, and are summarized in the following diagram.



Accessing a Data Base

In order to access a data base with QUERY/9000, you must first open the data base. This operation involves specifying the name of the data base, the volume name of the device which contains the root file of the data base, the directory path, and a password. When you start running the QUERY/9000 program, you can open the data base by filling in the screen as shown below:



The reverse of opening a data base is closing a data base. QUERY/9000 performs this operation automatically when you press the CLOSE DATA BASE softkey.

Looking at the Data Base

After you have opened a data base, the SHOW BASE INFO softkey is used to obtain information about the data base in four different ways.

- The GRAPHIC STRUCTURE softkey prints a diagram of the data base on the printer.
- The SCHEMA softkey prints the schema, or formal definition of the data base.
- The SET INFO softkey prints information about the individual sets.
- The THREADS softkey prints the threads that have been defined while using QUERY/9000.

Graphic Structure

The diagram produced by the GRAPHIC STRUCTURE softkey is useful during other data base operations to get information like data item names, quickly. If the password specified does not have access to some of the data sets, these sets are not printed. Within this diagram, the data set names, data items and their types, the data set capacity, and the data paths are shown in the diagram.

Set Information

If you just want to know about the data sets in a data base, use the SET INFO softkey. For each data set the following information appears on the CRT or is printed: the data set name, the type of set, the volume name, how many entries are in the set and the capacity of the set. You specify where you want the information to be printed.

The Data Base Schema

After pressing the SHOW BASE INFO softkey, the SCHEMA softkey is displayed. Pressing SCHEMA prints all the data sets, items, codes, pseudonyms and ranges of the data base in a format similar to the schema, or formal definition, used by IMAGE/9000. The information can be printed on an external printer, the internal printer or the CRT. Here is part of the information about the LIBRARY data base obtained by pressing SCHEMA:

```
     Data Base: LIBRARY
        Volume: QUERY
Directory Path: QUERY
 Lexical Order: ASCII
================================================================================

   Item Number: 1
     Item Name: AUTHOR, X25
   Item Format: NAME
    Pseudonyms: NOVELIST,WRITER,POET,AUTHOR_NAME
--------------------------------------------------------------------------------
   Item Number: 2
     Item Name: BORROW_DATE, R8
   Item Format: DATE
    Pseudonyms: DATE_BORROWED
         Range: 1 Jan 1900 TO 1 Jan 2000
    Null Value: 1 Jan 0
--------------------------------------------------------------------------------
   Item Number: 3
     Item Name: BORROWER_NAME, X25
   Item Format: NAME
    Pseudonyms: BORROWER
--------------------------------------------------------------------------------
```

```
   Item Number: 4
     Item Name: CALL_NUMBER, I4
   Item Format: M9D
         Range: 0 TO 999999999
    Null Value: 0
-------------------------------------------------------------------------------
   Item Number: 5
     Item Name: COPY_NUMBER, I2
   Item Format: M2D
    Pseudonyms: COPY,COPY_NO,NUMBER_OF_COPY
         Range: 0 TO 20
    Null Value: 0
-------------------------------------------------------------------------------
   Item Number: 6
     Item Name: EMPLOYEE_NO, I4
   Item Format: M9D
    Pseudonyms: EMPLOYEE
         Range: 0 TO 999999999
    Null Value: 0
-------------------------------------------------------------------------------
   Item Number: 7
     Item Name: EMPLOYEE_PHONE, X14
   Item Format: USER(0)
    Pseudonyms: PHONE_NO
-------------------------------------------------------------------------------
   Item Number: 8
     Item Name: LIBRARIAN, X25
   Item Format: NAME
    Pseudonyms: HEAD_LIBRARIAN
-------------------------------------------------------------------------------
   Item Number: 9
     Item Name: LOCATION, C2
   Item Format: USER(0)
    Pseudonyms: AREA_LOCATED,DEPARTMENT
         Codes: MANAGEMENT, MARKETING, PRODUCTION, R&D, FINANCE,
                PERSONNEL, QA
-------------------------------------------------------------------------------
   Item Number: 10
     Item Name: PHONE_NUMBER, X14
   Item Format: USER(0)
    Pseudonyms: PHONE,TELEPHONE,TELEPHONE_NO,LIBRARY_PHONE
-------------------------------------------------------------------------------
   Item Number: 11
     Item Name: PLANT_ADDRESS, X30
   Item Format: USER(0)
    Pseudonyms: PLANT_LOCATION,ADDRESS
     Dimension: 1:3
-------------------------------------------------------------------------------
   Item Number: 12
     Item Name: PLANT_NAME, X25
   Item Format: USER(0)
    Pseudonyms: LIBRARY_BRANCH,PLANT_LIBRARY
-------------------------------------------------------------------------------
   Item Number: 13
     Item Name: PRICE, R4
   Item Format: M3D.2D
    Pseudonyms: COST,BOOK_PRICE,PRICE_OF_BOOK,COST_OF_BOOK
         Range: 0.00 TO 500.00
    Null Value: -1.00
-------------------------------------------------------------------------------
   Item Number: 14
     Item Name: PUBLISHED_DATE, R8
   Item Format: DATE
    Pseudonyms: PUBLISH_DATE,DATE_PUBLISHED
         Range: 1 Jan 1900 TO 1 Jan 2000
    Null Value: 1 Jan 0
-------------------------------------------------------------------------------
```

```
   Item Number: 15
     Item Name: PUBLISHER, X30
   Item Format: USER(0)
    Pseudonyms: PUBLISHER_NAME
--------------------------------------------------------------------------------
   Item Number: 16
     Item Name: SUBJECT, X30
   Item Format: USER(0)
    Pseudonyms: TOPIC
--------------------------------------------------------------------------------
   Item Number: 17
     Item Name: TITLE, X50
   Item Format: USER(0)
    Pseudonyms: HEADING,NAME_OF_BOOK
================================================================================
    Set number: 1
      Set Name: AUTHOR
          Type: AUTO MASTER
        Volume: QUERY
Directory Path: QUERY
       Entries: 67/89

         Items: AUTHOR, X25

         Paths: AUTHOR is linked to BOOK.AUTHOR
--------------------------------------------------------------------------------

    Set number: 2
      Set Name: CALL_NUMBER
          Type: AUTO MASTER
        Volume: QUERY
Directory Path: QUERY
       Entries: 70/89

         Items: CALL_NUMBER, I4

         Paths: CALL_NUMBER is linked to BOOK.CALL_NUMBER
                CALL_NUMBER is linked to INVENTORY.CALL_NUMBER
--------------------------------------------------------------------------------

    Set number: 3
      Set Name: SUBJECT
          Type: MANUAL MASTER
        Volume: QUERY
Directory Path: QUERY
       Entries: 44/53

         Items: SUBJECT, X30

         Paths: SUBJECT is linked to BOOK.SUBJECT
--------------------------------------------------------------------------------

    Set number: 4
      Set Name: TITLE
          Type: AUTO MASTER
        Volume: QUERY
Directory Path: QUERY
       Entries: 68/89

         Items: TITLE, X50

         Paths: TITLE is linked to BOOK.TITLE
--------------------------------------------------------------------------------
```

```
    Set number: 5
      Set Name: LIBRARY
          Type: MANUAL MASTER
        Volume: QUERY
Directory Path: QUERY
       Entries: 8/13

         Items: PLANT_NAME, X25
                PLANT_ADDRESS, X30
                LIBRARIAN, X25
                PHONE_NUMBER, X14

         Paths: PLANT_NAME is linked to INVENTORY.PLANT_NAME
--------------------------------------------------------------------------------

    Set number: 6
      Set Name: BORROWER
          Type: MANUAL MASTER
        Volume: QUERY
Directory Path: QUERY
       Entries: 61/79

         Items: EMPLOYEE_NO, I4
                BORROWER_NAME, X25
                LOCATION, C2
                EMPLOYEE_PHONE, X14

         Paths: EMPLOYEE_NO is linked to INVENTORY.EMPLOYEE_NO
--------------------------------------------------------------------------------

    Set number: 7
      Set Name: BOOK
          Type: DETAIL
        Volume: QUERY
  Directory Path: QUERY
       Entries: 68/89

         Items: TITLE, X50
                CALL_NUMBER, I4
                AUTHOR, X25
                SUBJECT, X30
                PUBLISHED_DATE, R8
                PUBLISHER, X30
                PRICE, R4

         Paths: TITLE is linked to TITLE.TITLE
                CALL_NUMBER is linked to CALL_NUMBER.CALL_NUMBER
                AUTHOR is linked to AUTHOR.AUTHOR
                This path is sorted by TITLE
                SUBJECT is linked to SUBJECT.SUBJECT
  --------------------------------------------------------------------------------

    Set number: 8
      Set Name: INVENTORY
          Type: DETAIL
        Volume: QUERY
  Directory Path: QUERY
       Entries: 148/193

         Items: CALL_NUMBER, I4
                COPY_NUMBER, I2
                PLANT_NAME, X25
                EMPLOYEE_NO, I4
                BORROW_DATE, R8

         Paths: CALL_NUMBER is linked to CALL_NUMBER.CALL_NUMBER
                PLANT_NAME is linked to LIBRARY.PLANT_NAME
                EMPLOYEE_NO is linked to BORROWER.EMPLOYEE_NO
  ================================================================================
```

Listing the Defined Threads

A thread is required when performing a SEARCH that uses information from more than one data set. The THREADs softkey lists all the existing threads for the specified data base. Thread definition is covered in Chapter 5.

Browsing Through the Data

If you want to know what entries are found in a particular data set, press the BROWSE softkey at the beginning of QUERY/9000. After you specify the set to be examined, the entries are displayed one at a time on the CRT. The entries are read serially from the data set and are displayed in that order. The BROWSE softkey can help you become familiar with the data in an unfamiliar data base.

Chapter 5
Retrieving Data Meeting Certain Criteria

Introduction

The SEARCH subsystem enables you to retrieve selected data entries from the data base, then display, list, sort, or update them. The desired entries are retrieved by means of "translating" a natural language question into something that QUERY/9000 can understand. This is done by using a **search expression** which consists of an item name, a relational operator, and a value or an item name. For example, if you want to know what author wrote the book Programming Proverbs, the search expression would look like:

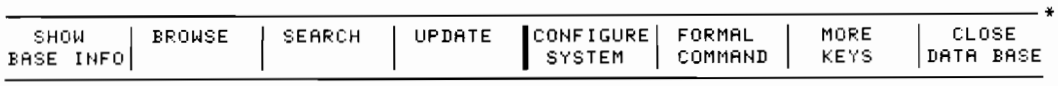
TITLE = PROGRAMMING PROVERBS

It is helpful to be familiar with the data base when using the SEARCH subsystem. If you do not know what type of data is stored in the data base, use the BROWSE feature, which is explained in Chapter 4.

The **search result** consists of pointers to the entries which meet the search criteria. These pointers are stored in a temporary file created by QUERY.

Accessing the SEARCH Subsystem

The following softkeys are displayed at the beginning of the SEARCH subsystem. You can then press the SEARCH softkey to specify a search expression.



After a search has been performed, the following softkeys are displayed. Another search can be performed by pressing the NEW SEARCH softkey.

If you are familiar with QUERY/9000, you can use the formal command mode to perform a search. This enables you to save time by typing the command rather than using the softkeys. If you're not sure about the information needed to make up a command, you can press the PROMPT softkey which helps you compose the expression. A listing of all the formal commands is found in Chapter 10.

Performing a Search

A search is implemented using a **search expression** to select data entries. The simplest form of a search expression is:

item relational operator value or item you are searching for

A few examples of simple search expressions are:

```
AUTHOR WITH LASTNAME ANDERSON
PRICE > 15.35
LOCATION = MARKETING
```

The following relational operators can be used in search expressions:

Relational Operator	Explanation
= , IS	retrieves entries whose item value exactly matches the search value.
< >,#,IS NOT	retrieves entries where the item value and the search value do not match.
< , BEFORE	retrieves entries whose item value is less than the search value.
> , AFTER	retrieves entries whose item value is greater than the search value.
< =	retrieves entries whose item value is less than or equal to the search value.
> =	retrieves entries whose item value is greater than or equal to the search value.
FROM ... TO	retrieves entries with values in the specified range, including the boundary conditions named.
BETWEEN ... AND	retrieves entries with values in the specified range, excluding the boundary conditions named.
WITH	retrieves entries where the item value (character or name type) contains the string specified by the search value anywhere within it.
STARTING WITH	retrieves entries where the item value (character or name type) starts with the string specified by the search value.
ENDING WITH	retrieves entries where the item value ends with the specified string. (Leading and trailing blanks are trimmed before the test.)
WITH FIRSTNAME	retrieves entries where the item value (name type only) contains the value ","&search value.
WITH LASTNAME	retrieves entries where the item value (name type only) starts with search value&",".

Considerations

Any character-string data that is entered with lowercase letters is stored with lowercase letters. Similarly, uppercase are stored as uppercase. This is important when you are specifying the string for search expressions; for example, a capital letter (A) does not equal a lowercase letter (a).

Compound Searches

Sometimes it is convenient to search for entries that meet several conditions rather than just one. QUERY/9000 enables **compound search expressions** to be used when the search involves more than one search condition. Compound search expressions are formed by connecting two simple expressions together using the conjunctions "AND" and "OR".

The "AND" operator specifies that both criteria must be met for the search expression to be true. The "OR" specifies that if either criteria is met, the search expression is true. For example, if you type the search expression:

SUBJECT = ROBOTS AND PRICE > 20.00

and an entry has the SUBJECT = ROBOTS but the PRICE = 15.95, then the search expression is evaluated to be false. Only one of the criteria is met.

If you type this search expression:

SUBJECT = ROBOTS OR PRICE > 20.00

and find the same entry described above, the search expression is evaluated true. The "OR" specifies that only one criterion must be met (SUBJECT = ROBOTS).

Search expressions are evaluated left to right with "AND" taking priority over "OR", but parentheses can be used to establish a different order. For example, using the LIBRARY data base, you can type in the following compound search expression:

PUBLISHED_DATE > 1 JAN 1976 AND PRICE = 20.00 OR SUBJECT = ROBOTS

and press RETURN. When the search is complete, the search result contains twelve entries that have qualified. If you press the NEW SEARCH softkey, indicate the BOOK data set again and type:

PUBLISHED_DATE > 1 JAN 1976 AND (PRICE = 20.00 OR SUBJECT = ROBOTS)

the search result contains three entries. This shows that the SEARCH program evaluates two nearly identical searches with greatly varying results. In the first example, parentheses are implicity placed like:

(PUBLISHED_DATE > 1 JAN 1976 AND PRICE = 20.00) OR SUBJECT = ROBOTS

Some other examples of compound searches are:

PUBLISHER = MCGRAW-HILL OR PUBLISHER = ADDISON-WESLEY
PLANT_ADDRESS(3) IS 97330 OR LIBARIAN WITH LASTNAME ROSS,
PRICE FROM 10.00 TO 20.00 AND TITLE WITH COMPUTER

Multiple Values

If you are using a search expression involving the "=" or "WITH" operators, multiple values can be specified. For example, this search expression:

SUBJECT = MATHEMATICS ; INDUSTRIAL MANAGEMENT

is the same as:

(SUBJECT = MATHEMATICS OR SUBJECT = INDUSTRIAL MANAGEMENT)

If you have a multiple value search along with another search criteria such as:

PRICE = 15.00;20.00 AND SUBJECT = COLOR

then the multiple-value search is evaluated first. Multiple values have the highest priority in a compound search expression if parentheses are not used. For example, the previous example is the same as:

(PRICE = 15.00 OR PRICE = 20.00) AND SUBJECT = COLOR

Using Quotes in a Search Expression

You can use quotes when specifying names or character strings but they are not required. For example, the following search expressions perform the same search:

PLANT = DCD
PLANT = "DCD"

There are, however, a few cases where quotes are required.

- When a search string contains reserved words or characters such as AND, OR, TO, FOR, $NULL, ;,or). $NULL is used to check if the search item has no value.
- When using the equals operator (=) to search for a character string whose value contains leading or trailing blanks. For example, "MARKETING" does not equal " MARKETING ".
- If the string value contains any quote marks, they must be doubled and the entire string must be enclosed in quotes. For example, the string AB"C would be typed "AB""C".
- If the search string is the same as an item name or a pseudonym name, quotes are needed. For example, TITLE = "SUBJECT" finds the books where the title is the string "SUBJECT". TITLE = SUBJECT finds any books where TITLE and SUBJECT are identical.

Item Names

Item names can be used as the search value in a search expression. For example, if you want to check if the librarian has borrowed a book, you would type:

LIBRARIAN = BORROWER_NAME.

Null Values

A data item contains a **null value** if nothing is entered for the item using the UPDATE subsystem. The word $NULL is a reserved keyword which indicates null data. A search expression can be entered to find items with a data value. $NULL can be used for any data item; codes, names and strings can also use " " for the null value.

For example, a search may be executed to find the BORROW_DATEs that have no value by typing:

```
BORROW_DATE = $NULL
```

Arithmetic Expressions

Arithmetic expressions can also be used in a search expression, for either side of the relational operator, if the data item on which you are searching is numeric.The following examples are valid search expressions:

```
PRICE + 5.00 > 20.00
EMPLOYEE_NO / 2 FROM 1200 TO 3090
CALL_NUMBER BETWEEN 100000 AND CALL_NUMBER * 3
```

Recalling Search Expressions

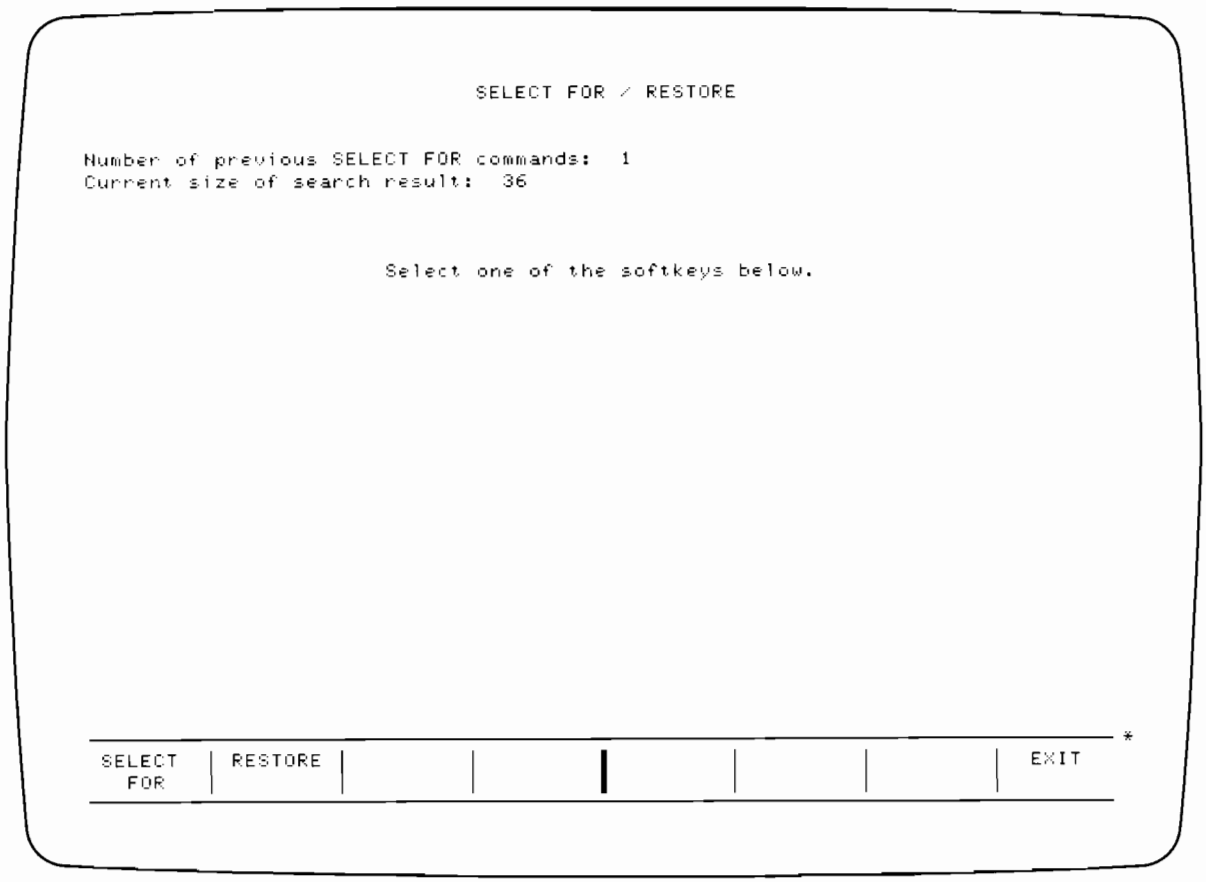
There is a softkey, RECALL, which is displayed as you are specifing the search expression. This softkey is used to re-display search expressions that have been previously entered allowing you to view, modify and reuse previous search expressions. Search expressions are stored into a 321-character buffer on a last in, first out basis. Each time the RECALL softkey is pressed, a previous search expression is displayed in the search expression area on the CRT.

When the recall buffer becomes full, each new search expression causes one or more of the oldest search expressions, depending on size, to be lost.

Narrowing Down the Search Result

After you have performed a search and established a search result, you can press the SELECT/RESTORE softkey to narrow down your current search result by selecting the entries that meet a new search criterion. The new search result is a subset of and replaces the previous search result. If no entries qualify, the previous search result is kept.

After you have narrowed down the original search result once, you can narrow down the new search result further or restore the original search result. After pressing the SELECT/RESTORE softkey the second time, this screen is displayed.



The SELECT FOR softkey is pressed if you want to narrow down the search result further and the RESTORE softkey performs the converse operation of the SELECT FOR softkey. Every time you press the RESTORE softkey, the previous search result is restored as the current search result.

Hints for Reducing Search Times

The search expression specified for a search affects how long the search takes. Most searches are performed by starting at the first entry in the set and reading each entry successively, determining whether the entry meets the specified criteria (serial searching). Here are a few hints to use when searching so that the searches are performed faster. They apply to single-set searches only.

- Searches are faster if a key item is used as the search item with "=" or "IS" as the operator. This enables QUERY/9000 to access the desired entry directly by way of calculated access. (Calculated access is explained in the IMAGE/DBM Programming Techniques manual.) For example, if you have ten thousand entries in a detail set and you are searching for entries with SUBJECT BEGINNING WITH DATA, QUERY/9000 has to read each entry (ten thousand) to check if it meets the search criterion. But, by using the search expression SUBJECT = DATA PROCESSING, QUERY/9000 accesses the occurrence of the subject in the master data set, then accesses all the detail entries directly because they are automatically chained together. This may mean reading only twenty entries. Explanations of chained access are found in the IMAGE/DBM Programming Techniques manual.

Note

When using LIBRARY to do the example searches, the time difference is not noticeable, but when you are searching a large data base and using large compound search expressions, the time difference may be significant.

- Here is another form of a compound search expression that enhances the speed of the search.

  PLANT = DTD AND (BORROW_DATE IS 1 FEB 1979 OR COPY_NUMBER >2)

- If you are using parentheses in the search expression, place the key item and the equals operator outside the parentheses (outer level).

Considerations

Search expressions can be constructed in many, many ways. Remember to compare data items of the same type, otherwise an error message is printed.

Multiple-Set Searches

When the criteria for retrieving entries is based on the information found in two or more sets, a **thread** must be specified in order to link these sets together through the key items. Specifying a thread enables QUERY/9000 to perform multiple-set searches.

Example

Suppose you know the call number of a book (7725348) and you want to know the names of the employees who have checked out that book. Borrower names are found in the BORROWER data set, but the call number is not. How do you find the necessary information?

Two separate searches need to be performed to find the employee number, then the borrower's name.

1. Search the INVENTORY data set for CALL_NUMBER = 7725348 and from the entry found, you know the EMPLOYEE_NO = 2843.
2. Search the BORROWER data set for EMPLOYEE_NO = 2843 to find the borrower's name.

The common data item is EMPLOYEE_NO; the information needed about EMPLOYEE_NO was found in the first search and used in the second search.

What if the search is more complex? Given a specific subject, (COMPUTER PROGRAMMING) how do you find out who has checked out books with this subject? The search expressions needed are:

1. Search BOOK data set to find SUBJECT = COMPUTER PROGRAMMING to get a list of the CALL_NUMBERs.
2. Search INVENTORY for CALL_NUMBER = ...;...;..., listing all the EMPLOYEE_NOs.
3. Search BORROWER for EMPLOYEE_NO = ...;...;... to find all the BORROWER_NAMEs.

If several CALL_NUMBERs or EMPLOYEE_NOs need to be listed for the searches, then many searches need to be made to find all the borrower names.

QUERY/9000 provides a feature to perform these multiple-set searches automatically in one pass using **threads**.

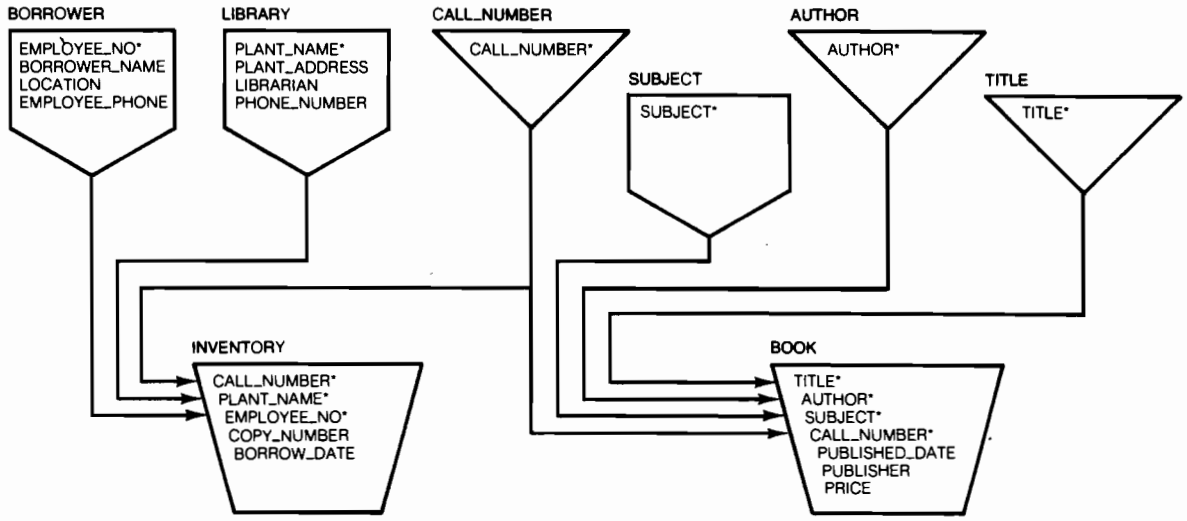
Threads

A **thread** is a named combination of data sets, specifying a unique path from one data set to the next, based on search items. A master set can only be linked to a detail set and a detail set can only be linked to a master set. QUERY/9000 enables you to name, define and store up to four threads with a maximum number of ten data sets linked in each thread.

In the previous example, a thread would need to be defined linking the BOOK data set to the BORROWER data set using common data items, CALL_NUMBER and EMPLOYEE_NO. Here is what the thread would look like:

LOAN = BOOK to CALL_NUMBER to INVENTORY to BORROWER

Here is a diagram of the thread defined:



A multiple-set search could then be performed to solve the above problem by using this search command:

FIND LOAN FOR SUBJECT = COMPUTER PROGRAMMING

Because you used a thread to perform the multiple-set search, you can list the data items TITLE and BORROWER_NAME using the list command.

Using Threads to Shorten the Searching Time

The normal method for performing searches is to do serial reads in a data set to find the entries that meet the specified criteria. The only exception to this is if an equality operator is used with a key item in the search expression like: SUBJECT = PHYSICS.

