Query Reference Manual
B.06.32

Edition E1202

Legal Notices

The information contained in this document is subject to change without notice.

MARXMEIER SOFTWARE AG MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Marxmeier Software AG shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c) (1,2).

Acknowledgments

Marxmeier Software AG
Besenbruchstrasse 9
42285 Wuppertal
Germany

Printing History

The manual printing date indicates its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. New editions are complete revisions of the manual.The dates on the title page change only when a new edition or a new update is published.

Manual updates may be issued between editions to correct errors or document product changes. Manuals that are published on the Eloquence website (www.hp-eloquence.com/doc) may be updated more often, please visit this website periodically for the most recent versions. To ensure that you receive the updated or new editions, you should also subscribe to the appropriate product support service.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Apr 1990	A.01.00
Second Edition	July 1991	A.03.00
Third Edition	February 1992	A.03.10
Fourth Edition	July 1997	A.06.00
Fifth Edition	October 1997	A.06.00
Sixth Edition (E1202)	December 2002	B.06.32

Printed in the Federal Republic of Germany.

Printing History

Table of Contents

Contents

Contents

Contents

1

Introduction to Query

The Query software provides a simple method of accessing an Eloquence database without programming effort. You may use Query to do the following:

- Retrieve data which meets selection criteria.
- Report on the data retrieved.
- Add or delete data entries.

- Modify data entries.

NOTE: In order to use Query, a VOLUME statement pointing to the directory /opt/eloquence/share/prog must be included in the global configuration file eloq.config. The sample global configuration file d.eloq.config contains such a statement (**`VOLUME SYSTEM :Z2,7,0 /opt/eloquence/share/prog`**).

NOTE: Query requires a character I/O interface and so it is not supported on the NT platform. This manual describes operation of the Query software for the most recent version of Eloquence. (At this edition of the *Eloquence Query Manual*, A.06.00 is the most recent version.)

Conventions

The statements in this manual use the same syntax conventions as in the *Eloquence Manual*.

- **Bold type** is used when a new term is introduced.
- **`Computer font`** indicates text to be input exactly as shown or text that is output from the system.
- *Italic type* is used for emphasis and titles of publications. It is also used to indicate parameters that are user defined.
- KEYCAP represents a key on the keyboard.
- Shading represents the softkeys displayed on the computer screen.
- …indicates that the previous variable can be repeated.
- [ ] indicates that information inside the brackets is optional. If there are brackets within brackets, the information within the inner bracket may only be specified if the information in the outer bracket is specified. Information may also be stacked in brackets. For example, A or B or neither may be selected when the following is shown:

  $$\left[\begin{matrix} A \\ B \end{matrix}\right]$$

- { } indicates that one of the choices stacked within the braces must be selected. For example, A or B or C must be selected when the following is shown:

  $$\left\{\begin{matrix} A \\ B \\ C \end{matrix}\right\}$$

NOTE: Notes contain important information that is set off from the text.

Introduction to Database Management

A **database** is a collection of related information, commonly accessible, that has been stored in such a way that easy access to the information is made possible. The person who designs the database decides what information will be in the database and how each piece of information is related to every other piece of information. To use Query, you need not know the process the designer uses to create the database, but you do need to know what the information is and how it is organized.

The information stored in the database depends upon the use of the database. Throughout this manual, a sales analysis example is used. This is only an example to show you how to work with the database. After you have read this first section, ask your manager or the person in charge of the computer system what information your database stores.

Data Item

The smallest piece of information in a database is called a **data item**. For example, a customer name would be a data item, a customer address would be another data item. When the database is designed, a certain amount of space is reserved for storing the contents of each data item. For example, assume the customer information is stored in a card file. The first line might be reserved for the customer's name, the next two lines for the address, and so on.

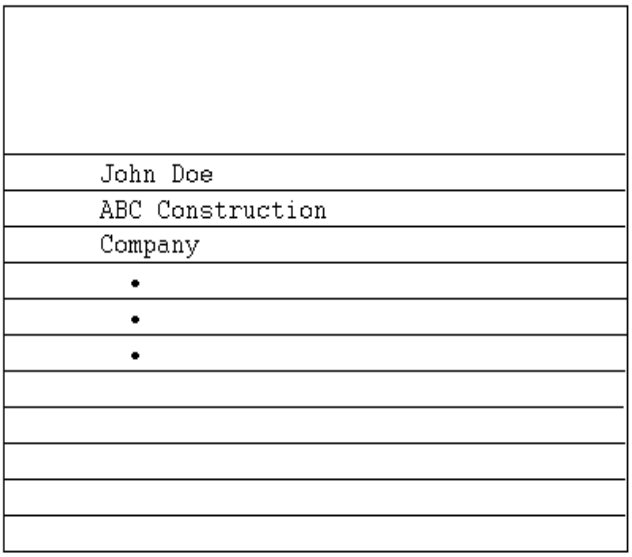


Figure 1 **Example Data Items**

Thus the first data item uses the first line and the second data item uses the next two lines. Note that cards are not really used for storage. This is only an example to help you visualize data items.