Suppose you want to find all the entries in the BOOK data set that have "SUBJECT WITH COMPUTER". This is not optimized because "WITH" is a relational operator. If there are 10,000 entries in the BOOK data set, a serial search can be time consuming.

However, the following thread can be defined to optimize the search time:

QUICK = SUBJECT to BOOK

Serial reads are now performed in the SUBJECT data set which has 100 entries. When a desired entry is found, the BOOK entry is accessed by way of a path. Since SUBJECT is much smaller than BOOK, the search is performed quicker.

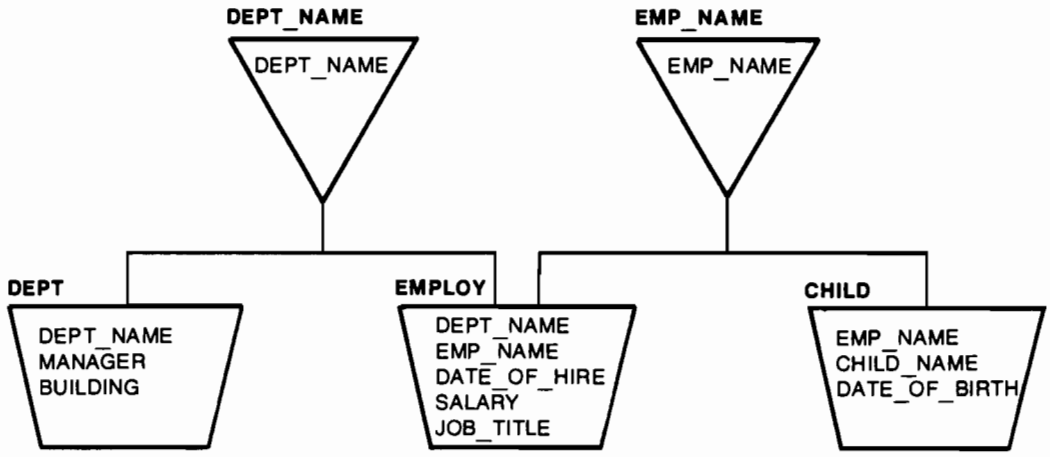
Optimizing Threads

When you are defining threads, the search is performed faster if:

- the thread starts from the smallest set and works toward the largest set.
- the sets involved in the search expression are defined towards the front of the thread. This means that QUERY/9000 does not have to read as many item values before making a true or false decision about the search criterion.
- the thread doesn't end with an automatic master data set, since no new data is stored in an automatic master.

Searching a Hierarchy

When defining a thread, it should contain only those sets necessary to find the desired information. For example, suppose you have a hierarchical data base that contains information about departments, employees and the employee's children; its diagram looks like:



If you wanted to find all the children of those employees who work in BUILDING #1, this thread is defined:

HIERARCHY = DEPT to DEPT_NAME to EMPLOY to EMP_NAME to CHILD

After a search is performed using this thread, an employee with no children does not appear in the search result. This illustrates that an entry is put in the search result only if the thread can be completely traversed from one end to the other. For example, if you wanted to find all employees working for the MANAGER Jack Smith, the HIERARCHY thread should not be used. Using the HIERARCHY thread, any employees without children who work for Jack Smith are not included in the search result. The following thread could be defined to find those employees that work for Jack Smith:

EMPLOYEES = DEPT to DEPT_NAME to EMPLOY

Sorting Data Entries

After a search has been performed, the SORT RESULT softkey may be used to sort the entries in the search result into a specific order. Sorting can be done using a single key by which to sort the entries; this is called a **simple sort**. For example, after a search to find all the books published after 1 JAN 1970, you can sort these books by author in ascending (alphabetical) or descending order. If the order is not specified, the entries are arranged in ascending order.

Sorting Using More Than One Key

Sorting the search result using many keys is called a **multiple-key sort**. With a multiple-key sort, the search result is sorted by the first key. Entries having the same value for the first key are then sorted by the second key. This process may continue for a total of **ten** keys. Ascending or descending order is specified for each of the keys.

For example, suppose a search found all the books which were borrowed after 1 JAN 1980. These books could be sorted by employee number and call number. All the employee numbers would be in ascending order, but if one employee checked out more than one book after 1 JAN 1980, the book with the lowest call number would be listed first.

The following sorts show the difference between single and multiple-key sorts. The single key sort was sorted by EMPLOYEE_NO. The multiple-key sort was sorted by EMPLOYEE_NO and CALL_NUMBER.

USING ONE SORT KEY

EMPLOYEE_NO	CALL_NUMBER	COPY_NUMBER
1140	749772	3
2359	70175572	1
2568	77159286	3
2732	777211	2
3143	7116962	1
3780	747127	2
3887	6828110	1
4046	7724414	2
4105	7519319	3
4105	6321473	2
4191	78314357	2
4191	738242	1
5901	728034	1
6942	7823624	2
7334	7922327	3
8107	7773953	1
8190	7422058	1
8190	71122679	1
8283	749772	2
8751	7528025	3
9122	748541	5
9790	7423929	5

duplicate employee number

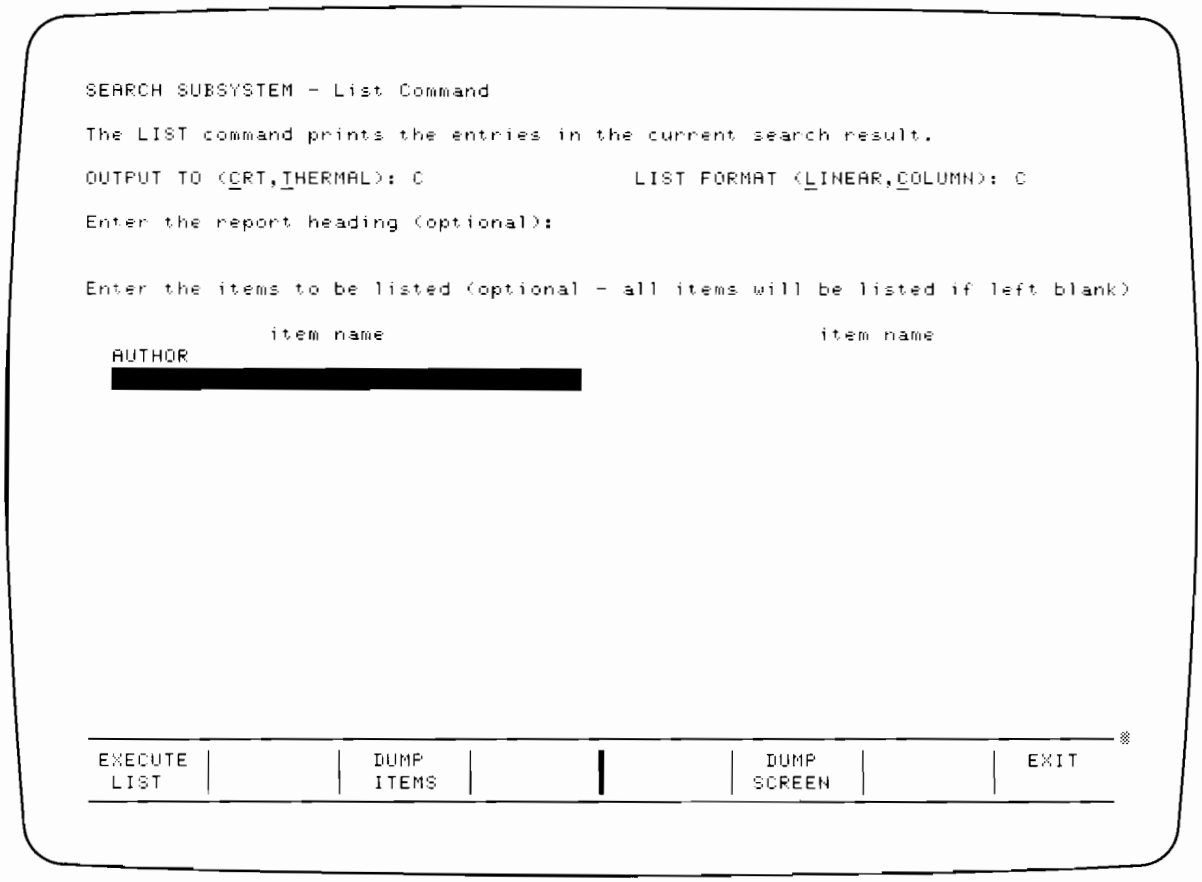
USING MORE THAN ONE SORT KEY

EMPLOYEE_NO	CALL_NUMBER	COPY_NUMBER
1140	749772	3
2359	70175572	1
2568	77159286	3
2732	777211	2
3143	7116962	1
3780	747127	2
3887	6828110	1
4046	7724414	2
4105	6321473	2
4105	7519319	3
4191	738242	1
4191	78314357	2
5901	728034	1
6942	7823624	2
7334	7922327	3
8107	7773953	1
8190	7422058	1
8190	71122679	1
8283	749772	2
8751	7528025	3
9122	748541	5
9790	7423929	5

call numbers change orders

Listing the Data Entries

After a search is performed, use the LIST softkey to print the values of all or selected items from the entries in the search result. Each item value is preceded by the item name and set name printed as SET.ITEM. The printout can be in either linear or columnar format. Linear format prints one item per line; any value longer than one line is printed on the next line. Columnar format prints several items in a simple table. Here is a diagram of the list screen when using columnar format.



Considerations for Columnar Format

When you specify columnar format, you must make sure that the sum of the maximum lengths of the items you want to list doesn't exceed the maximum line width. As you enter each item, this sum is indicated by the current line length. If the maximum length is exceeded, an error message is printed.

The maximum line length of the CRT or internal printer is 80 characters. If you are using an external printer, the maximum line length was indicated during the System Configuration section in the beginning of the QUERY/9000 program.

Specifying the Set

When item values are listed, the data item name is also listed, preceded by the set name and a period (set.item). When you specify the items to be listed, you can precede any item name with a set name and a period. If no set is specified, the set closest to the end of the thread that contains the item is used.

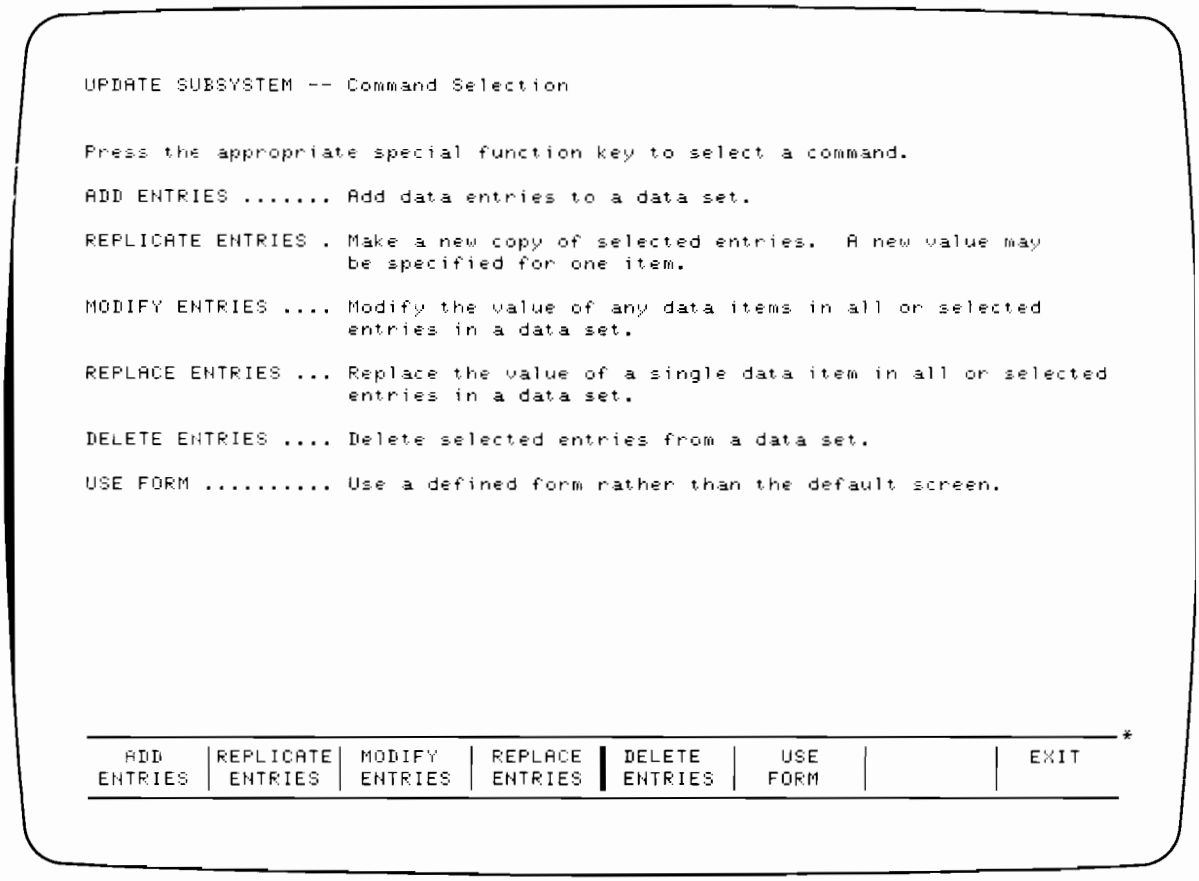
Chapter 6
Changing the Data in the Data Base

The UPDATE subsystem assists you in modifying the data stored in your data base. There are four operations that can be performed on a data set. You can:

- add a data entry
- delete a data entry
- modify a data entry (change one or more item values in an entry)
- replace an item (change the value of a single data item in one or more entries)
- replicate an item (duplicate one or more entries, changing the value of a single item)

Accessing the UPDATE Subsystem

To access the UPDATE subsystem, open the data base as explained in Chapter 2. Then press the UPDATE softkey. The screen now looks like:



After a softkey is pressed (ADD ENTRIES, DELETE ENTRIES, REPLACE ITEM, MODIFY ENTRIES), you are asked to select a data set to update from the data sets displayed on the screen. If any data sets are filled to their capacity (when adding entries) or empty (when deleting entries), they are listed next. The last line on the screen displays a message similar to:

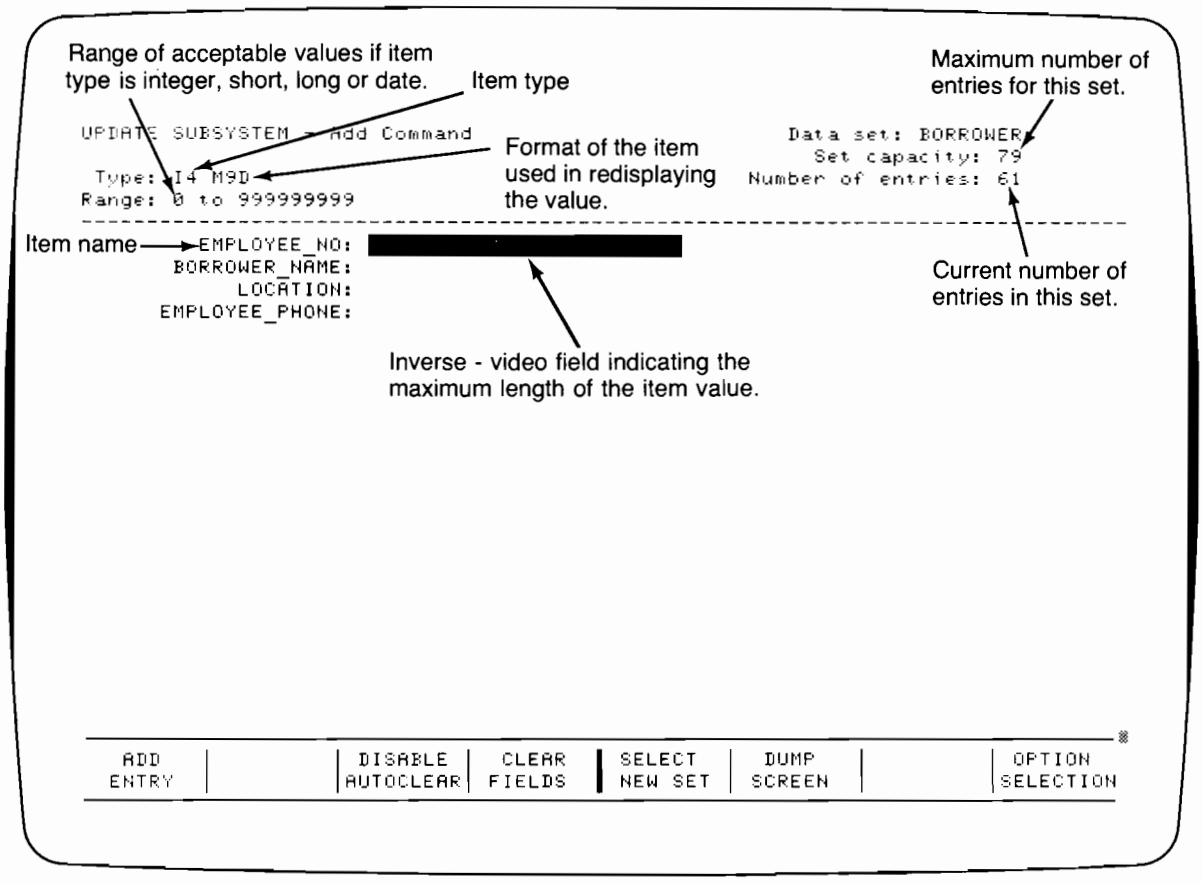
"The remaining 4 data sets are automatic masters or are password protected"

This number refers to all automatic master sets (since these are not directly updated) and all data sets to which you do not have write access with the password used to open the data base.

Adding Data Entries

To add entries to a manual master or detail data set press the ADD ENTRIES softkey. After a set is specified, the screen displays the data item names on the left side of the CRT in the same order as the data base definition.

As you access each item, the following information is displayed:



If the item is a code type, you may press the DUMP CODES softkey for a listing of the acceptable input values.

After a value is typed, press the RETURN, ↓, or TAB key. QUERY/9000 checks the value for correct type, range, and code values. If the value is valid, the cursor moves to the next item. To move the cursor to a previously filled field, hold down the SHIFT key, and press TAB, or press the ↑ key. Press the ADD ENTRY softkey to store the data entry into the data set.

Autoclear

There is a softkey defined at the beginning of the UPDATE subsystem that gives you the option of having AUTOCLEAR ENABLE or AUTOCLEAR DISABLE while adding entries. If AUTOCLEAR is enabled, all fields are cleared after you store an entry. If AUTOCLEAR is disabled, all fields are not cleared after you store the entry. This is useful when only one or two data item values change between entries.

Considerations

In general, always try to fill the manual master data sets first. When adding an entry to a detail data set, make sure that the values of key items are already stored in the related manual master(s). If you try to add an entry when the key value doesn't exist in the manual master, an error message is displayed. In this case, you can either add the key item value to the manual master without losing the detail data item values (press the ADD TO MANUAL softkey), or not add the entry into the detail data set (press the CLEAR FIELDS, SELECT NEW SET, or OPTION SELECTION softkey). Another choice is to change the field value for the item which is missing from the entry in the master data set (press the CHANGE KEY ITEM softkey).

Using a Form for Adding Entries

If you have defined a form (forms are covered in Chapter 7) for the data set, QUERY/9000 asks if you want to use it for data input. If you choose to use it, the screen is cleared and the form is displayed. The information that is normally displayed at the top of the screen when using the default screen image is not displayed while using a pre-defined form.

Deleting Data Entries

By pressing the DELETE ENTRIES softkey at the beginning of the UPDATE subsystem, you can search for and delete entries from a selected data set. You can only delete complete entries in a set, not individual data items.

A search expression is used to select the entries to be deleted. When the search is completed, the number of entries meeting the search criteria is displayed. Pressing the PAUSE DELETE softkey enables you to examine each of these entries. If the complete entry doesn't fit on the screen, the DOWN ARROW (↓) and UP ARROW (↑) keys enable you to scroll the entry. If you want to delete the entry displayed on the screen, press the DELETE ENTRY softkey; otherwise press the NEXT ENTRY softkey. If all the entries found by the search are to be deleted, press the DELETE ALL softkey.

Consideration

When deleting entries in a manual master data set, you need to ensure that the key item value in a particular entry is not found in a related detail set entry. If you try to delete a master entry that has detail entries linked to it, a warning is displayed on the CRT. You must delete or change the entries in the detail data sets before the manual master entry can be deleted.

Modifying Data Entries

By pressing the MODIFY ENTRIES softkey, you can change the values of data items in one or more data entries. For example, MODIFY might be used if a particular employee moves to a different department. A search expression is entered to select the entries to be modified. When the search is completed, each entry can be modified.

After a set is specified and a search is performed, the screen displays each entry in the same format as in the ADD command. If the current data item is a code type, you may press the DUMP CODES softkey for a listing of the acceptable values.

You now may edit the entry being displayed. Whether or not a data item value is modified, press RETURN to enter the value and move the cursor to the next item. This causes the value to be checked for correct type, range and code values.

If you don't want to modify the entry displayed, press the NEXT ENTRY softkey to display the next entry in the search result.

When you have changed all the desired items in a particular data entry, press the MODIFY ENTRY softkey. This replaces the old entry in the data base with the new entry, and then displays the next entry from the search result for editing.

Using a Form for Modifying Entries

If you have defined a form for the data set being modified, QUERY/9000 asks if you want to use it for data modification. If you elect to use the form, the form for the specified data set is displayed on the screen. The information that is displayed at the top of the screen if you are not using a form is not displayed while using a pre-defined form.

```
                    *** THE BOOK ENTRY FORM ***

Title: THE MYTHICAL MAN-MONTH              Author: BROOK,FREDERICK

Subject: ******************                          Price: 15.30

Publisher: ADDISON-WESLEY

Date Published: 22 Aug 1975                Call number: 744714
```

Using a form for modification has one limitation. If the current value for a data item is longer than the field defined on the form, the field is filled with asterisks (*) and you cannot move the cursor into the field to modify it. For example, when defining a form for the BOOK set, you may have thought that all the values for SUBJECT were less than 15 characters long. In one entry, however, the SUBJECT has 17 characters. When modifying this particular entry, the form would be displayed as shown next and you couldn't modify SUBJECT for this entry.

Considerations

When modifying an entry in a detail data set, make sure that the values of key items are already stored in the related manual master(s). If you try to modify an entry when the key value doesn't exist, the choices QUERY/9000 allows you to make are the same as in ADD.

Replacing Data Items

Press the REPLACE ITEM softkey when you want to change one particular value for a data item which is found in all or selected entries of a data set. For example, if the borrow date were entered as Sunday's date instead of Monday's date, you could correct all of the occurrences of the date by using the REPLACE ITEM softkey. You can replace any item value in a detail data set including the key item value. When replacing the key item value in a manual master data set, the key item can only be modified if one entry is found in the search result.

When replacing an item, you need to specify the data set containing the data item. You are then asked to replace the data item. If it is an array item, the particular subitem must be specified, such as ADDRESS(2).

After you have entered the data item name, enter its new value. If the item is not required, the new value can be left blank to store a null value for the item.

A search expression is then used to select the data entries in which the data item value is to be replaced. After the search is performed, press the REPLACE ALL softkey to replace the item value in all the entries found by the search, or press the PAUSE REPLACE softkey to see each entry. If you want to replace the entry displayed on the screen, press the REPLACE ENTRY softkey; otherwise press the NEXT ENTRY softkey.

Example

Replacing an item in the LIBRARY data base would be needed if DCD changed their telephone number from (303)226-3800 to (303)524-5683. Thus, all the entries in the BORROWER set with the old telephone number (303)226-3800 would need to be changed to the new telephone number (303)524-5683. You would press the REPLACE ITEM softkey and specify the BORROWER data set by typing the set number 2. For the name of the item to be replaced, type EMPLOYEE_PHONE; for the new value type (303)524-5683. When prompted for the search expression, type EMPLOYEE_PHONE = (303)226-3800. When the search is complete, the screen displays the number of entries found. You really don't need to press the PAUSE REPLACE softkey because you know that all phone numbers with (303)226-3800 need to be changed. Press the REPLACE ALL softkey and the replace is done for all the entries.

Replicating Data Entries

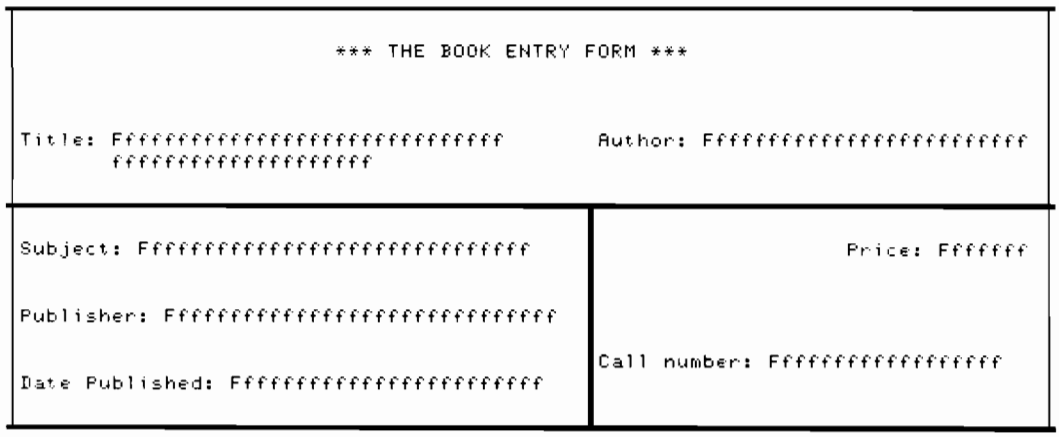
Replicating entries is a combination of duplicating and replacing. It is similar to the REPLACE command, except that the entries in the search result are duplicated and entered as new entries in the data set. The specified field is filled with the value you specify in the same manner as the REPLACE command.

Chapter 7
Using Forms with Your Data Base

Introduction

The FORMS subsystem enables you to define a form for each manual master or detail data set. The **form** is an image displayed on the CRT and can include heading, text, lines for dividing the form into sections, a field for entering each item value and additional information, such as a more complete description of a data item name. It can be used while adding or modifying data entries during the UPDATE subsystem. The advantage of using a form is that additional information can be displayed when entering data that is not available if a form is not used.

The following form is used for the examples in this chapter.



Forms can include the following features:

- Video highlighting for text and input fields (such as blinking)
- Line drawing characters to partition the form
- Definition of fields for data item values
- Definition of tab order for entering the data items

The FORMS subsystem is run in two stages. In the first stage you create the form. The second stage occurs after the form image has been created and consists of defining the tab order and storing the form.

Limitations

Each form has a maximum of 66 lines. One field must be defined for each item and subitem in the data set. This field cannot be longer than the maximum length of the item defined during the definition of the data base, but it can be shorter. For example, you might want to make the field three characters shorter for the item called PHONE_NO by leaving out spaces for the area code.

Creating and Modifying a Form

Pressing the DEFINE FORM softkey accesses the FORMS subsystem. This softkey is accessed by pressing the MORE KEYS softkey in the main QUERY subsystem. You then specify the data set for which you want to create a form. The only sets that you can create forms for are manual masters and detail data sets to which you have read/write access. The CRT is cleared and when the cursor appears at the upper left corner of the CRT, you are ready to begin creating the form.

A form is defined by drawing lines to divide the form into sections, entering text to name a data item field (or to provide additional information) and defining a field for each data item. Additionally, any of the text can be highlighted with video highlights such as inverse video. Text and line segments are entered at the current cursor position.

If a form exists for the specified data set, you can RECALL the form to modify it, DELETE the form so that it no longer is defined for that data set, define a NEW FORM which replaces the present form for that data set, or select a new data set.

If you RECALL the form and it has more than 20 lines, the bottom line of the CRT displays the loading of the extra lines of the form. When the form loading is complete, the first twenty lines are displayed on the CRT. When the cursor appears at the upper left corner of the CRT, you are ready to begin editing the form. The keys have the same definintion as when creating a form, as explained in the next section.

Key Definitions

All alphanumeric keys and the following keys retain their normal definitions while defining a form.

BACK SPACE	INS LN	TAB
DEL CHAR	REPEAT	TAB CLEAR
DEL LN	SHIFT	TAB SET
INS CHR	TYPWTR	

The following keys have been redefined so that pressing:

HOME	Moves the cursor to the left-most position of the current line.
SHIFT HOME	Moves the cursor to the top left of the screen.
STORE	Moves the cursor to the left-most position of the next line of the form.
CLEAR LINE	Clears the screen image from the cursor to the end of the line or to the next field, whichever occurs first.
CLEAR LINE	Clears the current line of the form, excluding fields.
SHIFT TAB	Moves the cursor backward to the previous tab position.
RECALL	Moves the cursor to the top left of the form. The top left of the form may not be visible on the screen if the form is longer than 20 lines.
SHIFT RECALL	Moves the cursor to the bottom left of the form. The bottom left of the form may not be visible on the screen if the form is longer than 20 lines.

Moving the Cursor

The ARROW keys are used to move the cursor on the screen in the direction specified by the arrow. The ROLL UP key moves the form up ten lines. The cursor remains in its relative screen position (the form moves with respect to the cursor). The ROLL DOWN key moves the form down ten lines. The cursor remains in its screen relative position.

Video Highlights

Video highlights can be used to accent important items in the form. The video highlights are accessed as they normally are on the HP 9000, by holding down CONTROL and pressing the appropriate SFKs. The highlight remains in effect until you press CTRL and the SFK or SFKs which you selected to turn it on.

CONTROL-SFK 0	Inverse Video
CONTROL-SFK 1	Blinking
CONTROL-SFK 2	Underline

Drawing Lines

After pressing the LINE DRAW CHARS softkey, the softkeys are defined with the line drawing characters. Pressing any of the first six softkeys draws that line segment at the present cursor location. Pressing MORE KEYS enables you to access the other sets of line drawing characters. Pressing EXIT returns you from the line drawing keys to the previous level of FORMS.

Example

Specify the BOOK data set when QUERY/9000 asks you for a data set. Use the LINE DRAW CHARS sets to draw the border around the example form, as shown here.

```
*** THE BOOK ENTRY FORM ***

Title:                                  Author:

Subject:                                        Price:

Publisher:

                                        Call number:
Date Published:
```

Defining Fields

Each item and subitem in the set must have a field to contain the value when data is entered. A field can be defined at any point in the creation of the form. Defining a field consists of specifying the starting position, the corresponding data item, the length and how long to make each line of the field. For example, a 200 character field could be defined as 4 lines of 50 characters each.

To define a field, position the cursor where you would like the field to begin and press the START A FIELD softkey. QUERY/9000 then asks you for the item number (which is based on the order of the items defined in the data set). The ITEM LISTING softkey lists the item names, their corresponding numbers, and maximum lengths. You need to enter the number of characters (for the particular field) displayed on a form-line, then the maximum number of characters for each field. The maximum length of the field is the maximum length specified when the item was defined.

The first character in a field is identified by an uppercase F. All subsequent characters are identified by lowercase f's. The order in which fields are defined is not important; order specification occurs after all fields are defined.

Note

No area in a form can be shared by two or more fields.

Example

After all fields in the example form have been defined, the form appears as:

```
                    *** THE BOOK ENTRY FORM ***

Title: Fffffffffffffffffffffffffffffff      Author: Ffffffffffffffffffffffff
       fffffffffffffffffff

Subject: Fffffffffffffffffffffffffffff                    Price: Ffffff

Publisher: Ffffffffffffffffffffffffffffff

                                          Call number: Fffffffffffffffff
Date Published: Ffffffffffffffffffffff
```

Deleting a Field

Deleting a field is accomplished by positioning the cursor anywhere within the field and pressing the DELETE A FIELD softkey.

Moving a Field

To move a field, position the cursor at the beginning of the field and press the MOVE A FIELD softkey. Move the cursor to the new location on the form and press the PLACE FIELD softkey. The field is placed in the new location unless it overlaps another field or it is at the end of the form. In this case, an error message is displayed and you must move the field to another location on the form. The CANCEL MOVE softkey puts the field back to its original position before the MOVE A FIELD softkey was pressed.

Processing the Form

After you have finished defining the lines, text and fields on the form, press the PROCESS FIELDS softkey which appears after you have defined a field for every data item. The FORMS subsystem then enters the portion of the program where you can define tab order.

Defining the Tab Order

Tab order is the sequence in which the cursor is moved from field to field when entering data using a form. There is a default order which corresponds to the order in which items appear in the definition of the set.

Pressing ORDER FORM enables you to define a different tab order. The form now has an arrow symbol (↑) pointing to the first character of the first field on the form. At this time you can change the tab order by typing in a number which specifies when, in the tab order, this field should be accessed; then press CONT. If you do not want to change the tab number of a field, press CONT.

The process of specifying tab order continues until each field has the desired values. Press PROCESS TABS to leave the tab definition mode.

Considerations

You can use either real numbers or integers for tab order. However, when the tab numbers are processed, they are renumbered with integers. When two fields have the same tab number, they are renumbered using the default order. For example, if two fields have a value of 4, the field with the lowest item number would be numbered 4, the other with 5.

Printing the Form

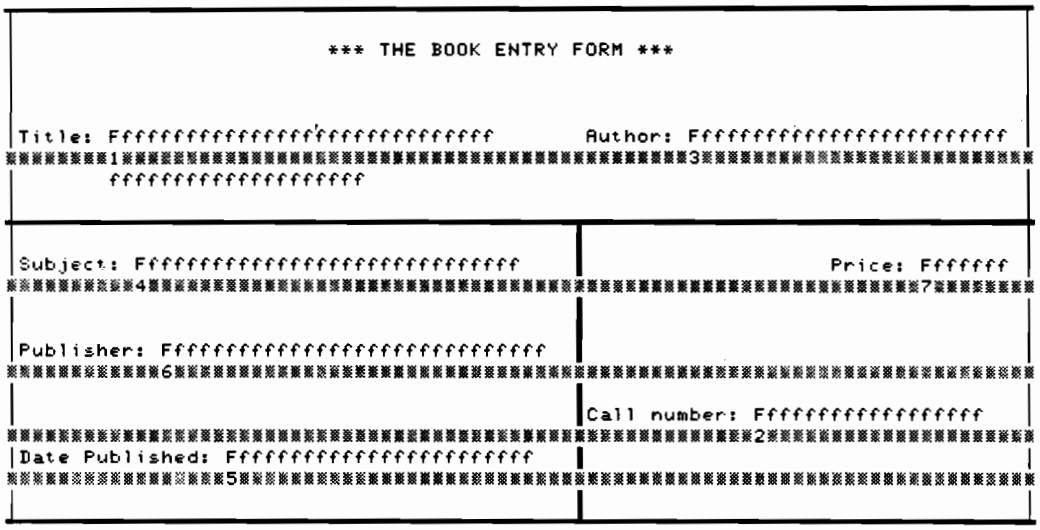
To see what the form looks like, press the PRINT REPORT softkey and three copies of the form are printed on the internal printer. The first copy shows you what the complete form looks like. The second copy shows you the item numbers as they are currently defined on the form. The third copy of the form shows you the tab order as it is currently defined for the form. The printout produced for the example form look like this.

```
                        *** THE BOOK ENTRY FORM ***


Title: Fffffffffffffffffffffffffffff        Author: Fffffffffffffffffffffff
       ffffffffffffffffff

Subject: Fffffffffffffffffffffffffffff                    Price: Ffffff

Publisher: Fffffffffffffffffffffffffffff

                                                  Call number: Fffffffffffffffff
Date Published: Fffffffffffffffffffff
```

Item Numbers

```
                        *** THE BOOK ENTRY FORM ***


Title: Fffffffffffffffffffffffffffff        Author: Fffffffffffffffffffffff
       ffffffffffffffffff

Subject: Fffffffffffffffffffffffffffff                    Price: Ffffff

Publisher: Fffffffffffffffffffffffffffff

                                                  Call number: Fffffffffffffffff
Date Published: Fffffffffffffffffffff
```

Tab Numbers



The item and tab numbers are read similarly; a line of null characters is printed under each line of the form which contains a field. At the position under the first character of the field is a number which represents its item number or the tab order. When there are several character fields that are adjacent to each other, item and tab numbers are specified by multiple lines of null characters under the fields. The numbers are staggered so that the initial number of each field is under the first character of its field as shown here.

```
FFFFFF
1▒▒4▒▒
▒2▒▒5▒
▒▒3▒▒6
```

Data Items for Data Set: BOOK

Item No.	Item Name	Maximum Length	Item No.	Item Name	Maximum Length
****	TITLE	50	****	PUBLISHED_DATE	24
****	CALL_NUMBER	24	****	PUBLISHER	30
****	AUTHOR	25	****	PRICE	24
****	SUBJECT	30			

If two fields are defined next to each other, each field contains 1 character, and their item numbers are 20 and 21, the numbers will be displayed as:

```
FF
22
01
```

Storing the Form

The final step in the creation of a form is to store it on a mass storage medium. The form is stored by pressing the STORE FORM softkey. You are asked for the volume name of the disc on which the form is to be stored. If you are modifying a form and you are going to store the form on the same volume as the original form, the original form is purged before the modified form is stored.

Press the EXIT softkey to return to the following softkey selections:

Chapter 8
Defining a Data Base

Introduction

The DEFINE subsystem is used to interactively define or redefine a data base to QUERY/9000. The final output of the DEFINE subsystem is a data base **root file** (containing all the structural information about the data base), and optionally, the data set files into which the data is stored.

This chapter has three main sections:

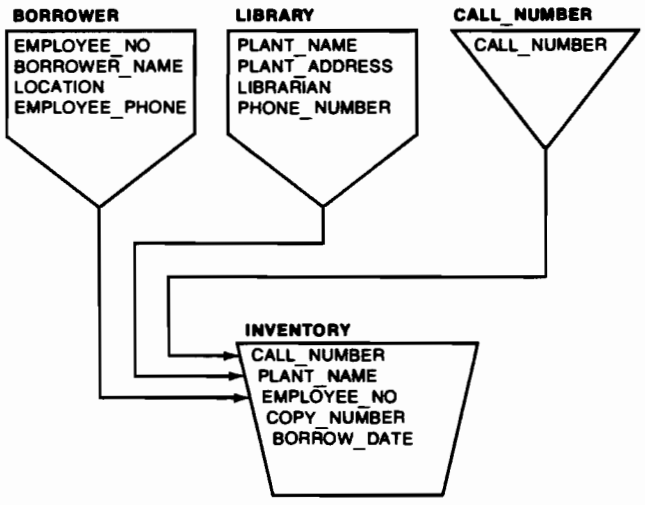
- Planning before you define a data base (definitions and terms)
- Using QUERY/9000 to define a data base
- A step-by-step example

Once you have designed a data base, the softkey definition section gives you the basic information needed to define a data base. Other information is provided on the screen. You may want to turn to the definitions or the example first to obtain added information before beginning.

Planning Before You Define a Data Base

The Data Base Diagram

The first step in defining a data base is to use the Data Base Design Kit to take you from a situation where a data base is needed to a diagram similar to this:



Item and Set Information

What does the previous diagram show? There are four different sets. A name is associated with each data set (LIBRARY, CALL_NUMBER, BORROWER and INVENTORY). Their shapes indicate that two sets are manual master data sets (LIBRARY and BORROWER), one is an automatic master data set (CALL_NUMBER),and one is a detail data set (INVENTORY). Additionally, the diagram contains all the item names. The key items are items on either end of the paths which link the sets together. Here is a list of all the names with the key and search items flagged.

Data Set Name	Item Name	Notes
LIBRARY	PLANT_NAME	Key item
	PLANT_ADDRESS	
	LIBRARIAN	
	PHONE_NUMBER	
CALL_NUMBER	CALL_NUMBER	Key item
BORROWER	EMPLOYEE_NUMBER	Key item
	BORROWER_NAME	
	LOCATION	
	EMPLOYEE_PHONE	
INVENTORY	CALL_NUMBER	Search item
	COPY_NUMBER	
	PLANT	Search item
	EMPLOYEE_NO	Search item
	BORROWER_DATE	

Beyond the Diagram

After the diagram is complete, the following things must be determined. These items are explained in detail in the following sections. This information can also be found on the screen during the appropriate part of DEFINE.

data base name — 1 to 16 alphanumeric characters; the first must be a letter. This name is used to identify the data base.

volume label — 1 to 16 ASCII character string excluding blanks, semi-colons, colons, commas, "/" and "<". This label must be printed* on the disc before you start running the DEFINE subsystem. A data base may reside on more than one disc and therefore have many labels. You should label each disc with a different name. You specify which data set or sets are to be stored on which disc when defining the data sets.

* Use LABEL expression in INITIALIZE statement or use PRINT LABEL statement.

passwords

1 to 16 characters in length including upper and lowercase letters, digits 0 thru 9 and the underscore character.

To access the data base, a valid password must be be given. Thus, you need only tell passwords to those people you want accessing the data base.

If no passwords are defined, unrestricted access is given for every data set in the data base. Any particular password can be defined to give access to any of the data sets. Each password can have READ-ONLY access, READ/WRITE access, or NO access to a particular data set.

The different types of access are specified with R for READ-ONLY, W for READ/WRITE, and nothing is typed for NO access.

If passwords are defined, the first password must be defined. It is a global password with write access to all sets. Up to 13 passwords can be defined per data base.

data set name

1 to 16 characters, including letters, digits 0 thru 9, and the underscore (_) character. The first character must be a letter. Each set name must be unique in the data base. Up to 63 data sets can be defined per data base.

data set type

M specifies MANUAL
A specifies AUTOMATIC
D specifies DETAIL

data set capacity

Integer from 1 to $2^{31} - 1$ or 2 147 483 647. For master data sets, choose a prime number and a capacity which is 25% greater than the number of entries you intend to have. This helps ensure rapid searches.

data item name

1 to 16 characters, including letters, digits 0 thru 9, and the underscore (_) character. The first character must be a letter. Each item name must be unique in the data set and in the data base.

A maximum of one data item can be defined for an AUTOMATIC data set, 1 023 items for MANUAL and DETAIL data sets, and 1 023 items for a data base.

pseudonyms

1 to 16 characters, including upper and lowercase letters, digits 0 thru 9, and the underscore (_) character. The first character must be a letter. Each pseudonym must be unique and cannot match an existing data item name.

Up to four pseudonyms can be defined for any data item. These serve as alternate names for an item. A maximum of 255 pseudonyms may be defined for a data base.

DATA ITEM TYPES:

character string	X represents CHARACTER string. Any number of characters from 1 to $2^{31}-1$ or 2 147 483 647. Note that a practical limit is imposed by the mass storage medium capacity on which the data base is to be located.
I2	A two byte integer (BASIC type integer) is denoted by I2. Any whole number between −32 768 and +32 767 may be represented.
I4	BASIC type DOUBLE which is a four byte integer is denoted by I4. Whole values between −2 147 483 647 and +2 147 483 647 can be represented in this numeric type.
R4	Floating point values between −3.402823E+38 and +3.402823E+38 can be represented to roughly seven significant figures using the BASIC type SHORT. IMAGE and QUERY use R4 to denote this type.
R8	Long real floating point values may be stored in type R8. The range of values span from −1.7976931348623 15E+308 to +1.797693134862315E+308 with roughly 16 significant digits.
C2	C2 represents a CODE. A code item value may be any string of 1 to 16 characters. Each code item may have up to 32 code values with a maximum 255 code values for the entire data base.
format (picture word)	An item may be extended using the item format. This specifies how to parse an input value or format an output value.
name	NAME format can be applied to X character types.
date, time, timedate	R8 items can be modified with this format. These items follow the same convention used by BASIC's timedate conversion routines.
dimensions	An item (not used as a key search or sort item) may be an array. The array may have from one to six dimensions. Each dimension must have a lower and upper bound. The total number of elements in the array must be less than 2 147 483 647. This limit of 2 147 483 647 must also be applied to the total number of bytes in an item. (In practice the capacity of the disc on which the data base resides, restricts this even more.)
maintenance word	1 to 16 ASCII character string. A maintenance word is used to protect the entire data base against unauthorized people purging, erasing, backing up, or modifying the data base. It is similar to a password, except passwords enable only certain people to update the data in a set; a maintenance word helps prevent purging or restructuring of the whole data base. If the field is left blank, anyone can purge or restructure the the data base.

Using QUERY/9000 to Define the Data Base

You are now ready to run the DEFINE subsystem of QUERY/9000. To load the DEFINE subsystem, press the DEFINE DATA BASE softkey at the beginning of QUERY/9000 when these softkeys are displayed:

When the DEFINE subsystem is loaded, press the DEFINE A DATA BASE softkey. Keep the diagram you drew handy to use as a reference when QUERY/9000 asks you for information.

The steps for defining a data base are:

1. Enter the data base information. Press PROCESS BASE INFO when the information is correct.
2. Press DEFINE DATA SET to define the first set.
3. Enter the information for the first data set. Press PROCESS SET INFO when the information is correct.
4. Press DEFINE DATA ITEM to define the first item.
5. Define the data items for the first data set.
6. Repeat steps 2 through 5 for each data set.

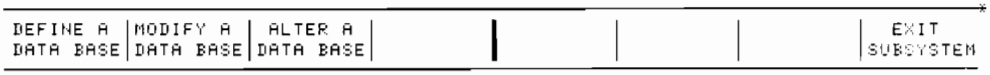
> **Note**
>
> It is recommended that you define master sets first, then detail sets.

7. Create the root file.
8. Create the data set files.

The softkeys for defining a data base are discussed in the following sections.

Data Base Options

When you begin defining a data base, these softkeys are displayed:



After a data base definition has been read into memory, this level of softkeys is expanded to:

If DEFINE, MODIFY, ALTER, or EXIT is pressed, the existing data base definition is cleared.

Defining a New Data Base

If you want to define a new data base, press the **DEFINE A DATA BASE** softkey. After this softkey is pressed, QUERY/9000 asks you for the information about the data base: the data base name, volume label, directory path, lexical order and passwords. Press PROCESS BASE INFO when the information is correct. You are then asked for the first set name.

Modifying an Existing Data Base Definition

The **MODIFY A DATA BASE** softkey enables you to modify an existing data base. You can either create a new data base from an existing data base definition or you can just change parts of the current definition.

After you specify which data base to restructure, DEFINE asks you to enter the data base name, directory path, root file volume and a maintenance word if one was used when the data base was created. A data base created with a maintenance word cannot be modified unless the same maintenance word is specified. If any error is found, an error message is displayed and the program returns to the beginning of DEFINE. If the data base is opened successfully, the data base information is displayed and you begin editing the definition.

Note

If the changes to the data base are too extensive, you may need to write an IMAGE program to transfer data from the old to the new data base.

Altering the Non-structural Parts of an Existing Data Base

QUERY/9000 allows you to change the non-structural definition of an existing data base. This includes the changing of the data base name, changing passwords (including adding passwords to a formerly public data base or making public a formerly protected one), set names, password access to sets, item names and item pseudonyms and item formats. Additionally, the capacity of sets may be expanded. (The extension of sets may be done from the keyboard or from an IMAGE/DBM program using the DBEXTEND statement. Refer to the BASIC Language Reference manual.)

Unlike MODIFY A DATA BASE, the contents of the data base remains intact over the ALTER. No UNLOAD or LOAD is necessary.

As with define you need to specify the data base name, root file volume, directory path, and maintenance word of the existing data base.

Note

While ALTERing a data base, no other user should be allowed to access the data base.

CONTINUE EDITING

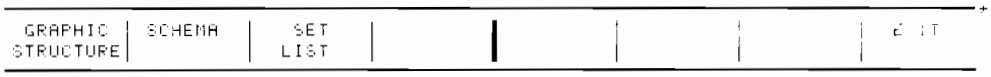
Press CONTINUE EDITING to retain the data base definition for further editing.

Creating the Data Base

When the data base definition is complete, press the **CREATE ROOT FILE** softkey. This procedure is discussed later in the chapter.

Show Data Base Information

When the **SHOW BASE INFO** softkey is pressed, the following softkeys are displayed:



Data Base Structure

The **GRAPHIC STRUCTURE** softkey is used to print the current structure of the data base using the internal printer. This diagram contains the defined data sets, the data items contained in each set and the paths defined between the sets.

Schema Listing

The **SCHEMA** softkey is used to print a word description of the data base using the internal printer. The schema listing contains the current data sets, data items, pseudonyms, ranges, code values and paths.

Set Info

The **SET INFORMATION** gives information on each set, including entry length and media record length, capacity, the number of sectors a set spans, and the volume on which it will appear.

Returning to QUERY

Press the **EXIT SUBSYSTEM** softkey to return to the beginning of QUERY/9000.

Data Set Options

When you begin defining a data set, these softkeys are displayed:

Only those softkeys which are active will be present. When defining a new data base, only DEFINE A DATA SET is present since there are no sets to change or delete. When doing an ALTER A DATA BASE, only the ALTER A DATA SET softkey is present because the others would cause structural changes to the data base, which is not allowed in that mode.

Define a New Data Set

Press the **DEFINE A DATA SET** softkey to define a new data set. You need to enter the set name, type, capacity, volume name, directory path and password access. Then press PROCESS SET INFO. You are then asked to define the items for that set.

Create a Data Set from an Existing Definition

The **MODIFY & RENAME** softkey defines a new data set from an existing data set definition. The original data set definition is not changed with the MODIFY & RENAME function. The set type may be changed. One exception to this is changing a set with more than one item to an automatic master set. If a set type is changed from a master to a detail or vice versa, the set number must also be changed. The new set contains the same data items although none of the data set paths are copied.

Changing the Definition of an Existing Data Set

If you want to change any part of an existing data set definition except the number or type of the set, the **ALTER & RESTORE** softkey is used. The altered data set definition is stored in place of the original set definition.

Deleting a Data Set Definition

The **DELETE DATA SET** softkey is used to remove an existing data set definition. The entire definition of the selected data set is deleted. After the data set is deleted, the remaining data sets are listed and you are asked to select another set to be deleted. Press the DATA SET OPTIONS softkey to cancel the delete option.

The DATA BASE OPTIONS softkey checks the data base definition for completeness before returning to the Data Base Options level. The test insures that:

- Every password has access to at least one set.
- There are no duplicate passwords.
- There are no duplicate item names or pseudonyms.
- There are no duplicate set names.
- Every set has at least one item.
- Key, search, and sort items are scalars not over 256 bytes in length.
- Automatic masters are linked to at least one detail set.

If these tests fail the program lists all of the discrepencies and allows you to select the best method of correction.

The EDIT BASE INFO softkey returns you to the screen where data base name, passwords, and other information about the whole data base is entered.

The DATA SET OPTIONS softkey branches to that level. From there items and sets may be altered to correct duplicate name problems, key and search item discrepencies, or to link detail sets to an unlinked automatic master.

DATA BASE OPTIONS returns you to the DATA BASE OPTIONS level. You may continue editing from that point, go to the SHOW subsystem to get a better picture of the outstanding inconsistencies in the data base definition, or write the incomplete copy of the root file to disc.

Note

You may store an inconsistent copy of the root file to modify at a later time. This data base cannot be created with the DBCREATE command until the inconsistencies have been corrected.

Data Item Options

At the beginning of defining a data item, one or more of these softkeys are displayed. The softkeys displayed depends on whether there are other items defined in the data base or that data set.

As with the Data Set Options, only active keys are displayed. If the first set is being defined, only the DEFINE A DATA ITEM softkey is present. Subsequent new sets also include MODIFY & RENAME and ADD A DATA ITEM. When one or more items have been added to the set, ALTER & RESTORE and ADD A DATA ITEM softkeys appear. If an automatic master set has one item or if a manual master or detail set has 1 023 items, only ALTER & RESTORE and DELETE A DATA ITEM are present. Finally, if you're in the ALTER A DATA BASE mode, only the ALTER & RESTORE softkey is available.

Defining a Data Item

To define a new data item, press the **DEFINE A DATA ITEM** softkey.

Information specified for each data item is:

- data item field number within the current set.
- data item name
- pseudonyms
- data item type
- format extensions to the item type
- range of the data item if the type is INTEGER, DOUBLE, SHORT, REAL, or LONG REAL.
- code values if the type is CODE

Once the data item definition is complete, press the **PROCESS ITEM INFO** softkey.

Create a Data Item from an Existing Data Item

If you want to create a new data item from an existing data item definition, press the **MODIFY & RENAME** softkey. The definition of the original data item is unchanged. The screen displays the definition of the selected data item. If the item name or pseudonyms are not changed, the duplication is deleted when the entire definition is checked.

When you modify a data item which is CODE type, there must be enough room to store a maximum of 32 more codes in the code table (which contains the code values for the entire data base). If 32 code values cannot be stored in the code table, the maximum that can be stored is indicated by the number of fields displayed on the screen. If there is no room in the code table, no code items may be defined. For example, if 245 code values have already been defined for the data base and you want to MODIFY & RENAME a data item of the type CODE, a maximum of ten code values can be defined for the specified data item. Therefore, when the data item information is displayed on the screen, only ten fields may be accessed.

Changing the Definition of an Existing Data Item

To change the definition of a data item which has been previously defined for the current data set, press the **ALTER & RESTORE** softkey. When in the ALTER A DATA BASE mode, only the item name, pseudonyms, format, and ranges or codes may be changed. When you complete the changes, the new definition replaces the original definition of the data item. If you alter a data item which is also found in other data sets, those data items are also changed.

When altering an item in the ALTER A DATA BASE mode, the number of codes in the code table must not be decreased. This prevents code values already in the data base from losing their corresponding code string values. The code values may be changed and new values may be added however.

Deleting a Data Item Definition

The **DELETE DATA ITEM** softkey is used to delete a data item from the current data set. You select the data item to be deleted from a list of the defined data items on the screen. The selected data item is deleted from the current data set along with its associated paths. The item definition remains, even if not used in any set, until the data base defining is checked.

Adding an Existing Data Item to a Data Set

Press the **ADD A DATA ITEM** softkey to add a previously defined data item to the data set being defined. The selected item is then included in the definition of the current data set if possible. If the selected item is already found in the current data set, an error message is displayed and the data item is not added. Press the DATA ITEM OPTIONS softkey to cancel the ADD A DATA ITEM command.

The DATA SET OPTIONS softkey prompts you for necessary linking information before returning to the Data Set Options level.

If the set is an automatic master, nothing is requested. If the set is a manual master with more than one item which can be a key, then the program prompts you to select the key item. (The current first item is always the default.) If the set is in a detail set, and one or more items can be linked to the key items of existing master sets, then the program branches to the Select Search and Sort Item level where you may define paths from this detail set to master sets.

Selecting a Key Item for a Manual Master Set

The key item of a manual master set must be a scalar item of length no greater than 256 bytes. When you have finished defining items for a given set and you press the DATA SET OPTIONS softkey, a test is made to see if more than one item can be the key item. If more than one item can be the key value, they are listed to allow the user to re-select, if necessary, a new key item. The default is always the first item, the current key item. If you do select a new key item, enter the corresponding value and press (RETURN). The change is made and you branch to the Data Set Options level.

If one or more detail search items have already been linked to the current key item, the DELETE PATHS softkey appears. If you are changing the key item, you must press this key to delete all paths from the original key item.

The DATA ITEM OPTIONS softkey and the DATA SET OPTIONS softkey return you to the respective levels. In the latter case, the key item is left unchanged.

Defining Paths from Detail Sets to Master Sets
As with key items in master sets, search items and sort items in detail sets must be scalars no longer than 256 bytes. Additionally there must be a master set with a key item which matches in type and length. When you press DATA SET OPTIONS from the Data Item Options softkey level while defining a detail data set, you are prompted to select from those items which meet the criteria above, search and sort items. If this detail set has already been linking to master sets, search items are indicated by an uppercase"s", sort items are indicated by a lowercase"s". Enter the number corresponding to the desired search item and press RETURN then do the same for the sort item. If no sort item is desired, you may leave that field blank. After you press RETURN on the sort item field, the program branches to select a master set. The program lists all master sets whose key items match the search item in type and length. Enter the number corresponding to the desired master set and press RETURN.

When you have defined all the items for the set you are defining, press **DATA SET OPTIONS**. A data set must contain at least one data item, so if no data items were defined for a data set, QUERY/9000 asks you to:

- edit the definition of the data set (**EDIT SET INFO** softkey)
- select the data item options (**DATA ITEM OPTIONS** softkey)
- delete the definition of the data set (**DATA SET OPTIONS** softkey)

Every manual master data set must have a key item. A list of the items contained in the set is displayed and you need to select the search item. Once the search item is selected, the data set options are redisplayed.

If you wish to delete an existing path, enter the number corresponding the search item in the search item field and press the DELETE PATH softkey. The screen is redisplayed with the upper case "S" removed from the search item and if the path had a sort item, the lowercase "s" removed also.

As with selecting key items, DATA ITEM OPTIONS and DATA SET OPTIONS branch to the appropriate level. After the entire set definition is completed, press the DATA SET OPTIONS.

Creating the Data Base

When you press the **WRITE ROOT FILE** softkey, QUERY/9000 checks the data base definition to determine that the definition represents a valid data base and does final processing on the data base. If the data base was found to be incomplete before, this lower level check indicates the first problem it encounters. The root file is still written to disc but is not be usable as a data base.

If the data base is valid and the volume name can be found, QUERY/9000 attempts to create the data base directory and the root file. If the volume name cannot be found, insert the correct volume in the device and press the **VOLUME MOUNTED** softkey.

If the directory and root file were created successfully, you may create the data sets or return to the data base options.

When you press the **CREATE DATA SETS** softkey, you are asked to enter a maintenance word which is used to protect the root file and data set files. Purging, erasing, backing up, or modifying the data base cannot be done without specifying the maintenance word. If you specify a maintenance word and later forget it, the data base can never be purged, backed up, or modified.

If some sets are not created, they are listed on the CRT. (This is usually caused by overflowing the medium on which they reside.) They may be created using DBCREATE from the keyboard. After the data sets have been created, the data base options are displayed on the screen.

If you selected ALTER A DATA BASE originally, the new root file definition must be written over the old. The altered definition is checked but is not written to disc. When the softkey ALTER DATA BASE is pressed the root file is rewritten, the data base directory and DSET data set files are renamed if necessary, and DSET files are extended.

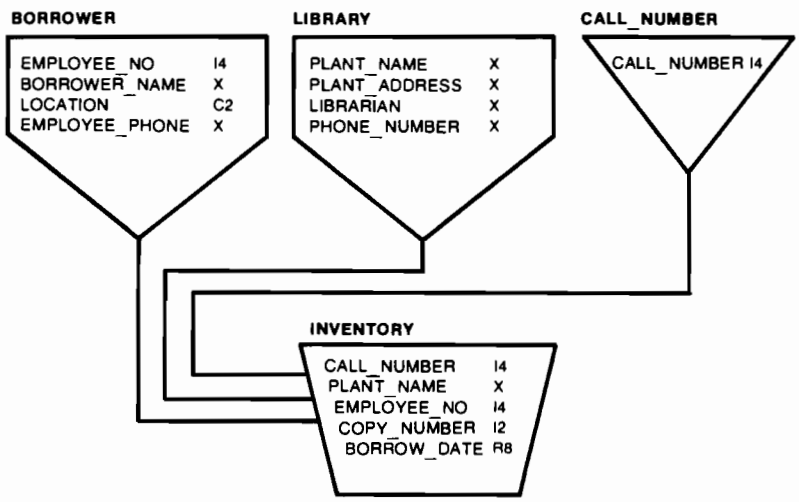
Should these operations fail for some reason, they are listed on the CRT. A RENAME on the appropriate file or DBEXTEND on the appropriate set can be used to complete the operation.

Note

The Data Base directory and all IMAGE and QUERY files are protected with the password IMAGE. If you have to do a COPY, PURGE, RENAME or other BASIC file operation, you must use this password.

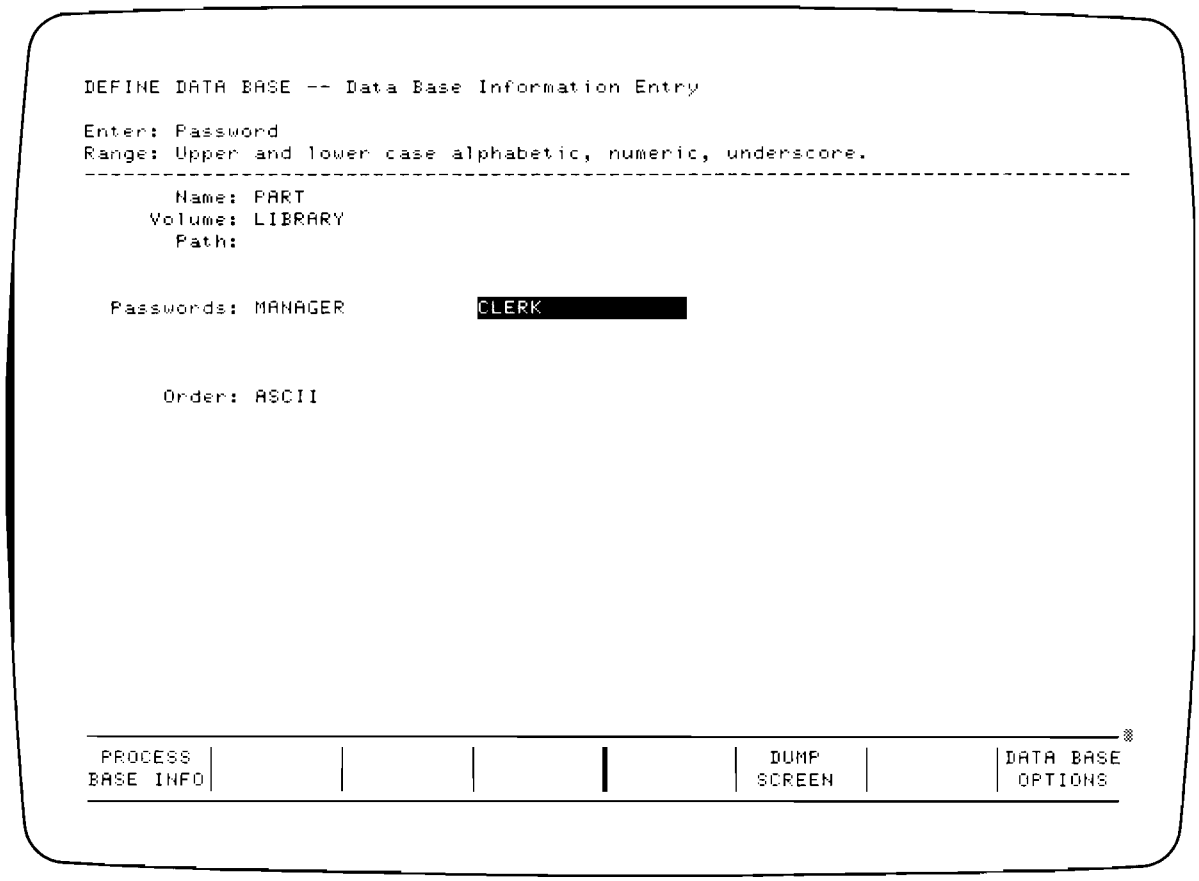
Example

The following example guides you through the definition of part of the LIBRARY data base. In addition to the medium on which QUERY/9000 is located, you will need a mini-floppy initialized to SDF format and labeled LIBRARY. This section also includes brief definitions (in shaded boxes) of important terms used while defining a data base. The diagram of the data base you are going to create is:



Data Base Information

1. For the data base name, type PART and press RETURN.
2. Fill out the rest of the screen to look like:



Defining the CALL_NUMBER Data Set

3. Press DEFINE A DATA SET at the Data Base Options softkey level.
4. For the first data set name, type CALL_NUMBER and press RETURN. The character between the two words is an underscore (_).
5. Press RETURN to enter the volume name of LIBRARY and RETURN again to keep the directory paths as a blank.
6. The data set type is automatic master, so type A. This type is indicated by the triangle shape in the diagram.
7. For the maximum capacity, type 23. Remember, since this is a master data set, the capacity should be a prime number.
8. For passwords, keep the W (for READ/WRITE access) for MANAGER and type R (for READ access) for CLERK.
9. Press the PROCESS SET INFO softkey to record the information.

```
DEFINE DATA BASE -- Data Set Information Entry        Data Base: PART

Enter: Password Access
Range: W (Read/Write), R (Read Only), Blank (No Access)
--------------------------------------------------------------------------------
      Number: 1
        Name: CALL_NUMBER
      Volume: LIBRARY
        Path:


        Type: A
    Capacity: 23
      Access: W MANAGER             R CLERK                 Public Access
```

PROCESS SET INFO				DATA SET OPTIONS	DUMP SCREEN		DATA BASE OPTIONS

The information about the first data set has been entered; now the data items in this data set (CALL_NUMBER) need to be entered.

Defining the Data Items for CALL_NUMBER

10. Press DEFINE A DATA ITEM at the Data Item Options softkey level.
11. Type `CALL_NUMBER` for the data item name.
12. You are now asked for the pseudonyms for the data item name, CALL_NUMBER.

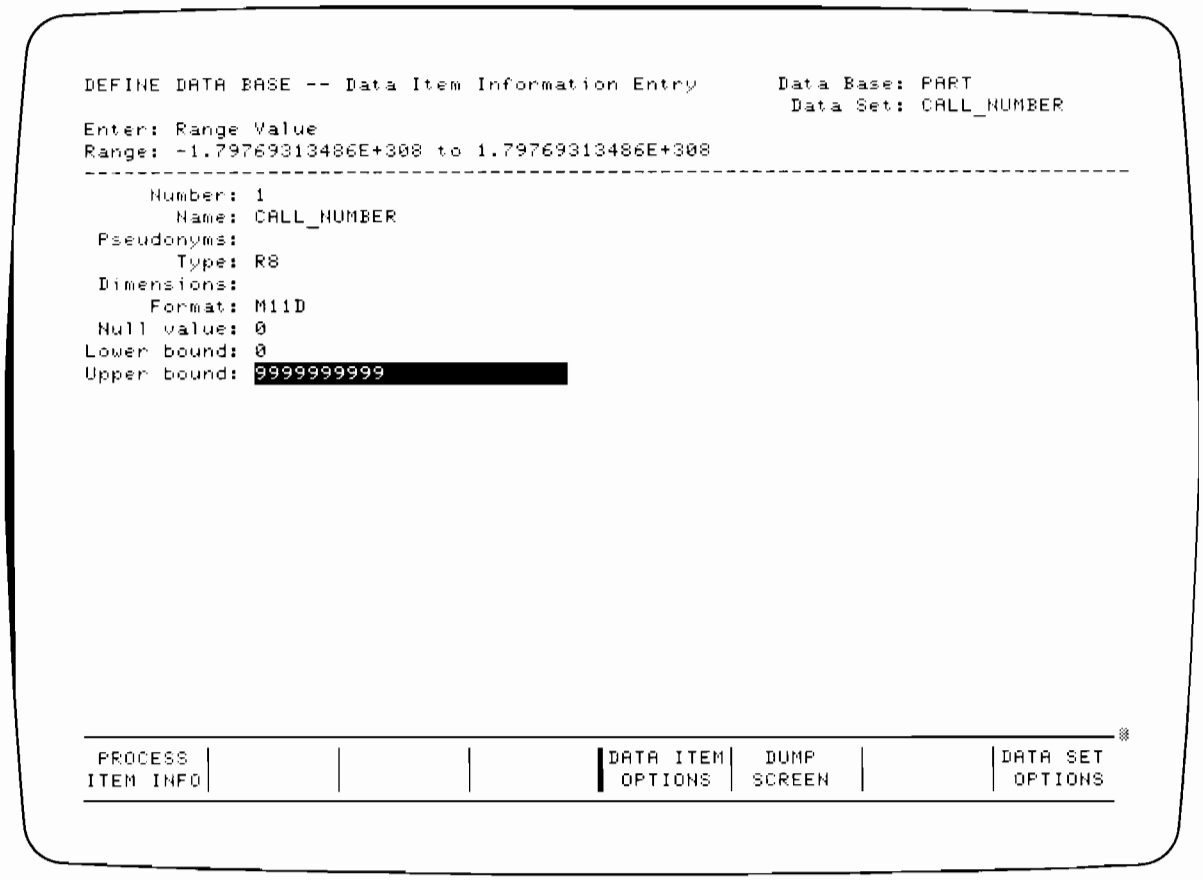
Pseudonyms are names that are used interchangeably with item names. For example, WRITER can be used instead of AUTHOR. Pseudonyms are useful when several people use a data base and the exact item names might not be remembered, or to abbreviate a long item name to just a few characters.

13. Press [↓] to leave all the pseudonyms fields blank so that no pseudonyms are defined for the item name CALL_NUMBER.
14. Indicate that this data item is of type LONG REAL by typing `R8`.
15. Enter M11D for the format. This means that the item is output with a minus sign and up to 11 integer digits only.

Items may have formatting extensions added to them. In the case of strings, a name extension may be added. This puts the input string in the canonical form for names. The output format of numbers may be controlled by specifying a sign for all numbers or negative only, the number of digits in the integer, the number of digits in the fraction, and, if desired, the number of digits in the exponent. LONG REAL numbers may be interpreted as DATE, TIME, or TIMEDATE according to the BASIC time date convention. You may also add your own extensions to each item by entering USER(n) where n is a number from 1 to 2 147 483 647. This may be retrieved by DBINFO in your IMAGE program.

16. The maximum range of a LONG REAL number is displayed (with lower and upper bounds). Call numbers are positive, so change the lower bound to `0`.

You are allowed to specify a null value as well as upper and lower bounds for each numeric item in the data base. IMAGE uses this null value for a list mode (DBPUT or DBUPDATE) where the numeric item is not included in the list. QUERY also uses the null value when a blank (no value) is entered for a numeric item. Note that if the null value is in range, a blank input in the update sybsystem is redisplayed by QUERY as the value entered here. If the null value falls outside the range, the value is redisplayed as a blank. Thus, the null value can be used as a default value (when in range) or a missing value (when out of range).

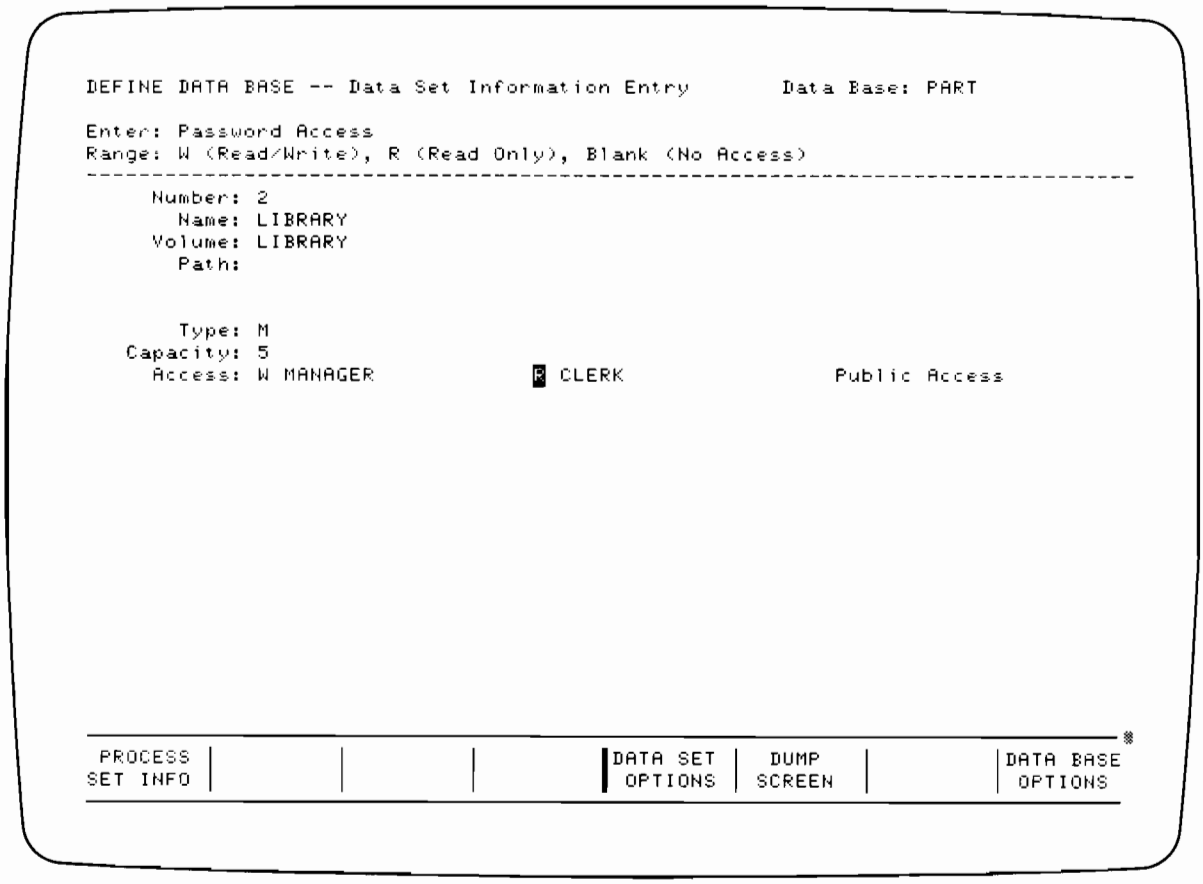


You may use the TAB and SHIFT TAB key or UP and DOWN arrow keys to edit any fields then press the PROCESS ITEM INFO softkey to put the definition in with the rest of the data base definition.

Since this is an automatic master set, only one data item can be defined. DEFINE then gives you the choice of changing the definition of the item or deleting the item. You currently want to leave the definition as it is, so you now can define another data set.

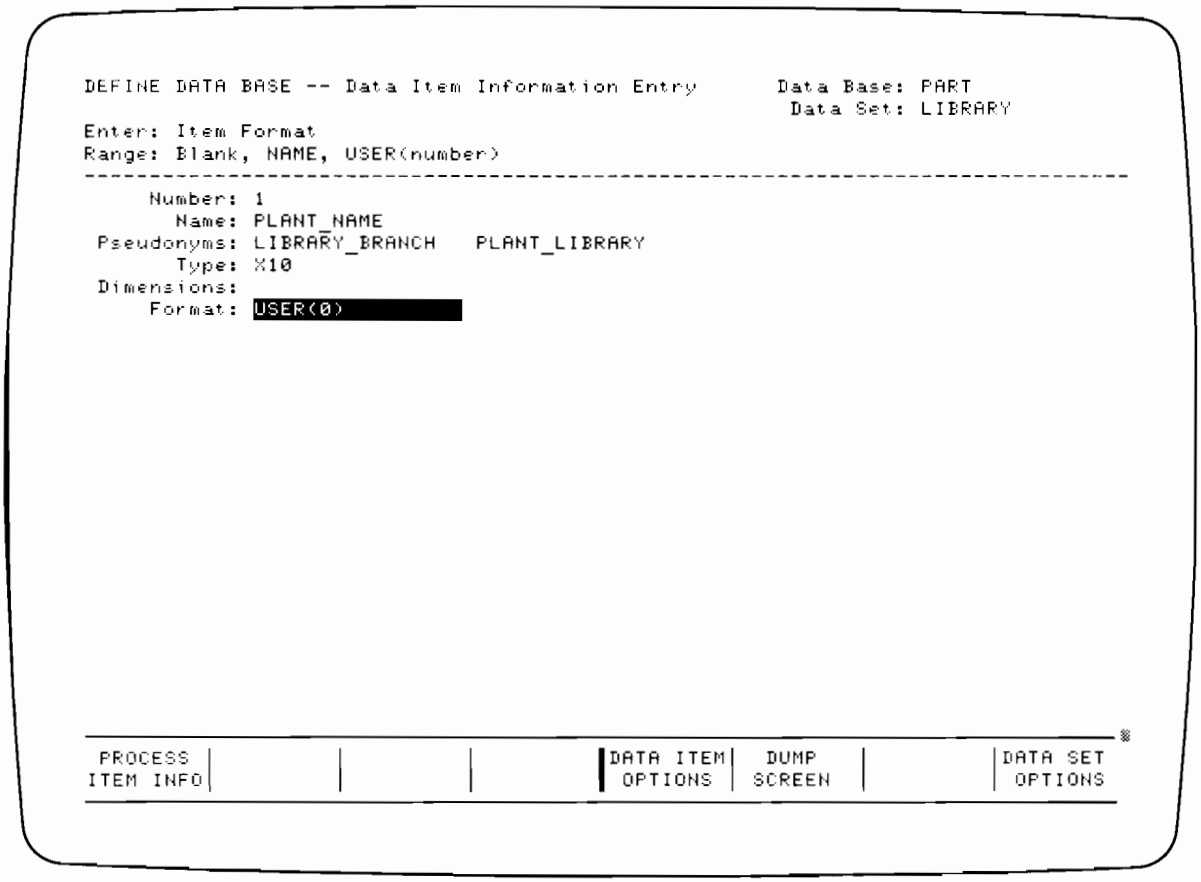
Defining the LIBRARY Data Set

17. Press the DATA SET OPTIONS softkey.
18. Now press the DEFINE A DATA SET softkey.
19. Enter the following information for the LIBRARY manual master data set. When the screen is completed, press the PROCESS SET INFO softkey.

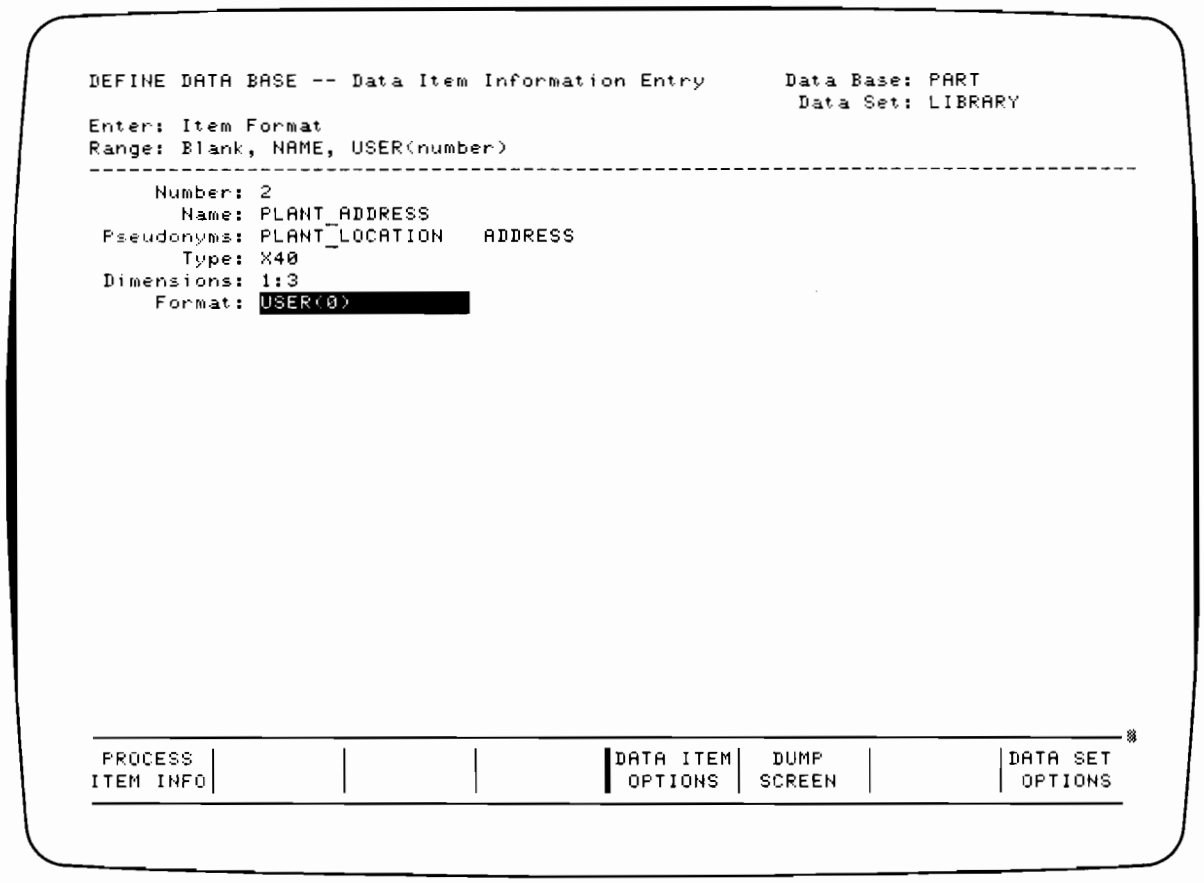


Defining the Data Items for LIBRARY

20. Now press the DEFINE A DATA ITEM softkey to define the data items in the LIBRARY set.
21. Type PLANT_NAME for the item name.
22. For pseudonyms, type LIBRARY_BRANCH and press RETURN, and type PLANT_LIBRARY.
23. Press RETURN twice to move past the pseudonym section.
24. Type X10 to specify that PLANT_NAME is a character-type (string) data item with ten characters. The length of each item is indicated on the data base diagram shown previously.
25. Now press the PROCESS ITEM INFO softkey.



The screen for the next item should be filled out as follows. When it is completed, press the PROCESS ITEM INFO softkey.



Note that this is an array item with three elements in one dimension. You may define up to six dimensions giving lower and upper bound for each. An array item may not be used as a key, search, or sort item.

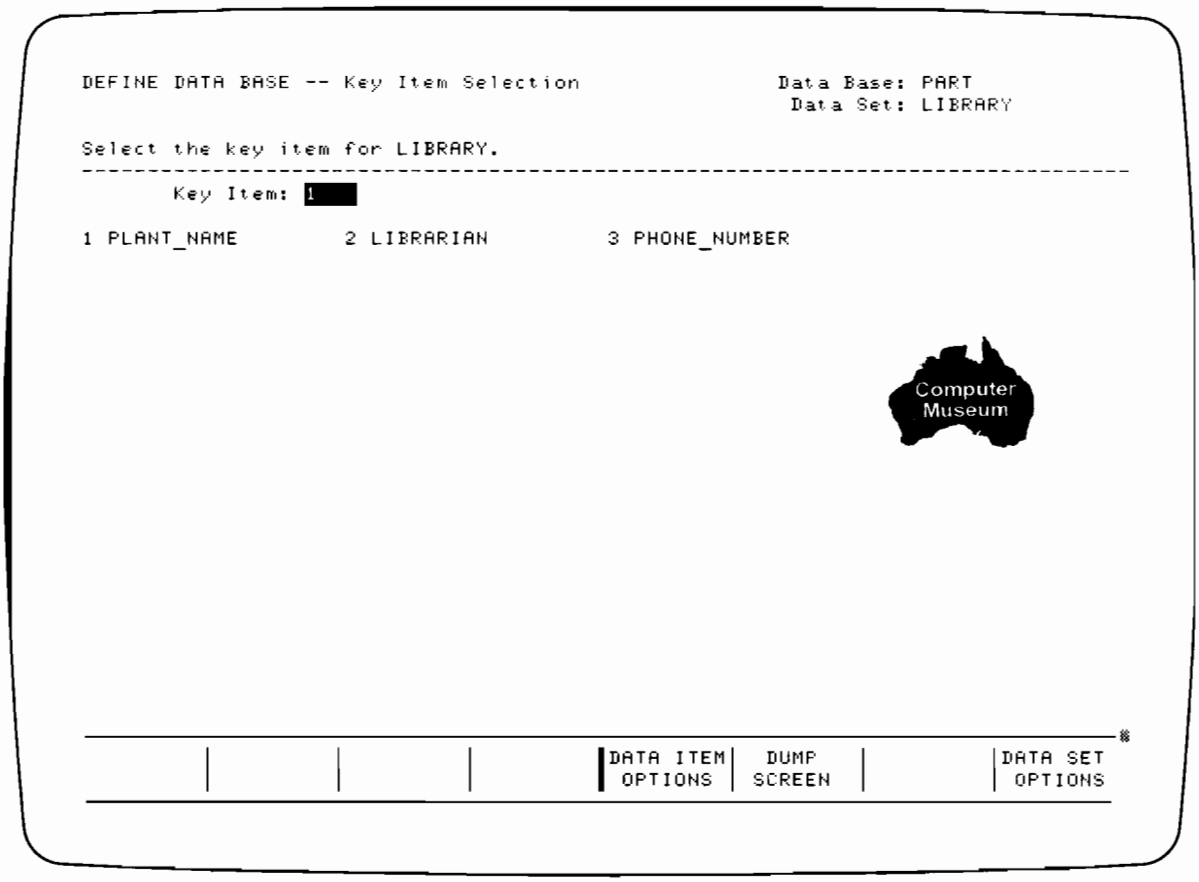
For the next two items, fill in the screens as shown, pressing the PROCESS ITEM INFO softkey after each one.

```
DEFINE DATA BASE -- Data Item Information Entry      Data Base: PART
                                                      Data Set: LIBRARY
Enter: Item Format
Range: Blank, NAME, USER(number)
--------------------------------------------------------------------------------
     Number: 3
       Name: LIBRARIAN
 Pseudonyms: HEAD_LIBRARIAN
       Type: X50
 Dimensions:
     Format: NAME

 PROCESS  |          |          |          | DATA ITEM | DUMP    |          | DATA SET
ITEM INFO |          |          |          |  OPTIONS  | SCREEN  |          |  OPTIONS
```

```
DEFINE DATA BASE -- Data Item Information Entry      Data Base: PART
                                                      Data Set: LIBRARY
Enter: Item Format
Range: Blank, NAME, USER(number)
--------------------------------------------------------------------------------
     Number: 4
       Name: PHONE_NUMBER
 Pseudonyms: PHONE            TELEPHONE        TELEPHONE_NO     LIBRARY_PHONE
       Type: X14
 Dimensions:
     Format: USER(0)

 PROCESS  |          |          |          | DATA ITEM | DUMP    |          | DATA SET
ITEM INFO |          |          |          |  OPTIONS  | SCREEN  |          |  OPTIONS
```

Now that you have defined the data items, press the DATA SET OPTIONS softkey. At this point, DEFINE asks you for the key or search item.

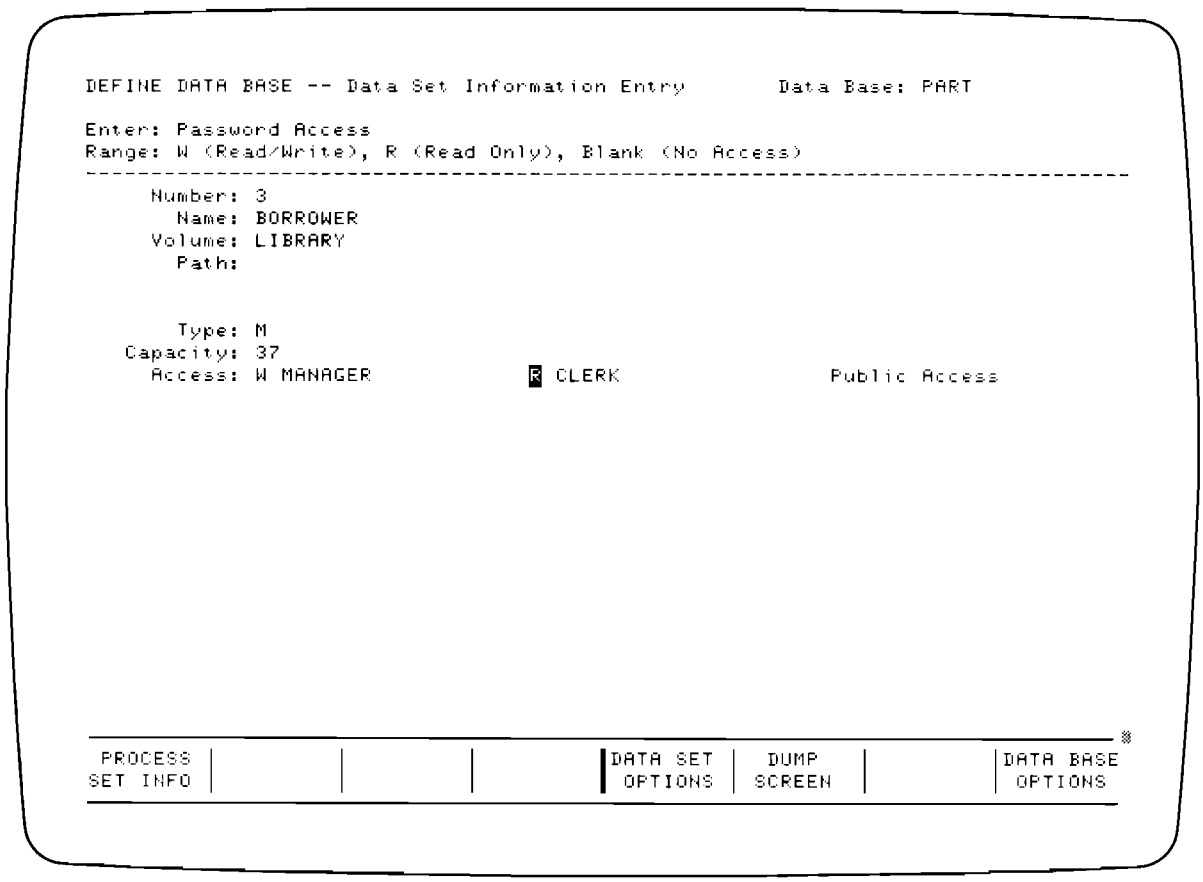
26. Type 1 to indicate PLANT_NAME is the key item. 1 is the item number of PLANT_NAME.



You are now ready to define another data set, so press the DEFINE A DATA SET softkey.

Defining the BORROWER Data Set

27. Type `BORROWER` for the name of the data set.
28. Press RETURN to leave the volume name as LIBRARY; press RETURN again to leave the directory path blank.
29. This is a manual master data set, so type `M` for the set type.
30. Type `37` for the capacity.
31. Change the CLERK access to READ ONLY access by typing `R` before CLERK while leaving the W before MANAGER.
32. Press the PROCESS SET INFO softkey.

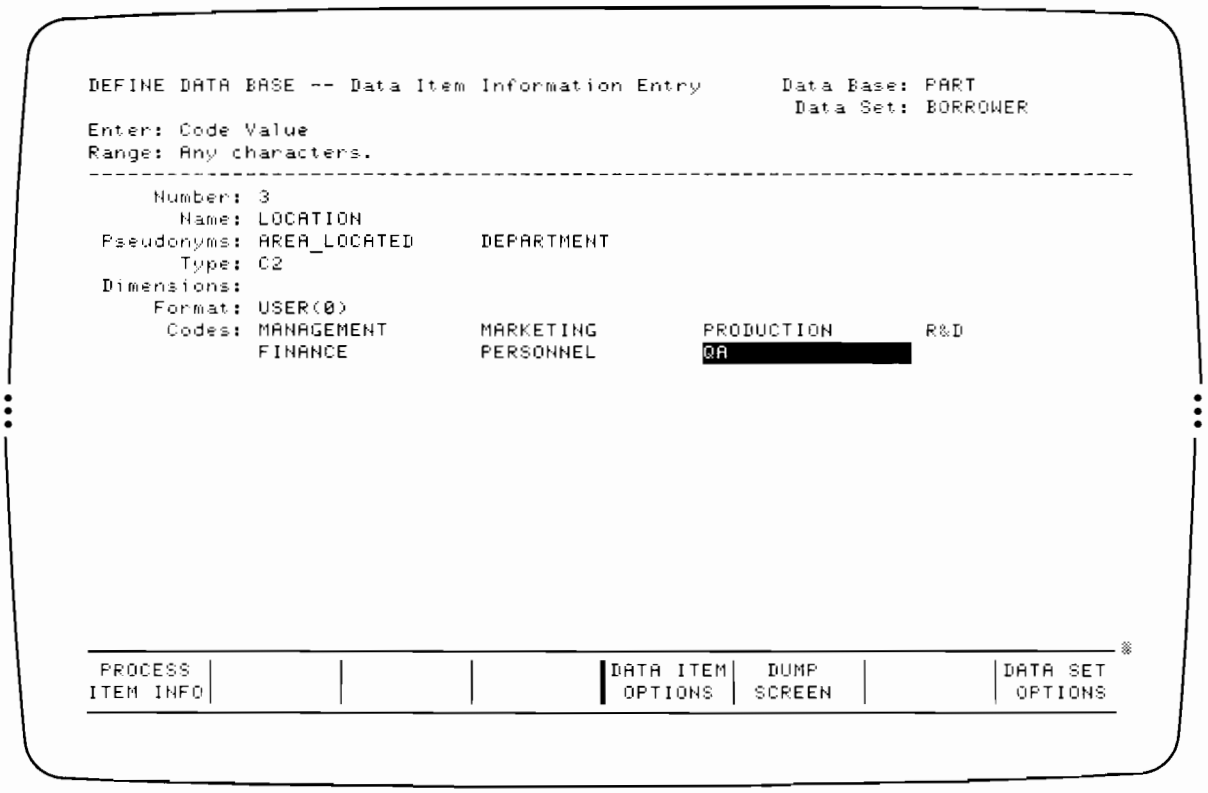


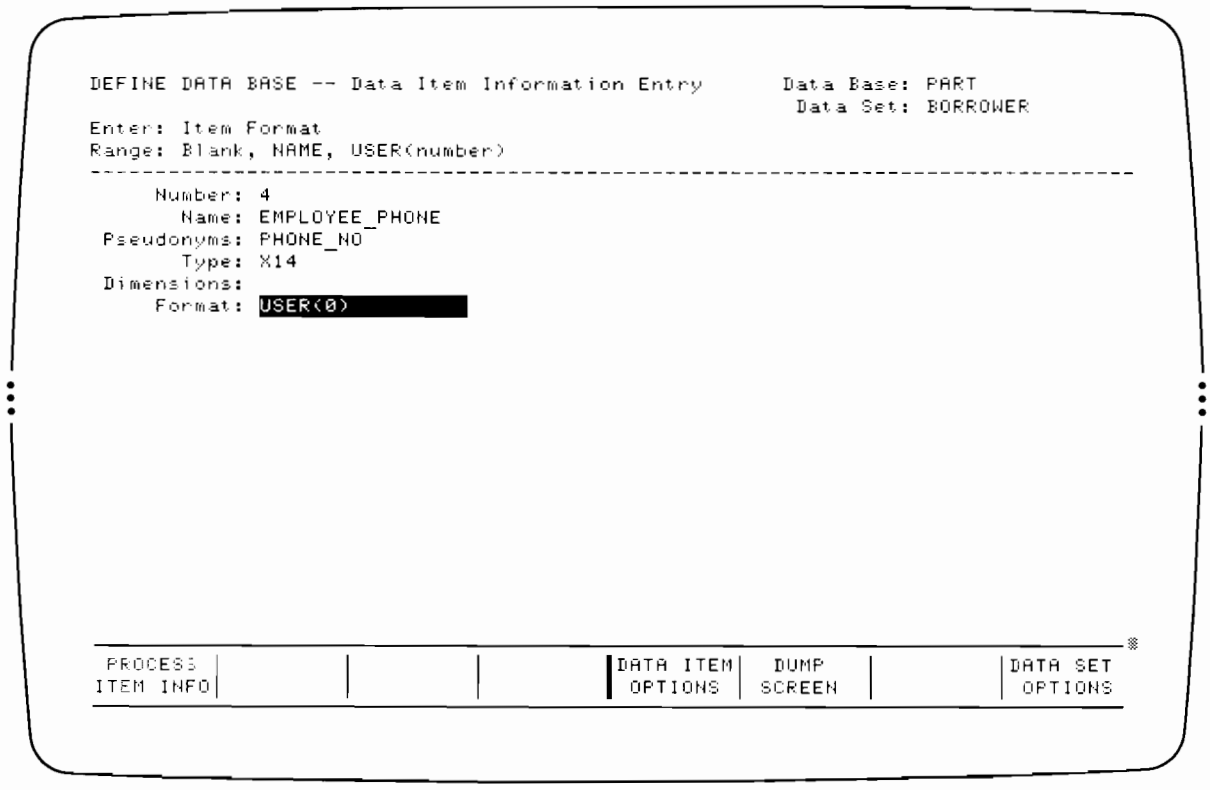
Defining the Data Items for BORROWER

You are now ready to define the items in the BORROWER data set. Complete the screens as shown:

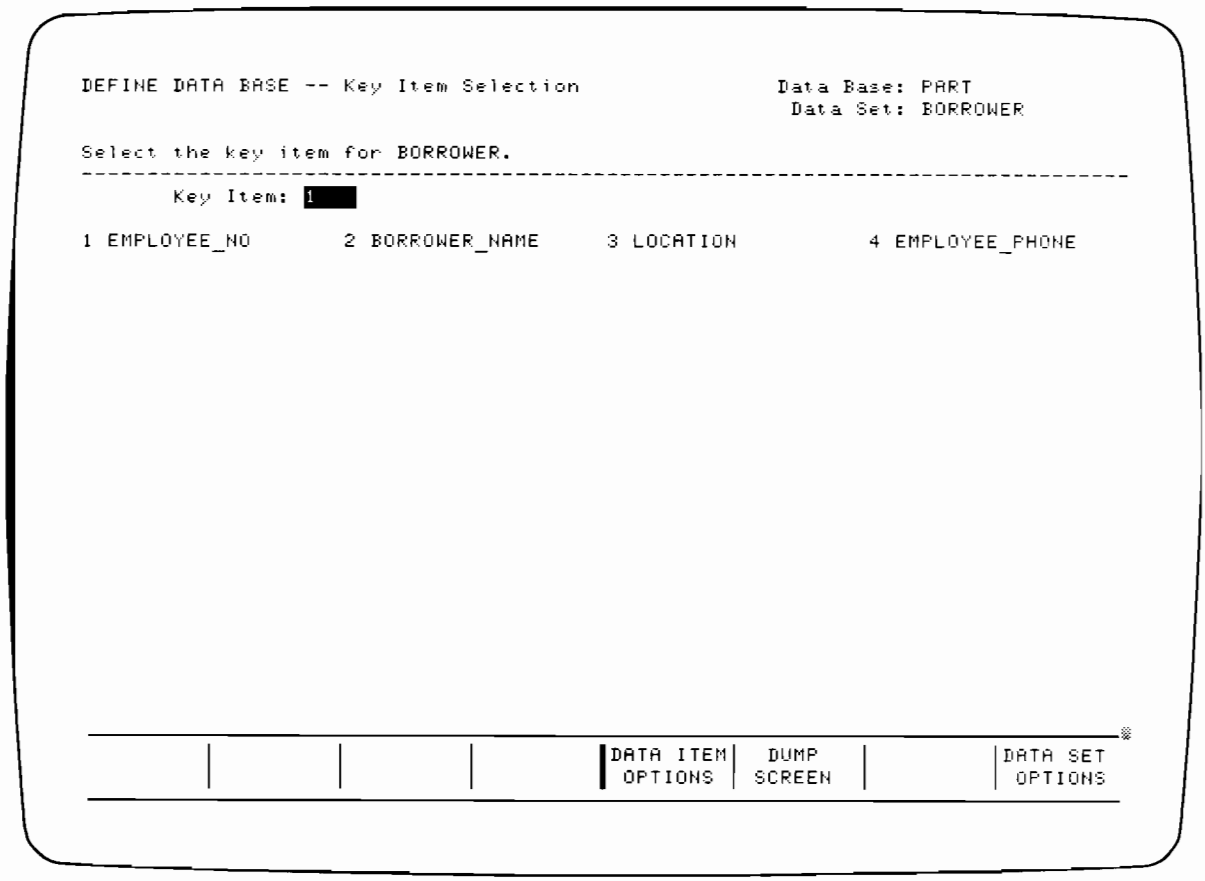
```
DEFINE DATA BASE -- Data Item Information Entry      Data Base: PART
                                                      Data Set: BORROWER
Enter: Range Value
Range: -32768 to 32767
-------------------+-----------------------------------------------------------
     Number: 1
       Name: EMPLOYEE_NO
 Pseudonyms: EMPLOYEE
       Type: I2
 Dimensions:
     Format: M6D
 Null value: 0
Lower bound: 0
Upper bound: 32767

 PROCESS    |          |          |          | DATA ITEM | DUMP      |          | DATA SET
ITEM INFO   |          |          |          |  OPTIONS  | SCREEN    |          |  OPTIONS
```

```
DEFINE DATA BASE -- Data Item Information Entry      Data Base: PART
                                                      Data Set: BORROWER
Enter: Item Format
Range: Blank, NAME, USER(number)
-------------------------------------------------------------------------------
     Number: 2
       Name: BORROWER_NAME
 Pseudonyms: BORROWER
       Type: X50
 Dimensions:
     Format: NAME

 PROCESS    |          |          |          | DATA ITEM | DUMP      |          | DATA SET
ITEM INFO   |          |          |          |  OPTIONS  | SCREEN    |          |  OPTIONS
```


DEFINE DATA BASE -- Data Item Information Entry
Data Base: PART
Data Set: BORROWER
Enter: Code Value
Range: Any characters.
Number: 3
Name: LOCATION
Pseudonyms: AREA_LOCATED DEPARTMENT
Type: C2
Dimensions:
Format: USER(0)
Codes: MANAGEMENT MARKETING PRODUCTION R&D
FINANCE PERSONNEL QA
PROCESS ITEM INFO
DATA ITEM OPTIONS
DUMP SCREEN
DATA SET OPTIONS


DEFINE DATA BASE -- Data Item Information Entry
Data Base: PART
Data Set: BORROWER
Enter: Item Format
Range: Blank, NAME, USER(number)
Number: 4
Name: EMPLOYEE_PHONE
Pseudonyms: PHONE_NO
Type: X14
Dimensions:
Format: USER(0)
PROCESS ITEM INFO
DATA ITEM OPTIONS
DUMP SCREEN
DATA SET OPTIONS

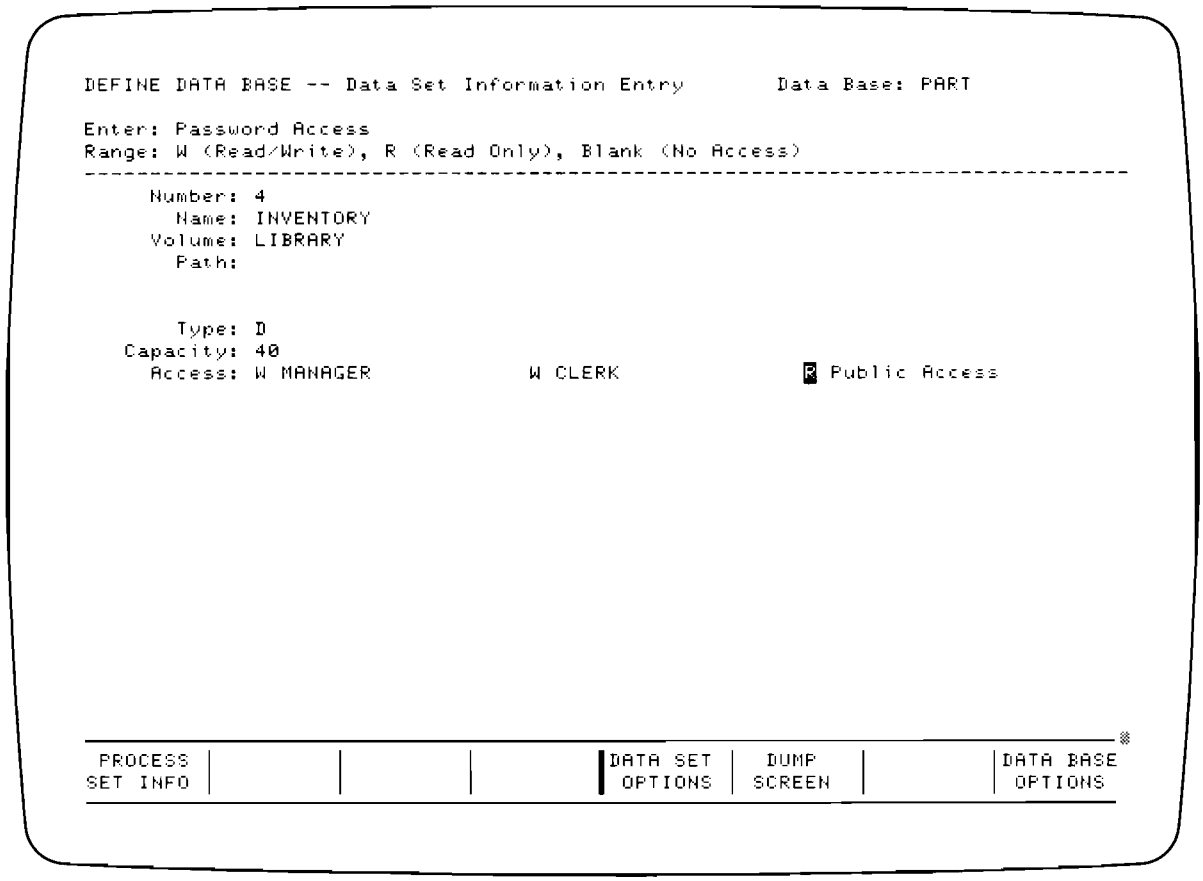
33. After all the data items are entered, press the DATA SET OPTIONS softkey.
34. Type `1` to specify that the key item for the BORROWER data set is EMPLOYEE_NO.



Defining the INVENTORY Data Set

35. Press DEFINE DATA SET to define the last data set.
36. Type `INVENTORY` for the name of the data set.

Fill in the rest of the screen as shown:

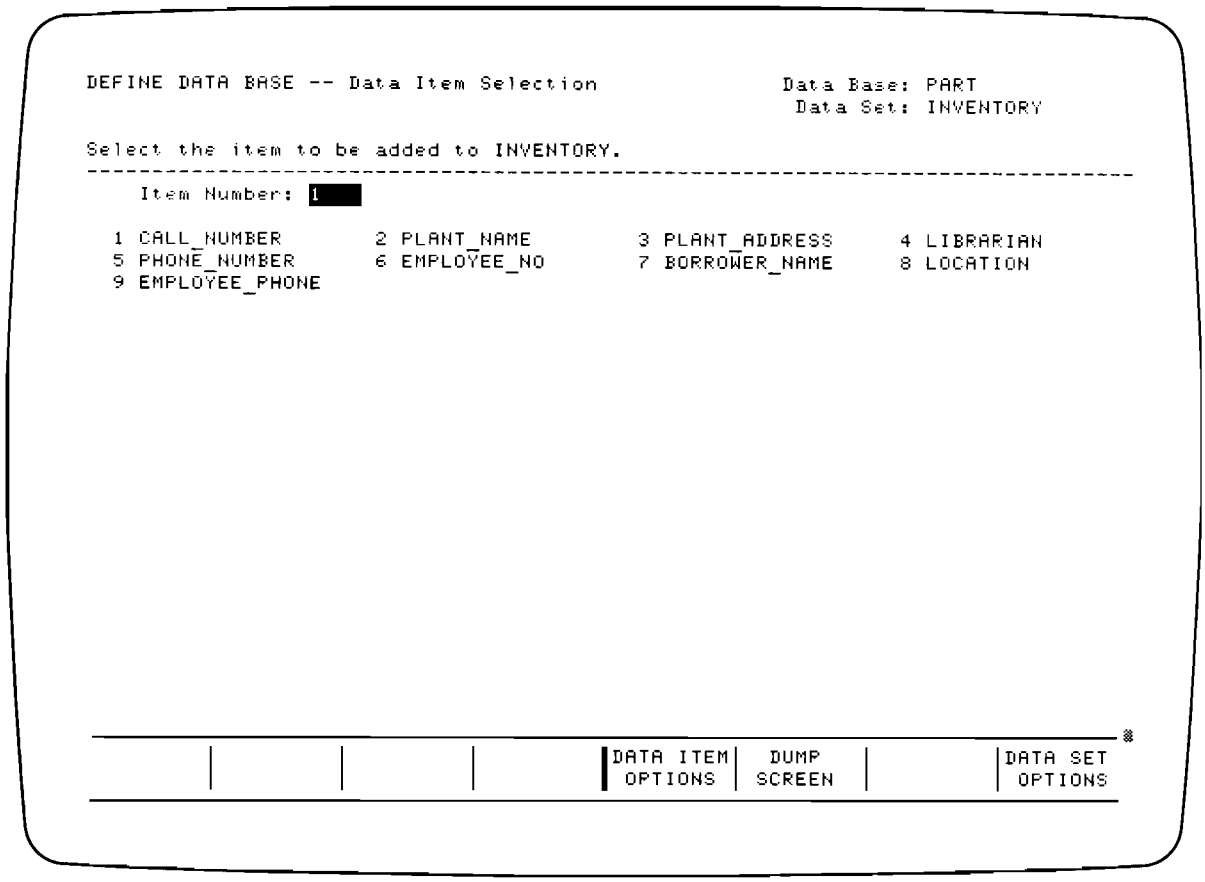


37. Press PROCESS SET INFO softkey to enter the data set information.

Defining the Data Items for INVENTORY

Since some of the items to be included in this set have already been defined, these can simply be added to this set as well.

38. Press ADD A DATA ITEM. Select CALL_NUMBER and press RETURN to add it to the INVENTORY data base.



39. Now press the DATA ITEM OPTIONS softkey to define other data items contained in INVENTORY.
40. Then press the DEFINE A DATA ITEM softkey.

41. Define the next two items as follows, pressing the PROCESS ITEM INFO softkey after each screen is completed.

```
DEFINE DATA BASE -- Data Item Information Entry        Data Base: PART
                                                        Data Set: INVENTORY
Enter: Item Format
Range: Blank, NAME, USER(number)
--------------------------------------------------------------------------------
     Number: 2
       Name: COPY_NUMBER
Pseudonyms: COPY              COPY_NO           NUMBER_OF_COPY
       Type: X10
 Dimensions:
     Format: USER(0)

 PROCESS    |          |          |          | DATA ITEM | DUMP      |          | DATA SET
ITEM INFO   |          |          |          |  OPTIONS  | SCREEN    |          |  OPTIONS
```

```
DEFINE DATA BASE -- Data Item Information Entry        Data Base: PART
                                                        Data Set: INVENTORY
Enter: Range Value
Range: 1 Jan -1469899 to 31 Dec 1469899
--------------------------------------------------------------------------------
     Number: 3
       Name: BORROW_DATE
Pseudonyms: DATE_BORROWED
       Type: R8
 Dimensions:
     Format: DATE
 Null value: 31 Dec 1936
Lower bound: 1 Jan 1937
Upper bound: 31 Dec 2000

 PROCESS    |          |          |          | DATA ITEM | DUMP      |          | DATA SET
ITEM INFO   |          |          |          |  OPTIONS  | SCREEN    |          |  OPTIONS
```

42. Now press the DATA ITEM OPTIONS since you need to define the two remaining key items.
43. Press ADD A DATA ITEM.
44. To add PLANT_NAME to the INVENTORY data set, enter 2 and press (RETURN).
45. Enter 6 and press (RETURN) to add EMPLOYEE_NO as well.

Linking INVENTORY to Master Data Sets

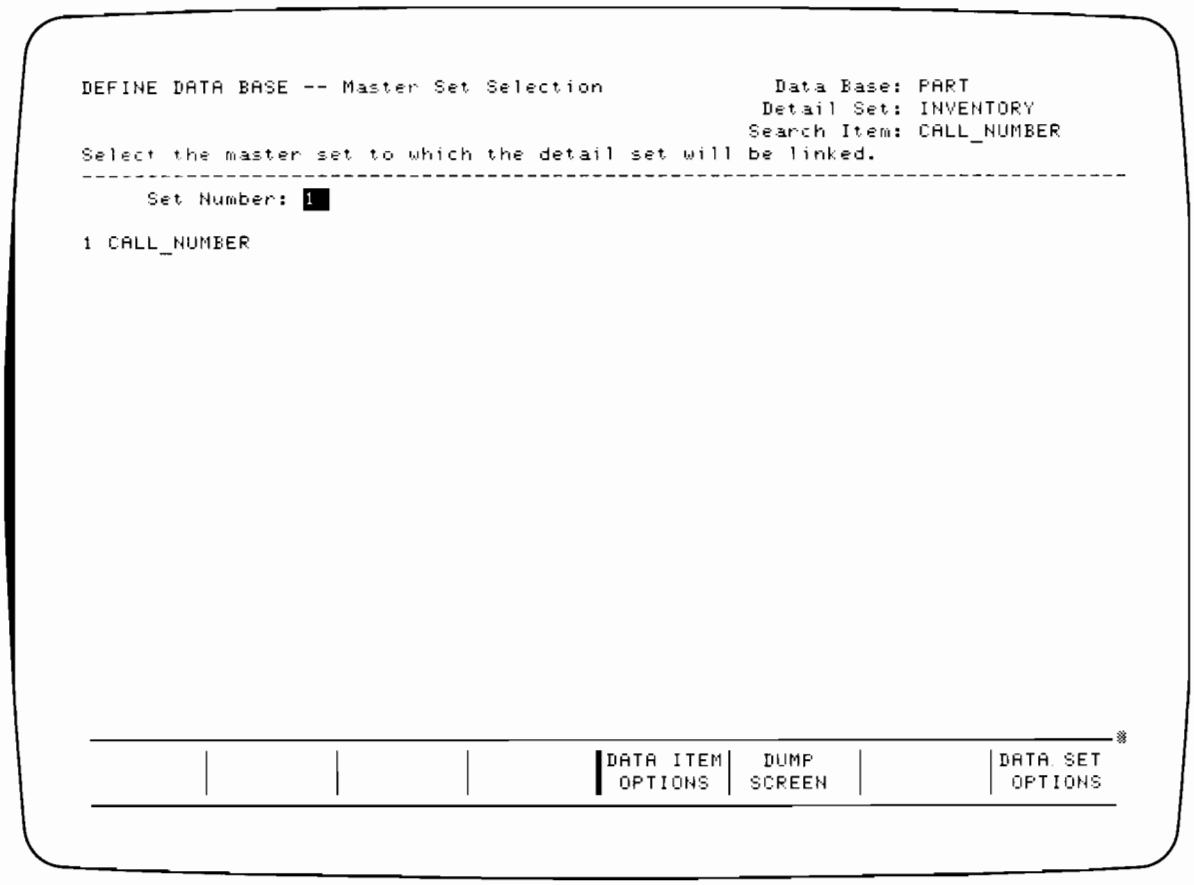
Paths are created by selecting a search item and optionally a sort item from the detail data set, then you select a master set to which the search item may be linked.

46. Press the DATA SET OPTION softkey to indicate that all items have been defined. Since this is a detail set, you can define paths which conect it to various master sets.
47. Select CALL_NUMBER as the search item and COPY_NUMBER as the sort item by entering 1 and pressing (RETURN), then entering 2 and pressing (RETURN)

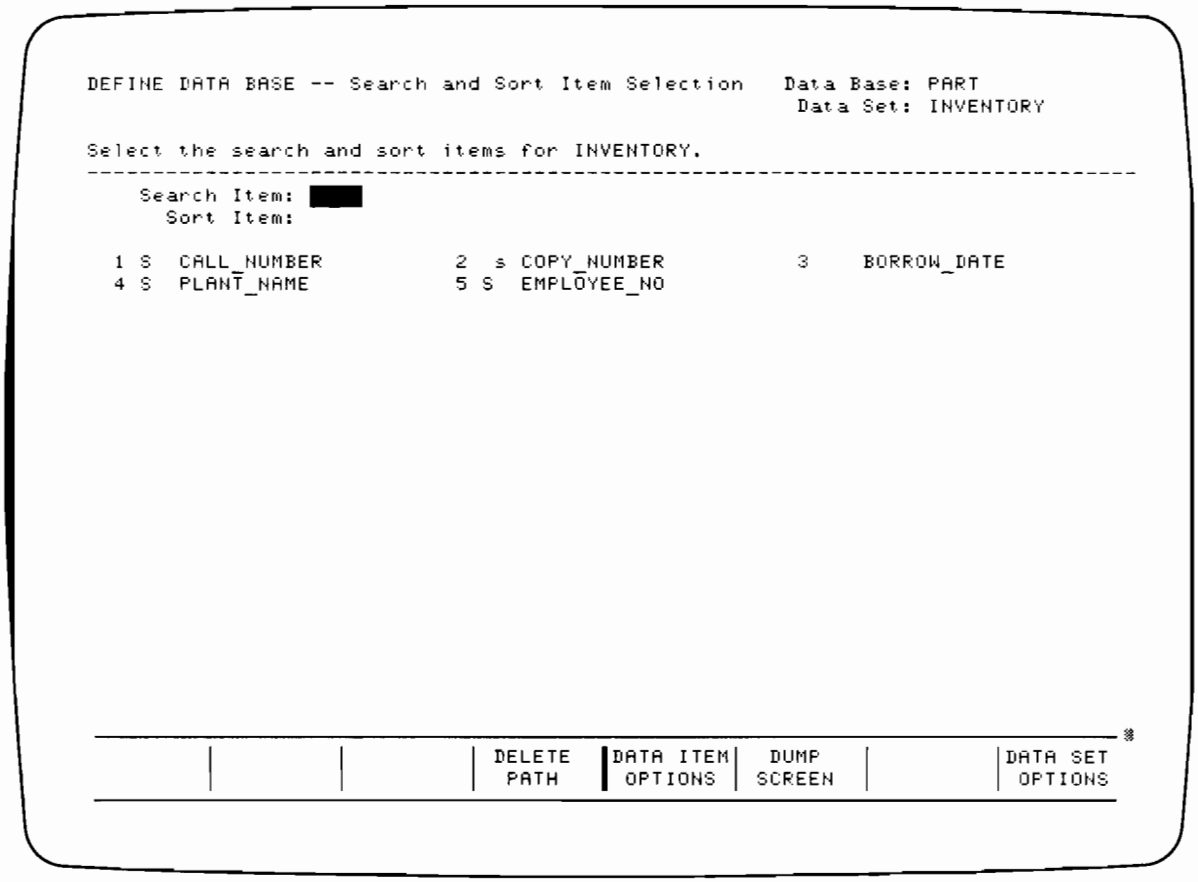
```
DEFINE DATA BASE -- Search and Sort Item Selection    Data Base: PART
                                                        Data Set: INVENTORY

Select the search and sort items for INVENTORY.
--------------------------------------------------------------------------------
    Search Item: 1
      Sort Item: 2

  1    CALL_NUMBER          2    COPY_NUMBER          3    BORROW_DATE
  4    PLANT_NAME           5    EMPLOYEE_NO

         |         |         | DELETE  |DATA ITEM| DUMP    |         |DATA SET
         |         |         |  PATH   | OPTIONS | SCREEN  |         | OPTIONS
```

All of the master sets whose key items may be linked to CALL_NUMBER are displayed.

48. Press RETURN to select the CALL_NUMBER master set as the one to which the CALL_NUMBER search item is linked.



Now that CALL_NUMBER is a search item and COPY_NUMBER is a sort item, these are indicated when the screen is redisplayed. Uppercase "S" appears by the former, lowercase "s" appears by the latter.



49. Enter 4 and press RETURN twice to form a path with PLANT_NAME as the search item; press RETURN to select PLANT_NAME as the master set for this path.
50. Enter 5 and press RETURN twice for an unsorted path with a search item of EMPLOYEE_NO. RETURN again links this to the BORROWER manual master data set.
51. Press DATA SET OPTIONS then DATA BASE OPTION since all items for INVENTORY and all sets for PART have been defined. The data base structure is checked for consistency.

Creating the Data Base

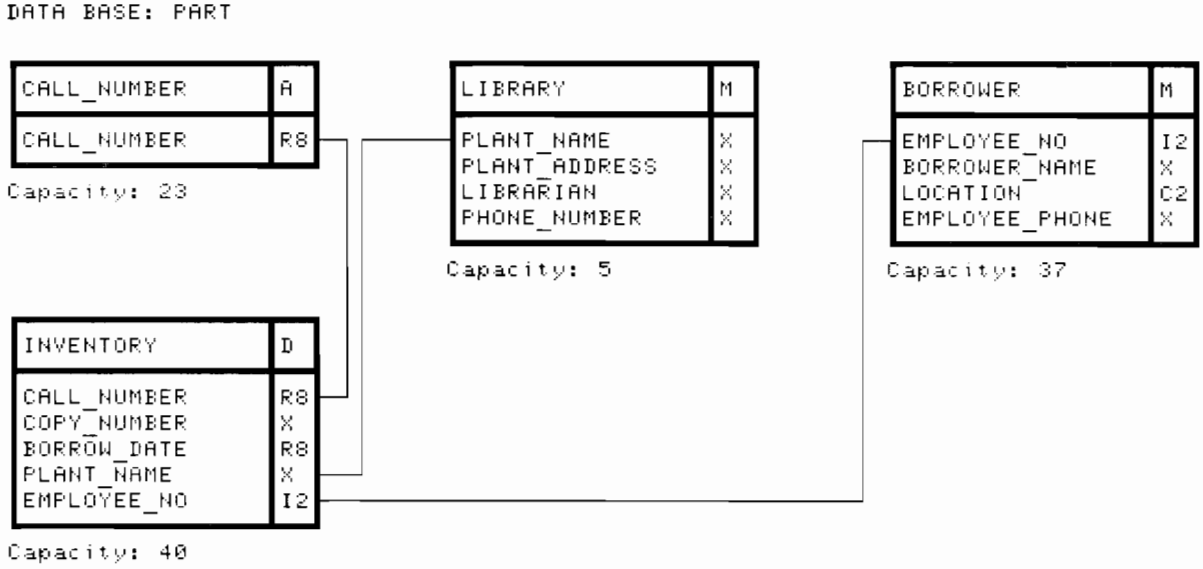
The PART data base is now ready to be created. This involves processing the set and item definitions, then creating the root file, information file and data set files.

52. Press the WRITE ROOT FILE softkey.
54. When the data base directory and root file have been created, press the CREATE DATA SETS softkey.
54. For the maintenance word, just press (RETURN).

A **maintenance word** is used to protect the data base from purging or restructuring by unauthorized people. It is similar to a password, except that passwords enable only certain people to update the the data in a set, where as a maintenance word helps prevent purging or restructuring of the whole data base. If the field is left blank, no maintenance word exists so anyone can purge or restructure the data base.

55. Press EXIT SUBSYSTEM to get out of the DEFINE subsystem.

Now your data base is ready for you to add data. If you were to get a graphic structure of the PART data base, it would look like this:



Chapter 9

Linking Other Subprograms to QUERY/9000

Run User Program is a feature of QUERY/9000 which enables you to write, load and execute one or more subprograms, using them as extensions to the main QUERY program. The subprograms can perform various operations, such as mathematical calculations and report generation. They have access to all COMMON variables defined by QUERY/9000 and may call various utility subprograms, which are described later in this chapter. These extensions should be written by experienced programmers, however, anyone can use the extension once it is written.

This chapter contains the information necessary for a programmer to write subprograms for use as extensions to QUERY/9000. The following information is included:

- How Run User Program Works
- Restrictions on the "Extend" Subprogram
- Using the Search Result
- Using COMMON Variables
- Descriptions of the Utility Subprograms
- An Example

How Run User Program Works

The Run User Program feature can be accessed either immediately after opening a data base or after a search has been performed. It is accessed by pressing the RUN USER PROGRAM softkey. When this is done, you are asked to enter the file name and the volume label of the file which contains the subprograms you have written.

When Run User Program is accessed, QUERY/9000 deletes all subprograms from memory except the utility subprograms described later. This maximizes the amount of memory available for your subprograms and variables. The exact amount of memory available depends on the size of the data base currently open and the configuration of your HP 9000 System. As an estimate, an HP 9000 Model 20 with 1Mbyte of memory with the ERROR (English), Mass Storage and IMAGE_DBM options loaded has approximately 275K bytes of memory available for extension subprograms.

The Extend Subprogram

The first subprogram in the file which you load must be called "Extend". QUERY/9000 begins running your extension subprograms by executing CALL "Extend". The remainder of the subprograms you want to use should follow the "Extend" subprogram. "Extend" can call them as necessary.

Here is an example of how a file of extension subprograms is structured:

File Name: Report

SUB Extend ...
SUB Compute
SUB Print

Restrictions on the "Extend" Subprogram

The file loaded by Run User Program must contain a subprogram named "Extend", which in turn can call the other subprograms. The first few lines of this subprogram must appear as follows:

```
    10  SUB Extend
    20    OPTION BASE 1
    30    COM /Mass_storage/ Storage$[150],Path$(1:3)[226],Base$(1:2)[16],Volume
$(1:3)[16]
    40    COM /Loader/ INTEGER Mode,Segment
    50    COM /Db_manipulation/ DOUBLE Err,Sts(1:10),INTEGER Acc_sets,All_sets,D
b$[284],Bf1$[2050],Bf2$[196]
    60    COM /Input/ Keys$[160],INTEGER Key,Keys
    70    COM /Command/ Recall$[321],Command$[321],Token$,INTEGER Token,Lineptr1
,Lineptr2
    80    COM /Search_syntax/ INTEGER Recursion_depth,Subptr1,Subptr2,Search_ite
m,Optimize,Thread_number
    90    COM /Search_result/ INTEGER Set_cnt,Thread(1:10,1:2),DOUBLE Select_dep
th,Record_cnt,Record_len,Max_rec_len
   100    COM /Update/ INTEGER Autoclear,Search,Updates
   110    COM /Sort/ INTEGER Key_cnt,Entry_cnt,Total_cnt
   120    COM /Output/ INTEGER List,Printer(1:5),Device(1:4)
   130    COM /Other/ INTEGER Typewriter,Formal_command,Check_read,Level
   140    ON ERROR CALL Global_error
   150    DISP
```

The ON ERROR statement stops the program for any error from 81 thru 99 (mass storage errors). Error 2 displays an "insufficient memory" comment and sets the variable Err to −2. All other errors cause an error message to be displayed. You may attempt to remedy the problem from the keyboard before pressing [CONT], or you may press [STOP].

CAUTION

DO NOT REMOVE YOUR DATA BASE BEFORE PRESSING [STOP] SO THAT IT MAY BE CLOSED.

The DISP statement clears the "Loading Extension" message that is displayed when Run User Program is begun.

The COMMON variables are explained in later sections of this chapter. Most of the COMMON variables should be preserved. This is described in further detail in the next two sections.

Programming Hint

At first it may be difficult to debug the extensions and match parameters properly. The TRACE PAUSE statement can be useful in debugging extensions.

Note

Some of the important features of QUERY/9000 include the use of softkeys and the use of "fields" to enter information. Use of the INPUT statement is discouraged in favor of the resident screen-input utility routines. Additionally, the subprograms you write should NOT execute an ON KBD or OFF KBD statement.

Using the Search Result

The majority of extension programs only need to access the actual data base records which have met the search criteria and are in the search result. The Get_rec subprogram is intended to handle this function including error recovery. Thus, most extension subprograms contain the following skeleton:

```
SUB Extend
    ALLOCATE Buffer $ [Max_rec_len]
    DOUBLE Ptr
    .
    .
    .
    Ptr = 0
    FOR Record = 1 TO Record_cnt
        CALL Get_rec (#1, Ptr, Buffer)
! Process record here.
    NEXT Record
    .
    .
    .
    SUBEND
```

There may be some extensions which need to access the actual search result scratch file. This file contains the entry numbers of entries which form the search result rather than containing the data itself.

There are several variables in common which pertain to the search result.

Parameter	Explanation
Thread(*)	the first dimension contains the list of sets in the thread. The second dimension contains item or field information and should not be referenced by extension subprograms.
Set_cnt	the number of sets in the current thread
Record_cnt	the number of entries (records) in the current search result.
Select_depth	the number of SELECT FOR's which have been done on the original search result.
Path$(3)	The directory path of the search result scratch file.
Volume$(3)	The volume label of the medium containing the search result scratch file.

The search result scratch file contains one record for each entry of the search result. Each record contains one double integer for each set in the thread. Each double integer contains the entry number of an entry which has met the search criteria.

Consider a simple example of a search on a single set. Assume that the search was to FIND SUBJECT = PHOTOGRAPHY in a library data base.

Set_cnt = 1
Thread(1,1) = 2

Suppose only 3 entries were found.

Record_cnt = 3
Select_depth = 0

Then the search scratch file (which is the file Path$(3)&"/SEARCH<IMAGE>:LABEL" &VOLUME$(3)) would contain four records with the following contents:

Record 1 : −5021
Record 2 : 38517
Record 3 : 38602
Record 4 : 0

The 0 in record four indicates the end of the current search result. If a SELECT FOR were done which was to SELECT_FOR PUBLISH_DATE BEFORE 1 OCTOBER 1905, the variables would be changed to:

Record_cnt = 2
Select_depth = 1

And the contents of the scratch file would become:

Record 1 : 38517
Record 2 : 38602
Record 3 : −5021
Record 4 : 0

The negative record number in record three indicates the end of the search result which met the SELECT FOR criteria.

A second example shows how threads are handled in the search scratch file. Assume the thread is set up from BOOK to CALL_NUMBER to INVENTORY in a library data base.

Set_cnt = 3
Thread (1,1) = 2 Thread (2,1) = 5 Thread (3,1) = 7

After performing the search only four entries were found.

Record_cnt = 4
Select_depth = 0

```
Record 1:   10,   10,    4
Record 2:   10,   10,    5
Record 3:   27,    5,   98
Record 4:   45,   18,   11
Record 5:    0,    0,    0
```

Each record in the search result is now large enough to hold three double integers (12 bytes).

A SELECT FOR would cause only the first number in a record to be set to a negative value.

Using LABELED COMMON Variables

Extensions to QUERY/9000 have all of the following variables available in LABELED COMMON. Variables in Mass_storage, Db_manipulation, Input, Command and Search_result are useful for most extension subprogram applications. The other variables are used internally by QUERY/9000 and are of no use to extension subprograms.

Extensions must preserve the contents of COMMON. It is recommended that the programmer avoid access to blocks of common other than those already mentioned. Further, it is recommended that all variables in Mass_storage, Acc_sets and All_sets in Db_manipulation, and all variables in Search_result be read only and not assigned new values. If changes are made, advanced applications must guarantee consistency of variables before returning to QUERY/9000.

COMMON Variables

Mass Storage LABELED COMMON

Variable	Explanation
Storage$	configured mass storage devices for VOLUME DEVICES ARE.
Path$(*)	mass storage directory paths.
Path$(1)	directory path of the directory containing the QUERY/9000 program files.
Path$(2)	directory path of the directory in which the currently open data base is located.
Path$(3)	directory path in which search and sort scratch files are to be located.
Base$(*)	Data base information
Base$(1)	the name of the open data base.
Base$(2)	the password used to open the current data base.
Volume$(*)	volume labels.
Volume$(1)	the name of the volume containing QUERY/9000 programs.
Volume$(2)	the name of the volume containing the currently opened data base.
Volume$(3)	the volume label of the device used for Search and Sort scratch files.

Loader LABELED COMMON

Variable	Explanation	Range
Mode	the granularity of the loading scheme.	1 to 4
	do not do any LOADSUBS.	1
	do LOADSUBS between every minor subsystem.	2
	do LOADSUBS between major subsystems such as Update, Search, etc.	3
	LOADSUB all of QUERY, DEFINE, or the utilitites into memory.	4
Segment	the currently loaded subsystem.	

Db manipulation LABELED COMMON

Variable	Explanation
Err	a flag indicating a localling detected error.
Sts(*)	status array for data base manipulation statements such as DBOPEN, DBINFO, DBGET, etc.
Acc_sets	number of accessible sets in the data base.
All_sets	total number of sets in the data base.
Db$	the data base string used for data base manipulation statements.
Bf1$	a large primary buffer for DBINFO's.
Bf2$	a small secondary buffer for DBINFO's

Input LABELED COMMON

Variable	Explanation	Range
Key$	the KBD$ buffer used by Kinput and Sinput.	
Key	the numeric value of a key detected by ON KBD mapped to the return value character location of the return string for Kinput or Sinput.	
Keys	controls the extent to which keystrokes are processed.	0 to 3
	throw away all keys except the PAUSE key.	0 or 1
	throw away all keys except the PAUSE key and softkey 8 (which is most often used as the ABORT key).	2
	trap all keys which are used by Kinput or Sinput subprograms.	3

Command LABELED COMMON

Variable	Explanation
Recall$	the previous formal command, which may be recalled.
Command$	the command line being parsed by the Scan subprogram.
Token$	the token found by the Scan subprogram.
Token	the token number found by the Scan subprogram.
Lineptr1	points to the beginning of token in command.
Lineptr2	points to the end of token in command.

Search_syntax LABELED COMMON

Variable	Explanation
Recursion_depth	indicates the number of recursions in syntaxing the search expression.
Subptr1	pointer to the beginning of a subcommand in the boolean program string used when parsing a search expression.
Subptr2	pointer to the end of a subcommand.
Search_item	search item used in optimizing a detail set search.
Optimize	optimization flag used in parsing the search command.
Thread_number	predetermined thread number being searched.

Search_result LABELED COMMON

Variable	Explanation
Set_cnt	the number of sets in the thread.
Thread(*)	the current thread or set being searched.
Thread(1,1)	the set number of the first set in the thread.
Thread(1,2)	the number of sets in the thread.
Thread(2,1)	the set number of the second set in the thread.
Thread(2,2)	the item number of the item connecting the first two sets in the thread.
Select_depth	is the SELECT FOR depth counter.
Record_cnt	the number of entries in the search result.
Record_len	length of Buffer$ needed by Get_rec subprogram to perform DBGETS to concatenate all sets in the thread.
Max_rec_len	the maximum length to which Buffer$ may be allocated.

Update LABELED COMMON

Variable	Explanation
Autoclear	indicates whether Update should do AUTOCLEAR after each Add.
Search	indicates how the search result ws obtained in Update.
Updates	the number of updates since the last backup.

Sort LABELED COMMON

Variable	Explanation
Key_cnt	number of keys to sort
Entry_cnt	the number of entries in the Sort array.
Total_cnt	the length of each entry in the Sort array.

Output LABELED COMMON

Variable	Explanation
List	indicates whether a List should be LINEAR or COLUMNAR.
Printer(*)	information about the current output device.
Printer(1)	device specifier.
Printer(2)	line width.
Printer(3)	page length.
Printer(4)	current line number.
Printer(5)	current page number.
Device(*)	information about the optional external print, if any.
Device(1)	device specifier
Device(2)	line width.
Device(3)	page length.
Device(4)	check read flag.

Other LABELED COMMON

Variable	Explanation
Typewriter	indicates the local typewriter mode to Sinput.
Formal_command	indicates if the formal command mode is in effect.
Check_read	check read flag.
Level	indicates re-entry information to MAIN.

Utility Subprograms Summary

Function	Subprogram	Explanation
Handling fields on the screen:	Kinput	handles softkey inputs
	Sinput	allows a multi-line field to be put on the screen for string editing
	Ninput	allows editing of a numeric field
	Soutput	multi-line field output routine
Softkey handling:	Clear_label	clears softkey labels
	Print_label	labels softkeys
Value handling routines:	Scan	a scanner for finding token in input strings
	Encode_s value	parses string item input values
	Encode_n value	parses numeric item input values
	Decode_n value	formats numeric output values
	Encode_c value	parses input values for code items
	Decode_c value	gets output value for code items
	Name_parse	puts a name string in canonical form
	FN Number_check 1	verifies that a string is a number
	Date_parse	converts a date string to a number
	Time_parse	converts a time string value to a number
Item handling routines:	Get_item	gets information based on an item name
	Dump_items	prints the names of items in current thread
	Dump_codes	prints the code values for a code item
	F_error	displays error messages
Disc volume routine:	Assign	finds a volume, assigns a file
Printer routines:	Printer	sets the OUTPUT TO device
	Lprt	prints to the OUTPUT TO device
Search result access:	Get_rec	reads a record from the current search result

The KINPUT Subprogram

SUB Kinput (Return$,Return)

Sample Call:

```
Return$ = "C@@DEF@GBAA"
CALL Kinput (Return $, Return)
ON RETURN GOTO Tab_forward, Tab_back, k1, k4, k5, k6, k8
```

This subprogram returns a numeric value indicating which softkey or other non-character key was pressed. It is used when no input string is desired, but a branch decision needs to be made. The actual value of the Return variable depends on the input string Return$ which contains the returnable values and it depends on the key which was pressed.

Parameters	Explanation	Range
Return$	A string of one to 21 characters; contains return values as described below.	—
Return	A long real number containing the value corresponding to the key which was pressed.	1 to 26

Common Variable	Explanation
Keys	Temporarily set to 3 to enable trapping of all key strokes.
Key	Set to 0.

Key	Position in the String
Softkeys 1 to 8	Character position 1 to 8
SHIFT TAB	9
TAB	10
RETURN , EXECUTE	11
↑ , SHIFT ↑	12
↓ , SHIFT ↓	13
ROLL ↑ , SHIFT ROLL ↑	14
ROLL ↓ , SHIFT ROLL ↓	15
SHIFT RECALL .	16
RECALL	17
DEL LN	18
INS LN	19
TAB CLEAR	20
TAB SET	21

Note that Special Functions Keys [0] = [8] = [16] = [24] = softkey returning a value of 1. All eight softkeys can be obtained in four different ways.

The value returned is based on the character in the return code string Return$. (The subprogram with commercial-at-signs will fill the string so only non-commercial-at-sign characters need to be included.)

Character in String	Return Value
@	Do not return, beep and wait for another keystroke
A	1
B	2
C	3
etc	

In general, the value returned is NUM ((Character)) – 64 where character is the character in the Return$ string corresponsing to the key which was pressed.

The SINPUT Subprogram

SUB Sinput (INTEGER M, Fr, Fc, Nc, Sc, Tc, H, Return$, Field$,
Return, Status)

Sample Call:

```
Return$ = "A@@BC@@DEFFEF"
Field$ = " "
Return = Status = 0
CALL Sinput (2, 10, 20, 20, 1, 53, 129, Return$, Field$, Return, Status)
IF Status = 1 OR Status = 3 THEN GOSUB New_value
ON Return GOTO k1, k4, k5, k8, Backwards, Forward
```

This subprogram inputs the value of a field located anywhere on the screen. The field is rectangular with each line (except possibly the last) having the same number of characters as the first. The field contents may be modified using the alphanumeric keys. BACKSPACE, HOME, TO END, CLEAR, CLEAR TO END, LEFT ARROW, RIGHT ARROW, INSERT CHARACTER, DELETE CHARACTER, CAPS. The UP ARROW and DOWN ARROW also remain inside the field if possible. Return from the subprogram is forced by the keys described below. The printer must be set to CRT.

Parameters:

Parameter	Explanation	Range
M	Mode	1 to 3
	Display field and return	1
	Display and edit field	2
	Edit displayed field	3
Fr	Row of first (upper left) character	0 to 19
Fc	Column of first (upper left) character	0 to 79
Nc	Number of characters in the first line	1 to 80*
Sc	Starting character of the cursor	1 to Tc
Tc	Total number of characters	1 to 1600*
H	ASCII number of highlight character (Negative values give no echo typing)	−135 to −128 +128 to +135
Return$	A string of one to 21 characters. Each character position contains the value returned by a given keystroke as described below.	—
Field$	The string to be edited.	—
Return	The numeric value as encoded in the character in the character position in Return$ corresponding to the key that was pressed. (See table below.)	1 to 26
Status	Field status code	1 to 4
	Field has changed and is not blank	1
	Field has not changed and is not blank	2
	Field has changed to blank	3
	Field is still blank.	4

* The programmer is responsible to see than the right and lower edges of the field fall inside the screen boundary by controlling the number of characters per line (Nc) and the total number of characters (Tc).

Common Variables:

Variable	Explanation	Range
Keys$	Contains the characters returned by ON KBD from the keyboard buffer.	—
Typewriter	Flag indicating local caps lock handling	0 to 1
	Do not toggle case of input character	0
	Toggle the case of the input character	1

Documented programs which call Sinput:

Ninput

Key	Position in the String
Softkeys 1 to 8	Character position 1 to 8
SHIFT TAB	9
TAB	10
RETURN , EXECUTE	11
↑ , SHIFT ↑	12
↓ , SHIFT ↓	13
ROLL ↑ , SHIFT ROLL ↑	14
ROLL ↓ , SHIFT ROLL ↓	15
SHIFT RECALL	16
RECALL	17
DEL LN	18
INS LN	19
TAB CLEAR	20
TAB SET	21

Note that Special Functions Keys 0 = 8 = 16 = 24 = softkey returning a value of 1. All eight softkeys can be obtained in four different ways.

The value returned is based on the character in the return code string Return$. (The subprogram with commercial-at-sign will fill the string so only non-commercial-at-signs characters need to be included.)

Character in String	Return Value
@	Do not return, beep and wait for another keystroke
A	1
B	2
C	3
•	•
•	•
•	•
Z	26

In general, the value returned is NUM ((Character)) – 64 where character is the character in the Return$ string corresponsing to the key which was pressed.

The NINPUT Subprogram

SUB Ninput (INTEGER M, Fr, Fc, Tc, T, E, H, Return$, Return, REAL Null, Lower, Upper, Value)

Sample Call:

Return$ = "@@@@@@@cdABB"
Value = 0
CALL Ninput (2, 5, 36, 5, 2, 0, 133, Return$, Return, 0, 1, 32767, Value)
ON Return GOTO Up, Down, k7, k8.

This subprogram is used to input strictly numeric values. The incoming value is first compared with the value of Null. If equal, then a blank field is given, otherwise the value in Value is shown. This field is then edited as a string as described for Sinput. When editing is completed, the value is checked to see that it is a valid number, that it is in range, and that the value can be represented exactly in the selected numeric type. (This checking may be disabled on a key-by-key basis by using the lowercase character rather than the upper case.)

Parameters:

Parameter	Explanation	Range
M	Mode	1 to 3
	Display and return	1
	Display and edit	2
	Edit previously displayed number	3
Fr	Row of first character	0 to 19
Fc	Column of first character	0 to 79
Tc	Total number of characters	1 to 24
T	Number type	
	2 = integer	
	3 = double	
	4 = short real	
	5 = long real	
E	unused	0
H	Highlight ASCII number	−135 to −128
	(Negative for no-echo typing)	+128 to +135
Return$	String of 1 to 21 characters containing encoded return values for various keys. See the following tables.	—
Return	Real variable. Returns value corresponding to key which was pressed.	1 to 26
Null	If value equals null on entry, display blank; if blank displayed on exit use this value.	Dependent on numeric type.
Lower	Lower bound used in range checking on exit. Value must not be less than Lower	Dependent on numeric type
Upper	Upper bound used in checking on exit. Value must not be greater than Upper.	Dependent on numeric type
Value	The numeric value being considered.	Dependent on numeric type

Key	Position in the String
Softkeys 1 to 8	Character position 1 to 8
SHIFT TAB	9
TAB	10
RETURN , EXECUTE	11
↑ , SHIFT ↑	12
↓ , SHIFT ↓	13
ROLL ↑ , SHIFT ROLL ↑	14
ROLL ↓ , SHIFT ROLL ↓	15
SHIFT RECALL	16
RECALL	17
DEL LN	18
INS LN	19
TAB CLEAR	20
TAB SET	21

Note that Special Functions Keys [0] = [8] = [16] = [24] = softkey returning a value of 1. All eight softkeys can be obtained in four different ways.

The value returned is based on the character in the return code string Return$. (The subprogram with commercial-at-sign will fill the string so only non-commercial-at-signs characters need to be included.)

Character in String	Return Value
@	Do not return, beep and wait for another keystroke
A	1
B	2
C	3
etc	

In general, the value returned is NUM ((Character)) – 64 where character is the character in the Return$ string corresponsing to the key which was pressed.

The SOUTPUT Subprogram

SUB Soutput (INTEGER M, Fr, Fc, Nc, Tc, H, Field$)

Sample Call:

Field$ = "Data Base"
CALL Soutput (0, 0, 52, 9, 9, 128, Field$)

This subprogram is a short form for the display and return mode of Sinput. The sample call above is equivalent to:

CALL Sinput (1, 0, 52, 9, 1, 9, 128, " ", Field$, 0, 0)

The CLEAR_LABEL Subprogram

SUB Clear_label

Sample Call:

Call Clear_label

This subprogram clears the softkey area of the CRT.

The PRINT LABEL Subprogram

SUB Print_label(A$,B$)

Sample Call:

CALL Print_label("|||||DUMP||EXIT","|||||SCREEN")

screen dump goes here

This subprogram displays the eight softkey labels on the CRT. Each label can have two lines, corresponding to the A$ and B$ parameters. The labels are automatically centered in their fields.

Parameters:

Parameter	Explanation
A$	the top text line for all the labels
B$	the bottom text line for all the labels

Argument Values for the Subprogram Call:

A$ and B$ contain characters for the softkey labels. The maximum number of character per softkey is nine per row (top or bottom). Use vertical bars to separate the individual labels, even when a softkey isn't labeled. A$ and B$ must contain labels and either vertical bars for all the labels up to the right-most label to be displayed; the remaining vertical bars may be omitted.

Argument Values at Subprogram Exit:

A$ and B$ are unchanged.

Documented Subprograms that Call Print_label:

Assign

The SCAN Subprogram

SUB Scan(Cmd)

Sample Calls:

```
Command$ = Field$    ! Assume FNString_input just read something into Field$
CALL Scan(0)         ! get first token
! process token
CALL Scan(1)         ! get second token
! process next token
```

This subprogram scans Command$ for the presence of tokens. The numerical values returned in "Token" are:

Value	Token	Value	Token	Value	Token
0	END-OF-LINE	10	"	20	<ALPHA>
1	,	11	:	21	<ALPHA> >16 bytes
2	.	12	=	30	<NUMBER>
3	;	13	# or <>	31	<NUMBER> >16 bytes
4	+	14	<		
5	−	15	>		
6	*	16	<=		
7	/	17	>=		
8	(	18	$NULL		
9	)	19	INVALID CHARACTER		

"ALPHA" tokens must be 16 or less characters. Examples include "A", "A12", "A14B_12X". Case conversion is not done. Alpha tokens longer than 16 characters are given a token number of 21 but are not returned in Token$. They are delimited by Lineptr1 and Lineptr2.

"NUMBER" tokens are integers with 16 or less characters. Longer numbers are given a token of 31 and not returned in Token$. The number +1.5E6 is broken up into tokens for the "+", "1", ".", "5", "E", and "6" substrings.

Parameters:

Parameter	Explanation
Cmd	zero: reset line pointers and get first token. non-zero: get next token.

Common:

Variable	Explanation
Command$	the command line to be scanned
Lineptr1	pointer into Command$ for beginning of token
Lineptr2	pointer into Command$ for end of token
Token	the token number returned
Token$	the token itself

The ENCODE_SVALUE Subprogram

SUB Encode_svalue (INTEGER Item, Input_value$, Output_value$, INTEGER C,E)

Sample Call:

CALL Encode_svalue (17, Input$, Output$, C,E)
IF E>0 THEN Name parse error

This subprogram takes an input string for a given item number and makes any necessary checks or adjustments before returning it in canonical form. In particular, if the item is a name, then the string is parsed as a name and put into canonical form, or errors preventing such are returned.

Parameters:

Parameter	Explanation	Range
Item	Item number of the string item	1 to 1023
Input_value$	Input value string	—
Output_value$	Output value in canonical form	

Common:

Common Variable	Explanation	Range
Command$	String variable used in name parsing	
Err	Variable which flags error during name parse	0 – no error non-0 – error.

The ENCODE_NVALUE Subprogram

Sub Encode_nvalue (INTEGER Item, Type, DOUBLE Format, Value$, REAL Value, INTEGER C,E)

Sample Calls:

CALL Encode_nvalue (0, 3, 0, "57.2", Value, C,E)
CALL Encode_nvalue (37, 5, −536870921, "1 Jan 1984", Value, C,E)

This subprogram takes an input string and parses it according to the type and format of the item for which it is a value. There are two modes which can be used in calling this subprogram. Either the item number and item type must be non-zero, or the item type and format. In the first case, the item format and range are taken from the data base definition for parsing and range checking. In the second case, the format is used as entered, and the range is the default for the numeric type. If the input string is blank, the null value is used. Otherwise, the input string is parsed according to the format of this item. The range of the resultant value is checked. If the string is not a valid number, the number is out of range, or the value cannot be represented exactly in the numeric type, an error is returned.

Parameter	Explanation	Range
Item	Item number of current item*	0 to 1023
	Use default values for item type	0
	Get format and range from root file	1 to 1023
Type	item type	2 to 5
	integer	2
	double	3
	short real	4
	long real	5
Format	Format extension to item type	−2147483647 to +2147483647
	QUERY interpreted	<0
	USER interpreted	>=0
Value$	Input string	—
Value	Output value	—
C	Current character position	1 to LEN (Value$)
E	Error flag	0 to 4
	no error	0
	Illegal number	1
	Out of range	2
	Cannot represent exactly	3
	Illegal data, time, timedate	4

* The programmer must verify that this is a numeric item.

The DECODE_NVALUE Subprogram

SUB Decode_nvalue (INTEGER Item, TYPE, DOUBLE Format, REAL Value, Value$)

Sample Call:

```
Value$ = "1 January     1984"
CALL Encode_nvalue (15, 5, 0, Value$, Value, C,E)
IF E = 0 THEN
   CALL Decode_nvalue (15, 5, 0, Value, Value$)
END IF
```

This subprogram is the complement of Encode_nvalue. It takes a numeric value and expresses it as a string in canonical form, if possible. As with Encode_nvalue there are two modes to call it. First an item number and item type may be used. The range and format are taken from the root file. If the item number is zero, the format used is that which is passed in; the range values are the default for the item type. If the item value is out of range and equal to the null value, then the value returned is blank. Otherwise, the value is expressed in the prescribed format. If this is not possible, the return string is asterisk filled.

Parameter	Explanation	Range
Item	Current item number*	0 – number of items in DB
	Use default range, passed in format	0
	Use format and ranges from root file	1 – number of items
Type	Type of current item	2 to 5
	integer	2
	double	3
	short real	4
	long real	5
Format	Format extension of item type (used if item number = 0)	–2147483647 to +2147483647
	QUERY interpreted	<0
	USER interpreted	>=0
Value	Input numeric value	—
Value$	Output string value	—

* The programmer must verify that this is a numeric item.

ENCODE_CVALUE

```
SUB Encode_cvalue (INTEGER Item, Value$, INTEGER, Value, C,E)

Value$ = "Code 1"
CALL Encode_cvalue (32, Value$, Code_value, C,E)
IF E = 0 THEN PACKUSING Code fmt; Buffer$ [1]
```

This subprogram takes the input string and attempts to convert it into a numeric value based on the item number. Errors are reported.

Parameter	Explanation	Range
Item	Item number of code item*	1 to number of items
Value$	Input string containing code value	—
Value	Output number containing numeric code value.	0 to 32
C	Character position	1
E	Error reporting parameter	0 to 1
	No error	0
	Not a code value	1

* The programmer must verify that this is a code item.

DECODE_CVALUE

```
SUB Decode_cvalue (INTEGER Item, Value, Value$)

For I = 0 TO 32
   CALL Decode_cvalue (41, I, Code$)
   .
   .
   .
NEXT I
```

This routine pulls the string code value corresponding to a numeric code value from the root file. If the code value is zero, a blank is returned. If an invalid code value is passed in, an asterisk-filled string is returned.

Parameter	Explanation	Range
Item	Code item number*	1 to number of items
Value	Input numeric value of code.	0 to 32
Value$	Output string value of code	—

* The programmer must verify that this is a code item.

The NAME_PARSE Subprogram

SUB Name_parse

Sample Call:

```
Command$ = "Sam Smith"
CALL Name_parse    ! When the program returns, Command$ contains "Smith, Sam"
```

The NAME_PARSE subprogram attempts to convert the string in Command$ to the format:

LAST,FIRST [MIDDLE] [,SUFFIX]

If the string is already in this canonical form, no changes are made. The string must contain at least two words for it to be a valid name. If this is not the case and the conversion cannot be made, the subprogram returns with Err set to 18.

Common Variable	Explanation
Command$	Working string used in transforming the name into canonical form.
Err	Used to report any errors.

The NUMBER_CHECK1 Function

DEF FN Number_check1 (N$,V,INTEGER C)

Sample Call:

N$ = "12345"
ON FN Number_check (N$,V,C) GOTO Not_number

This function parses a character string to verify that it is a valid number. A zero is returned for the function if the number is valid, otherwise a message printed and a 1 is returned.

Parameters:

Parameter	Explanation
N$	the string to be tested to see if is a number
V	the numeric value of the number
C	The position of the first invalid character found

Argument Values for the Subprogram Call:

Parameter	Explanation
N$	string variable, character string that may contain a number

Argument Values at Subprogram Exit:

Parameter	Explanation
N$	unchanged
V	value of the number in the string, if any
C	C>1: position of the first invalid character found, if any C = 1: VAL(N$) failed

Possible Values for the Function:

0: success
1: failure

Error Conditions Checked:

String contains an invalid number (wrong character).

The DATE_PARSE Subprogram

SUB Date_parse(Cmd,REAL Date)

Sample Call:

```
Command$ = "1 JAN 1984"
CALL Date_parse(0,Date)    ! If successful, numerically encoded dates are stored in
                           ! the variable Date
```

This subprogram parses the string in Command$ to determine if it is an accepted date format. If so, it returns a numeric value for the date. If the string cannot be parsed correctly, Date_parse returns to its caller with Err set to 1. The HP 9000 BASIC TIMEDATE convention is used.

Parameters:

Parameters	Explanation
Cmd	Non-zero indicates continuation of parsing Command$. The token delimited by the variables Lineptr1 and Lineptr2 is examined first. Zero indicates that Command$ should be parsed from the beginning to the end of the line.
Date	Long real variable returning the encoded date.

Common Variable	Explanation
Command	String variable used in parsing.
Err	Error flag. Zero if no error, one if error.

The TIME_PARSE Subprogram

SUB Time_parse (Cmd,REAL Time)

Sample Call:

```
Command$ = "TIME IS 23:59:23"
CALL Scan(0)
! Process the word "TIME"
  CALL Scan(1)
!Process the word "IS"
 CALL Time_parse (1,Time)
```

This subprogram parses some or all of the string in Command$ to determine if it is a valid time value. Like Data_parse, it returns the HP 9000 BASIC TIMEDATE compatible value if the parse is successful. If unsuccessful, a value of 1 is returned in Err.

Parameter	Explanation
Cmd	0 if Command$ should contain only a complete true value, 1 if the next tokens in Command$ should be a time value.
Time	Long real return variable for time value

Common Variable	Explanation
Command$	String variable containing the value to be parsed.
Err	Error flag. Zero if no error, one if error.

The GET_ITEM Subprogram

SUB Get_item (INTEGER Set, Item, Itype, Ftype Tindex DOUBLE Ilen, Ioff, Ifor, Idim(*), Isub(*)

Sample Call:

```
DOUBLE Idim (0:6,1:2), Isub (1:6)
Command$ = Field$"Set.Item (1, 2, 3, 4, 5, 6)"
CALL Scan(0)
CALL Get_item (Set, Item, Itype, Etype, Tindex, Ilen, Ioff, Ifor, Idim(*), Isub(*) )
```

This subprogram parses Command$ for an item name in the format

[SET.] ITEM [(subscripts)]

If a thread was not set up on entry, a check is made to see if the item exists in a single set or if it is in several sets, a unique manual master or detail set. If so, the thread is set up to that set. Otherwise a "non-unique" error comment is generated. If the thread is set up on entry, the item must occur in one of the sets of the thread. If no set name is explicitly present, the left-most set in the thread in which the item occurs is used. On entry, Token$ must be set up to the first token to be used in the parse. On exit, Token$ points to the next token past the item name.

Parameter	Explanation	Range
Set	The set number in which the item occurs.	—
Item	The item number.	—
Itype	The item type from 1 to 6	
	STRING	1
	INTEGER	2
	DOUBLE	3
	SHORT	4
	LONG	5
	CODE	6
Ftype	The item type extended by the item format.	1
	STRING	2
	INTEGER	3
	DOUBLE	4
	SHORT	5
	LONG	6
	CODE	7
	NAME	8
	DATE	9
	TIME	10
	CLOCK	
Tindex	How far into the thread this set is.	
Ilen	The item length in bytes.	
Ioff	The offset from the beginning of a set. (accounts for subitem index)	
Ifor	Format value for this item.	
Idim(*)	array dimension for this item	
Idim(0,1)	Number of dimensions.	
Idim(0,2)	Total number of elements.	
Idim(I,1)	Lower bound for this dimension	
Idim(I,2)	Number of elements for this dimension.	
Isub(*)	Array subscript specified.	
Isub(I)	Subscript of this dimension.	

Errors:

Error	Explanation
3	Non-existent item or item not in set.
4	Subitem index required.
5	Subitem index out of range or invalid.
11	Parenthesis expected.
14	Invalid set name or set not in thread.
16	Non-unique item. Set must be specified.

The DUMP_ITEMS Subprogram

SUB Dump_items

Sample Call:

CALL Dump_items

This subprogram prints item names in current thread using the internal printer.

The DUMP_CODES Subprogram

Sub Dump_codes (INTEGER Item)

Sample Call:

```
DBINFO (B$, Item, 102, 5(*), S$)
If S$[17;1]=C THEN CALL Dump_codes (Item)
```

The DUMP_CODES subprogram prints the codes for an item using the internal printer. Item is the number of the item. It is the programmer's responsibility to see that the item is of type code.

The F_ERROR Subprogram

SUB F_error(INTEGER I)

Sample Call:

CALL F_error(Err)

This subprogram displays one of the error comments below:

Error	Explanation
1	Invalid character.
2	Improper expression.
3	Non-existent item or item not in set.
4	Subscript required.
5	Subscript out of range or invalid.
6	Invalid number.
7	Type incompatibility.
8	Invalid date.
9	Improper string.
10	Invalid code value.
11	Parenthesis expected.
12	Comma expected.
13	String too long.
14	Invalid set name or set not in thread.
15	Value entered is not within allowable range.
16	Item is not unique; a set must be specified.
17	Maximum number of items exceeded.
18	Invalid NAME format.
19	An A or D or blank must be entered.
20	There is nothing to sort.
21	Invalid thread name.
22	Command ignored; search result is empty.
23	Quotes must enclose entire string due to reserved keyword.
25	There is nothing to restore.
26	Width of items to be listed exceeds width of printer.
28	End of line expected.
29	No more than 10 nested parentheses are allowed.
30	The maximum number of multiple values or OR's has been exceeded.
31	An item name must occur on the left-hand side of the relational operator.
32	Ambiguous request: FIND ALL ENTRIES requires a blank search expression.
33	Search result work file cannot be accessed as currently configured.

The ASSIGN Subprogram

SUB Assign (V$, F$, A, H,:#1, Return)

Sample Call:

```
V$ = "QUERY"
F$ = "Directory1/QUERY/QLEX"
CALL Assign (V$, F$, 3, 0, #5, Return)
IF Return >1 THEN Did not work
```

The ASSIGN subprogram locates the mass storage device in which the disc labeled "V$" (volume name) is mounted. If found, it examines the variable "A" to determine if it should assign a file or just return the select code of the mass storage device. If more than one disc is mounted with the same volume name, neither can be accessed by volume label. If the volume isn't mounted, the program requests that the user mount the volume or abort the request.

Parameters:

Parameter	Explanation	Range
V$	Volume label	—
F$	Directory path and file name of the file to be assigned (if any)	—
A	Assignable flag	1 to 3
	Verify that volume V$ is mounted	1
	Verify that volume and file are present by doing assign. (Ignore errors since file is unassignable)	2
	Assign file	3
#1	File specifier	#1 to #10
Return	Return variable which reports the success of the operation.	
	Task complete	1
	Found file but could not assign it (but should have since A = 3)	2
	User aborts request	3

Common:

Common Variable	Explanation
Storage$	VOLUME DEVICES ARE string; string of all devices expected to be accessed by label.

The PRINTER Subprogram

SUB Printer(I)

Sample Call:

CALL Printer(0)

This subprogram sets the current "OUTPUT TO" device. The passed parameter should be 0 for the CRT, 1 for the thermal printer, and 2 for the configured external printer. This subprogram sets the values for the Printer(*) array in COMMON. The Lprt subprogram then uses this array to determine where to send the output.

Parameters:

Parameter	Explanation	Range
I	Target printer	0 to 2
	CRT	0
	Thermal printer	1
	External printer (if configured)	2

Common:

Common Variable	Explanation	Range
Printer(1)	Device specifier of current printer	(see device specifier)
Printer(2)	Page width	—
Printer(3)	Page length	—
Printer(4)	Current line	1 to page length
Printer(5)	Current page	—
Device(1)	External printer device specifier (if any)	0 or device specification
Device(2)	Page width of external part	—
Device(3)	Page length of external part	—

The LPRT Subprogram

SUB Lprt(Buffer$,Cctl)

Sample Call:

```
DIM Buffer$[100]
CALL Printer(0)    ! Set CRT as OUTPUT device
CALL Lprt (" ",0)    ! initializing the printer
FOR I=1 TO 100
  Buffer$="This is line number" & VAL$(I)&"."
  CALL Lprt (Buffer$,3) ! Print each line
  IF Err THEN Abort
NEXT I
Abort: CALL Lprt(" ",2) ! Print the last page number, do a page eject and shut off
                        ! the printer
```

This subprogram prints the Buffer$ to the current OUTPUT TO device. A blank line followed by a line with the page number is printed at the bottom of each page. For example, if the page length is 65 lines, 63 text lines are printed. At each invocation, PRINTER IS is set to the current output device specified in COMMON.

If the current output device is the CRT, the Lprt subprogram automatically pauses between pages enabling you to list the next page, dump the screen, or stop the subprogram. You can abort the subprogram at any time while printing by pressing the ABORT softkey which causes Err to be set to non-zero.

While using the Lprt subprogram, each line printed must be a string and therefore all numbers must be converted using VAL$.

Parameters:

Parameter	Explanation
Buffer$	output line buffer
Cctl	Carriage Control: 0 = RESET (Page #1, PRINTER IS, SOFTKEYS, Set KEYSW to −1.) 1 = PAGE EJECT 2 = EXIT (Page eject. Set KEYSW to −2) 3 = If necessary, do a page eject. Then print the line in LBF$.

Common:

Printer(1) = HP-IB address
Printer(2) = Page width
Printer(3) = Page length
Printer(4) = Current line number
Printer(5) = Current page number
Err = ABORT flag.

The GET_REC Subprogram

SUB Get_rec(#1,DOUBLE Ptr,Buffer$)

Sample Call:

```
DOUBLE Ptr
Ptr = 0
FOR I = 1 TO Record_cnt
  CALL Get_rec(#1,Ptr,Buffer$)
  ! Process the record
NEXT I
```

The GET_REC subprogram reads a record from the SEARCH RESULT into Buffer$. Buffer$ contains the concatenated records of all sets in the thread that are positive. For example, if the thread contains +1, +3, −5, +6, then records from sets 1, 3, and 6 are concatenated. The first call to Get_rec should have Ptr = 0. Ptr is incremented on return. GLOBAL_ERROR should be in effect. Since the Assign subprogram is called, the softkeys may be set to ABORT, the DISP line may be blanked, the CRT may be left intact or blanked, and PRINTER IS may be reset to CRT. On exit, Err is set to 1 if any error or abort occurs.

Parameters:

Parameters	Explanation
#1	for search scratch file assignment.
Ptr	pointer into the search result.
Buffer$	to return result.

```
   10 ! ************************************************************************
   20 !
   30 !  (C) Copyright Hewlett-Packard Company, 1982.  All rights are reserved
   40 !  Copying or other reproduction of this program except for archival
   50 !  purposes is prohibited without the prior written consent of
   60 !  Hewlett-Packard Company.
   70 !
   80 !                       RESTRICTED RIGHTS LEGEND
   90 !
  100 !  Use, duplication, or disclosure by the Government is subject to
  110 !  restrictions as set forth in paragraph (b)(3)(B) of the Rights to
  120 !  Technical Data and Computer Software clause in DAR 7-104.9(a).
  130 !
  140 !  HEWLETT-PACKARD COMPANY
  150 !  Engineering Systems Division
  160 !  3404 East Harmony Road
  170 !  Fort Collins, Colorado 80525
  180 !
  190 ! ************************************************************************
  200 !
  210 !        SUM:  a sample QUERY/9000 extension which is executed by the
  220 !              RUN USER PROGRAM feature.
  230 !
  240 !
  250 !        This subprogram will request the name of an item and then
  260 !        read all entries in the search result, adding together the
  270 !        requested field.  It will then display the number of
  280 !        non-null entries, and the sum, mean, and standard deviation.
  290 !
  300 !        Sample use:  The user executes a FIND to get all components
  310 !                     that occur on PC board "ABC".  This extension
  320 !                     then adds up the prices of all components.
  330 !
  340 !        ON ENTRY:  The DISP line states "==> Loading Extension."
  350 !                   SUM  should clear the DISP line and set up the
  360 !                   ON ERROR trap.
  370 !
  380 ! ************************************************************************
  390 !
  400 !        Major Variables:
  410 !
  420 !          Record_count    - number of entries in search result
  430 !          Field$          - the item name entered by the user
  440 !          Lineptr1        - the cursor position within Field$
  450 !          Set             - the SET # the item occurs in
  460 !          Item            - the ITEM #
  470 !          Itype           - the item type (a number from 1 to 7)
  480 !          Ilen            - the item length
  490 !          Ioff            - the zero-based item offset within the set
  500 !          Total           - the calculated sum
  510 !          Sumsq           - the calculated sum of squares
  520 !          Counter         - the number of non-null entries read
  530 !          Ptr             - Counter for number of calls to GET_REC
  540 !          Keysw           - set to -1 to indicate ABORT only
  550 !
  560 ! ************************************************************************
  570 !
  580  SUB Extend
  590    OPTION BASE 1
  600    COM /Mass_storage/ Storage$[150],Path$(1:3)[226],Base$(1:2)[16],Volume
$(1:3)[16]
  610    COM /Loader/ INTEGER Mode,Segment
  620    COM /Db_manipulation/ DOUBLE Err,Sts(1:10),INTEGER Acc_sets,All_sets,D
b$[284],Bf1$[2050],Bf2$[196]
```

```
   630     COM /Input/ Keys$[160],INTEGER Key,Keys
   640     COM /Command/ Recall$[321],Command$[321],Token$,INTEGER Token,Lineptr1
,Lineptr2
   650     COM /Search_syntax/ INTEGER Recursion_depth,Subptr1,Subptr2,Search_ite
m,Optimize,Thread_number
   660     COM /Search_result/ INTEGER Set_cnt,Thread(1:10,1:2),DOUBLE Select_dep
th,Record_cnt,Record_len,Max_rec_len
   670     COM /Update/ INTEGER Autoclear,Search,Updates
   680     COM /Sort/ INTEGER Key_cnt,Entry_cnt,Total_cnt
   690     COM /Output/ INTEGER List,Printer(1:5),Device(1:4)
   700     COM /Other/ INTEGER Typewriter,Formal_command,Check_read,Level
   710 !
   720     DIM Field$[36]
   730     INTEGER Integer,Set,Item,Itype,Ftype,Tindex,I
   740     DOUBLE Double,Ilen,Ioff,Ifor,Idim(0:6,1:2),Isub(6),Ptr
   750     SHORT Short
   760 !
   770 Intfmt:PACKFMT Integer
   780 Dblfmt:PACKFMT Double
   790 Shtfmt:PACKFMT Short
   800 Lngfmt:PACKFMT Long
   810 !
   820 !
   830 !         Initialize a few parameters and see if there is a search result.
   840 !
   850     ON ERROR CALL Global_error
   860     ALLOCATE Buffer$[Record_len]
   870     IF Record_cnt THEN Start
   880     DISP "==> There is no search result;  SUM extension aborted."
   890     BEEP
   900     SUBEXIT
   910 !
   920 !         Put up the screen display and softkeys.
   930 !
   940 Start:Field$=""
   950     PRINT "";SPA(33);"SUM ENTRIES";LIN(2)
   960     PRINT "The SUM ENTRIES extension to QUERY/9000 will read all entries i
n the current"
   970     PRINT "search result and calculate the sum, mean and standard deviatio
n for the item"
   980     PRINT "you specify below.  Null entries will be ignored.";LIN(2)
   990     PRINT "       Enter the item you want summed: "
  1000     CALL Print_label("||DUMP|||||EXIT","||ITEMS")
  1010     Lineptr1=1
  1020 !
  1030 !         Read and process the item name.
  1040 !
  1050 Read:Err=0
  1060     CALL Sinput(2,8,40,36,Lineptr1,36,129,"@@A@@@@B@CC",Field$,Return,Stat
us)
  1070     ON Return GOTO Di,Exit,Check
  1080 !
  1090 !         DUMP ITEMS was pressed.
  1100 !
  1110 Di:CALL Dump_items
  1120     GOTO Read
  1130 !
  1140 !         The item name was entered.  Check to see if it is a valid item n
ame.
  1150 !
  1160 Check:Command$=Field$=TRIM$(Field$)
  1170     CALL Scan(0)
  1180     CALL Get_item(Set,Item,Itype,Ftype,Tindex,Ilen,Ioff,Ifor,Idim(*),Isub(
*))
  1190     IF NOT Err AND Token THEN Err=28
  1200     IF NOT Err THEN Numcheck
```

```
1210    CALL F_error((Err))
1220    GOTO Read .
1230 !
1240 !        We have a valid item name.  Make sure it is numeric item.
1250 !
1260 Numcheck:IF (Ftype>1) AND (Ftype<6) THEN Sum
1270    DISP "==> Item type is not numeric."
1280    BEEP
1290    Lineptr1=1
1300    GOTO Read
1310 !
1320 !        Item type is numeric.  Set the thread negative except for only
1330 !        that set that the item occurs in (GET_REC will read only from
1340 !        positive set numbers).  Read the entries and calculate the sum
1350 !        and average.  Also, get the null value for this item.
1360 !
1370 Sum:Total=Counter=Ptr=Sumsq=0
1380    Ioff=Ioff+1
1390    FOR I=1 TO Set_cnt
1400      Thread(I,1)=-Thread(I,1)
1410    NEXT I
1420    Thread(Tindex,1)=ABS(Thread(Tindex,1))
1430    Keysw=2
1440    Key=0
1450    DBINFO (Db$,Item,109,Sts(*),Bf1$)
1460    ON Itype-1 GOSUB Ni,Nd,Ns,Nl
1470    CALL Print_label("|||||||ABORT","")
1480    GOTO Sum1
1490 !
1500 Ni:UNPACK USING Intfmt;Bf1$[5]
1510    Null=Integer
1520    RETURN
1530 Nd:UNPACK USING Dblfmt;Bf1$[5]
1540    Null=Double
1550    RETURN
1560 Ns:UNPACK USING Shtfmt;Bf1$[5]
1570    Null=Short
1580    RETURN
1590 Nl:UNPACK USING Lngfmt;Bf1$[5]
1600    Null=Long
1610    RETURN
1620 !
1630 Sum1:IF Key OR Ptr=Record_cnt THEN Done
1640    DISP "Reading entry #";VAL$(Ptr+1);".  Cumulative sum: ";VAL$(Total)&"
."
1650    CALL Get_rec(#1,Ptr,Buffer$)
1660    IF Err THEN Done
1670    ON Ftype-1 GOSUB Int,Dbl,Sht,Lng
1680    IF Value=Null THEN Sum1
1690    Total=Total+Value
1700    Sumsq=Sumsq+Value*Value
1710    Counter=Counter+1
1720    GOTO Sum1
1730 !
1740 Int:UNPACK USING Intfmt;Buffer$[Ioff]
1750     Value=Integer
1760     RETURN
1770 !
1780 Dbl:UNPACK USING Dblfmt;Buffer$[Ioff]
1790     Value=Double
1800     RETURN
1810 !
1820 Sht:UNPACK USING Shtfmt;Buffer$[Ioff]
1830     Value=Short
1840     RETURN
1850 !
```

```
  1860 Lng:UNPACK USING Lngfmt;Buffer$[Ioff]
  1870     Value=Long
  1880     RETURN
  1890 !
  1900 !         The sum is complete.  Display it and fix the thread.
  1910 !
  1920 Done:DISP
  1930    PRINT PAGE;
  1940 Stats:PRINT "Statistics for "&TRIM$(Field$)&":";LIN(1)
  1950    PRINT SPA(6);"Number of non-null entries:";Counter
  1960    IF NOT Counter THEN Barf
  1970    PRINT SPA(6);"Sum:";Total
  1980    PRINT SPA(6);"Mean:";Total/Counter
  1990    PRINT SPA(6);"Standard Deviation:";SQR((Counter*Sumsq-Total*Total)/(Co
unter*(Counter-1)))
  2000 Barf:PRINT LIN(3)
  2010    FOR I=1 TO Set_cnt
  2020      Thread(I,1)=ABS(Thread(I,1))
  2030    NEXT I
  2040    CALL Print_label("NEW|||||DUMP||EXIT","SUM|||||SCREEN")
  2050    CALL Kinput("A@@@@B@C",Return)
  2060    ON Return GOTO Start,Dump,Exit
  2070 !
  2080 Dump:DUMP ALPHA
  2090    GOTO Start
  2100 !
  2110 !         The EXIT softkey was pressed.
  2120 !
  2130 Exit:Err=0
  2140  SUBEND
```

Chapter 10
Using the Formal Command Language

When you are familiar with various features of QUERY/9000, you can save time by using the formal command language rather than the softkeys. To use the formal commands, press the FORMAL COMMAND softkey at the beginning of QUERY/9000. When you have accessed the formal command, press [RETURN] to perform the function.

Another feature that can save you time is the RECALL softkey. This softkey is used to re-display any command that has been previously entered, enabling you to view, modify and reuse previous commands. The commands are stored into a 321-character buffer on a last in, first out basis. Each time the RECALL softkey is pressed, a previous command is displayed in the command area on the CRT.

When the recall buffer becomes full, each new command causes one or more of the oldest commands, depending on size, to be lost.

Also, formal commands can be SAVEd in a BASIC Data file and executed automatically by means of the EXECUTE command.

This chapter lists the formal command syntax for each of the subsystems.

Syntax Guidelines

The following guidelines are used in these formal command descriptions.

[ ]	All items enclosed in brackets are optional.
|	A vertical line between two parameters means "or"; only one of the parameters can be included.
DOT MATRIX	All items in dox matrix must appear exactly as shown.
...	Three dots indicate that the previous item can be repeated.

MAIN QUERY SYSTEM

```
*OUTPUT TO     CRT | THERMAL | PRINTER
*SHOW    BASES | STRUCTURE | SCHEMA | SETS | THREADS
*STOP
 DEFINE    FORM | THREAD
 RUN file name
*EXECUTE file
*CLOSE
```

UPDATE SUBSYSTEM

```
ADD IN set name [USING FORM]
   *[PAUSE] DELETE IN set FOR search expression
    MODIFY IN set [USING FORM ] FOR search expression
   *[PAUSE] REPLACE IN set BY item = value FOR search expression
   *[PAUSE] REPLICATE IN set BY item = value FOR search expression
```

SEARCH SUBSYSTEM

```
BROWSE set
   *FIND [{set | thread} FOR] search expression
   *SELECT FOR search expression
   *RESTORE
   *[LINEAR] LIST ["title"] item, item, item
   *SORT BY item [A | D], item [A | D], ...
```

The EXECUTE formal command expects a data file (such as produced by a SAVE from the BASIC editor), each line of which is a formal command. The EXECUTE reads each line from the data file and performs to requested operation. Those statements starred above execute without any user interaction, (unless the PAUSE keyword is included in update statements).

For example:

```
! OUTPUT TO PRINTER
! FIND BOOK FOR SUBJECT WITH COMPUTER
! SORT BY AUTHOR,TITLE
! LIST AUTHOR,TITLE
```

This searches the open LIBRARY data base for all books with COMPUTER as part of the subject. These are sorted and listed to the external printer.

Chapter 11
Utilities

Introduction

The QUERY/9000 system is designed to maintain the integrity of IMAGE/9000 data bases. There is the possibility, however, of losing data or structural information due to hardware failure, user error, operating system error, or some external cause such as a power failure.

The "Data Base Maintenance" chapter of the IMAGE/Data Base Programming Techniques manual covers data base maintenance procedures. In addition, six utility programs are provided by QUERY/9000 to help you maintain the integrity of your data base.

- BACKUP – creates a backup of your data base.
- RECOVER – restores the data base using the backup.
- UNLOAD – creates an unload file which contains all or part of the data in the data base.
- LOAD – puts the unloaded data back into a data base.
- ERASE – erases all data entries and related path information from all sets in a data base.
- PURGE – purges the specified data base.

You can access these utilities through the initial QUERY/9000 softkeys. First load QUERY/9000 as explained in the Chapter 1 section entitled "Running QUERY/9000", then press [RUN]. When the initial QUERY menu is displayed, you need not open the data base by supplying the information that is requested. Simply press the DATA BASE UTILITIES softkey to begin.

Note

If you begin to use one of the utilities, then decide you don't want to continue, the ABORT and UTILITY OPTIONS softkeys enable you to stop. ABORT actually terminates the particular utility operation, and UTILITY OPTIONS enables you to exit the particular utility module.

Backup and Recovery

When data or structural information is lost, the data base is **corrupt**. Therefore, it is essential to adopt regular backup procedures in anticipation of possible trouble. A backup version can be used to return the data base (via the RECOVER utility) to the state it was in when the backup was made.

The BACKUP utility creates several backup files. These files can either be on a different mass storage medium than the data base, or on the same one. It is best, however, to place the backup files on a separate medium that is not needed for normal operations to ensure security of the backup version.

The BACKUP utility also enables you to create a "working" copy of your data base. It ensures that all of the necessary files are copied, and it maintains the same file names, directory structure, and protect codes as the source data base.

Note

Your data base must be **closed** in order to perform a backup or recover.

Frequency of Backup

It is recommended that you back up your data at regular intervals. The frequency is dependent on the number of changes made and how critical the changes are. The DBOPEN statement returns the number of changes made since the last complete backup in element 8 of the status array.

Data Base Validation

You should inspect your data base periodically to insure that it is not corrupt. It is especially important to do this prior to data base backup. A backup of a corrupt data base is worthless. Two methods for verifying that your data base is not corrupt are described in the "Data Base Maintenance" chapter of the IMAGE/Data Base Programming Techniques manual.

Using the BACKUP Utility

Before attempting to use BACKUP, note that a destination directory must exist. Its path name should include all directories up to, but not including, the data base name. BACKUP creates a new directory with the same name as the data base, and copies the root file and all of the data sets into the new directory. (If you back up to the root directory, then you don't need to specify a destination directory.)

Note

You may want to use the PURGE utility to clear the destination directory of an outdated backup.

Enter the UTILITIES subsystem by pressing the DATA BASE UTILITIES softkey from the initial QUERY/9000 menu. To run the BACKUP utility, press the softkey labeled BACKUP DATA BASE.

The program then asks for information about the data base which you want to back up. Use RETURN to move the cursor after you have entered information for each category.

- Data base name
- Root file volume
- Root file path
- Data base maintenance word

Then answer these three questions with Y or N:

- `Do you want a new destination device for each new Data Base volume?` Answer yes (Y) if you want to make a multi-volume backup of a multi-volume data base; you are prompted during the backup to mount the volumes and enter each volume label. (You can remove the root file volume after its files have been copied.)
- `Do you want to back up any form files?` It may not be necessary to back up form files each time, since they rarely change. Form files should be backed up at least once and not purged. The backup takes longer if you answer yes (Y).
- `Do you want to back up any thread file?` It may not be necessary to back up the thread file each time, since it does not change very often. The thread file should be backed up at least once and not purged. The backup takes longer if you answer yes (Y).

When you have entered all the information requested and have answered the questions, press the PROCESS BASE INFO softkey to continue.

The program then asks for information about the destination (backup). Use RETURN to move the cursor after you have entered information for each category.

- Destination volume
- Destination path

When you have provided this information, press the EXECUTE BACKUP softkey to run the utility.

After the data base has been successfully copied, the message `BACKUP COMPLETE` is displayed, and the program returns to the original menu of the UTILITY subsystem.

Special Considerations

If BACKUP runs out of space on the destination, it requests a new destination volume, then continues.

If the data base directory or forms directory already exists on the destination device, you are asked if that directory should be used or if you want to specify another directory.

Note that if the directory already exists, it cannot contain file names identical to those in the data base to be copied. If you attempt to copy to such a directory, this message is displayed:

```
*** DUPLICATE FILE FOUND, BACKUP ABORTED
```

and the program returns to the original menu of the UTILITY subsystem.

Note

If any error occurs that causes BACKUP to abort, the backup is invalid and should not be used. The only exception to this is error 83. If you back up a write-protected disc, the message "Error 83 – Backup Aborted" is displayed, even though BACKUP has completed successfully and the backup is valid.

After completing successfully, BACKUP updates the modification count of the source data base. It also updates the create date in the root file of the data base.

Using the RECOVER Utility

RECOVER copies all of the data base files from the source device (backup) to the destination data base. It creates the data base directory if necessary, and prompts for any missing volumes.

RECOVER does not purge any files. You must purge all of the data sets and the root file from the the destination data base before the RECOVER is executed. The form and thread files must also be purged if they are to be recovered.

Enter the UTILITIES subsystem by pressing the DATA BASE UTILITIES softkey from the initial QUERY/9000 menu. To run the RECOVER utility, press the softkey labeled RECOVER DATA BASE.

The program then asks for information about the destination data base. Use RETURN to move the cursor after you have entered information for each category.

- Data base name
- Root file volume
- Root file path
- Data base maintenance word

Then answer these three questions with Y or N:

- `Do you want a new destination device for each new data base volume?` It does not matter if you answer yes or no to this question. The RECOVER utility recovers all volumes of a multi-volume backup.
- `Do you want to recover any form files?` It may not be necessary to recover form files each time, since they rarely change. The recovery takes longer if you answer yes (Y).
- `Do you want to recover any thread files?` It may not be necessary to recover the thread file each time, since it does not change very often. The recovery takes longer if you answer yes (Y).

When you have entered all the information requested and have answered the questions, press the PROCESS BASE INFO softkey to continue.

The program then asks for information about the source data base (backup). Use RETURN to move the cursor after you have entered information for each category.

- Source volume
- Source path

When you have provided this information, press the EXECUTE RECOVER softkey to run the utility.

After the data base has been successfully recovered, the message `RECOVER COMPLETE` is displayed, and the program returns to the original menu of the UTILITY subsystem.

Special Considerations
If the data base or forms directory you specify already exists, you are asked if that directory should be used or if you want to specify another directory.

Note that if the directory already exists, it cannot contain file names identical to those in the data base to be copied. If you attempt to copy to such a directory, this message is displayed:

```
*** DUPLICATE FILE FOUND, RECOVER ABORTED
```

and the program returns to the original menu of the UTILITY subsystem.

Note

If any error occurs that causes the RECOVER to abort, the recovery is invalid and should not be used.

Recovery of a Corrupt Data Base

In the event that your data base becomes corrupt and there is no backup version, it may be possible to salvage all or most of the data by using the UNLOAD and LOAD utilities.

UNLOAD "unloads" all or part of the data into a file which contains raw data but no structural information. LOAD then transfers the data back into the structure, automatically reestablishing linkage information. These operations can be fairly time-consuming, but keep you from having to transfer the data yourself.

The UNLOAD utility does not transfer data entries from automatic master sets. Automatic masters are regenerated automatically by the LOAD utility.

There is another method by which data can be read from a corrupt data base. It involves opening the data base with DBOPEN mode 8, which returns +94 in the Condition Word of the status array, indicating that a corrupt data base was successfully opened. In general, most of the data entries can be read using DBGET, using either serial mode or directed mode. Retrieval can continue in this fashion unless invalid information is found in the root file. At this point, remaining sets may or may not be readable.

Using the UNLOAD Utility

Note

The destination (unload) file cannot exist before running UNLOAD, since it is created by the utility program.

Enter the UTILITIES subsystem by pressing the DATA BASE UTILITIES softkey from the initial QUERY/9000 menu. To run the UNLOAD utility, press the softkey labeled UNLOAD DATA BASE.

The program then asks for information about the data base you want to unload. Use RETURN to move the cursor after you have entered information for each category.

- Data base name
- Root file volume
- Root file path
- Data base password

If you do not specify a root file volume or a root file path, the program searches the current directory for the data base.

When you have entered all the information requested, press the PROCESS BASE INFO softkey to continue.

The program then asks for information about the destination file. Use RETURN to move the cursor after you have entered information for each category.

- Load file name
- Destination volume
- Destination path
- File protect code (leave blank if no protect code is desired)

If you do not specify a destination volume or a destination path, the load file name is created in the current directory.

When you have provided this information, press the EXECUTE UNLOAD softkey to run the utility.

UNLOAD Options
Four softkeys are displayed so you can choose how the data is to be unloaded:

- SERIAL BASE UNLD
- CHAINED BASE UNLD
- SERIAL SET UNLD
- CHAINED SET UNLD

You may unload either the entire data base (all sets except automatic masters, which are regenerated automatically by LOAD) or a selected set.

Note

You can find out which sets (if any) are corrupt by using the DBSTATUS statement. See DBSTATUS in the BASIC Language Reference.

You can also specify either a chained or serial unload. Serial mode uses the physical order of the entries and is faster; it must be used for recovering from a corrupt data base. Chained mode is recommended for non-corrupt detail sets. It can improve access times in the new data base by placing chained detail data entries close together.

If you choose to do a serial set or chained set unload, you can then select the sets to be unloaded. Use the YES and NO softkeys to move through the sets listed. When you are finished selecting, press EXECUTE UNLOAD to continue.

Messages similar to these are displayed in all cases:

```
Creating Unload file XXXXXX
Unloading set XXXXXX                record XX of XX
```

After the data base has been successfully unloaded, the message UNLOAD COMPLETE is displayed, and the program returns to the original menu of the UTILITY subsystem.

Using the LOAD Utility

LOAD does not erase any files. You must erase the data sets or the data base from the destination data base before the LOAD is executed.

Enter the UTILITIES subsystem by pressing the DATA BASE UTILITIES softkey from the initial QUERY/9000 menu. To run the LOAD utility, press the softkey labeled LOAD DATA BASE.

The program then asks for information about the destination data base. Use RETURN to move the cursor after you have entered information for each category.

- Data base name
- Root file volume
- Root file path
- Data base password

If you do not specify a root file volume or a root file path, the program searches the current directory for the data base.

When you have entered all the information requested, press the PROCESS BASE INFO softkey to continue.

The program then asks for information about the source file. Use RETURN to move the cursor after you have entered information for each category.

- Load file name
- Source volume
- Source path
- File protect code

If you do not specify a source volume or a source path, the program searches the current directory for the load file.

When you have provided this information, press the EXECUTE LOAD softkey to run the utility.

Messages similar to this are displayed:

```
Loading set XXXXXX          record XX of XX
```

After the data base has been successfully loaded, the message `LOAD COMPLETE` is displayed, and the program returns to the original menu of the UTILITY subsystem.

Special Considerations

If a duplicate key item already exists before the LOAD, this message is displayed, and the LOAD continues:

```
Duplicate key item.  Item entry XX not added to set XXXXXX
```

If you attempt to overload a data set (the number of data entries added is greater than the set capacity), this message is displayed:

```
Data set XXXXXX is full.  Loading stopped on item XX!
```

If you attempt to add a detail data entry with a search item value that does not match any existing key item value in the corresponding manual master set, this message is displayed:

```
No chain head for set XXXXXX.
Item entry XX not added to set XXXXXX
```

If you attempt to add a detail data entry with a search item value that does not match any existing key item value in the corresponding automatic master set, and that automatic master set is full (the number of data entries added is greater than the set capacity), this message is displayed:

```
Automatic master path XXXXXX is full.
Item entry XX not added to set XXXXXX
```

Erasing Data Base Information

There are two large-scale erasure utilites that can be performed on a data base. These should be used with care.

- ERASE erases all the data entries and related path information from all of the data base sets, returning the data sets to the state they were in when DBCREATE was executed. It can be used when the information in a data set is unwanted. It can also be used before running the LOAD utility (to recover from a corrupt data base).
- PURGE either removes all of the data set files, or all data set files, form files, thread file, root file, and the data base directory, from the mass storage medium. After a set is purged, its data is lost. After the root file is purged, the entire data base structure is lost. PURGE can be used to purge an unwanted data base. It can also be used before a RECOVER is performed.

Using the ERASE Utility

Enter the UTILITIES subsystem by pressing the DATA BASE UTILITIES softkey from the initial QUERY/9000 menu. To run the ERASE utility, press the softkey labeled ERASE DATA BASE.

The program then asks for information about the data base to be erased. Use (RETURN) to move the cursor after you have entered information for each category.

- Data base name
- Root file volume
- Root file path
- Data base maintenance word

If you do not specify a root file volume or a root file path, the program searches the current directory for the data base.

When you have provided this information, press the EXECUTE ERASE softkey to run the utility.

After the data base has been successfully erased, the message ERASE COMPLETE is displayed along with three softkeys:

- DUMP STATUS – prints the same report that DBSTATUS does, listing all sets, what type they are, their capacities (in records), number of entries, their status (erased, purged, or corrupt) and location in file hierarchy.
- CAT DB-FILES – prints the same report that CAT does, listing every data set, its level, system type, file type, number of records and record length, latest modification date and time, public access and open status.
- UTILITIES OPTION – returns to the original menu of the UTILITY subsystem.

Using the PURGE Utility

Enter the UTILITIES subsystem by pressing the DATA BASE UTILITIES softkey from the initial QUERY/9000 menu. To run the PURGE utility, press the softkey labeled PURGE DATA BASE.

The program then asks for information about the data base to be purged. Use (RETURN) to move the cursor after you have entered information for each category.

- Data base name
- Root file volume
- Root file path
- Data base maintenance word

When you have provided this information, press the EXECUTE PURGE softkey. Then choose one of two functions displayed:

- PURGE SETS – purges all sets in the data base.
- PURGE ALL – purges all sets, form files, thread file, and the root file of the data base, along with the data base directory.

After pressing one of the two PURGE softkeys (and the data base has been successfully purged) or the ABORT softkey, the message PURGE COMPLETE is displayed along with three softkeys:

- DUMP STATUS – prints the same report that DBSTATUS does, listing all sets, what type they are, their capacities (in records), number of entries, their status (erased, purged, or corrupt) and location in file hierarchy.

  However, if you have done a PURGE ALL, DUMP STATUS displays `Root file purged or not found.`
- CAT DB-FILES – prints the same report that CAT does, listing every data set, its level, system type, file type, number of records and record length, latest modification date and time, public access and open status.

  However, if you have done a PURGE ALL, CAT DB-FILES displays `Data base directory purged.`
- UTILITIES OPTION – returns to the original menu of the UTILITY subsystem.

Appendix A
Materials Supplied and System Configuration

Introduction

This appendix documents the material supplied with your HP 97053A IMAGE/DBM product. It is divided into two sections, one for the materials supplied, one for system configuration.

Materials Supplied

The HP 97053A IMAGE/DBM and QUERY/9000 software product is supported only in the HP 9000 BASIC Language System. It consists of a BIN option file which contains the BASIC Language data base statements, and several files containing QUERY, SCHEMA, the sample data base, and other utility programs and a set of manuals. The 97053A product is available only with media option 042, which supplies the software on five 5-¼ inch flexible discs. These discs are identified below. The revision level is 1.0.

The HP 97053A product includes the manuals listed below. To ensure proper operation of data base programs, your manual part numbers, edition dates and revision levels should match those shown here. If there is any discrepancy, contact your Hewlett-Packard Sales and Support office.

Part Number	Contents or Manual Title	Volume Label	Revision or Edition
97053-10024	IMAGE/BASIC Language Statements	97053-10024R0100	1.0
97053-10025	QUERY main and subprograms	97053-10025R0100	1.0
97053-10026	QUERY subprograms	97053-10026R0100	1.0
97053-10027	QUERY subprograms	97053-10027R0100	1.0
97053-10028	QUERY subprograms, SCHEMA and transport utilities	97053-10028R0100	1.0
97053-90000	IMAGE Data Base Programming Techniques manual		E1282
97053-90001	Data Base Design Kit manual		E1282
97053-90002	QUERY User's Guide		E1282
*97050-90005	BASIC Language Reference for the HP 9000 Model 20		E1282

* This manual is supplied with the BASIC Language System product (HP 97050) but is necessary here for correct use of the IMAGE statements.

System Configuration

To run your QUERY/9000 Data Base Management System, you need:

- HP 9000 Model 20 with at least 350K bytes of Read/Write Memory (after all binaries are loaded).
- An internal printer.
- Mass Storage Option (MS_B0100 binary).
- IMAGE Option (IMAGE_DBM_B0100 binary).

Also recommended:

- Error messages (ERRORS_ENG_B0100 binary).

Now refer to Chapter 1 of this manual for instructions on how to install and use QUERY/9000.

Manual Comment Sheet Instructions

If you have any comments or questions regarding this manual, write them on the enclosed comment sheets and place them in the mail. Include page numbers with your comments wherever possible.

If there is a revision number, (found on the Printing History page), include it on the comment sheet. Also include a return address so that we can respond as soon as possible.

The sheets are designed to be folded into thirds along the dotted lines and taped closed. Do not use staples.

Thank you for your time and interest.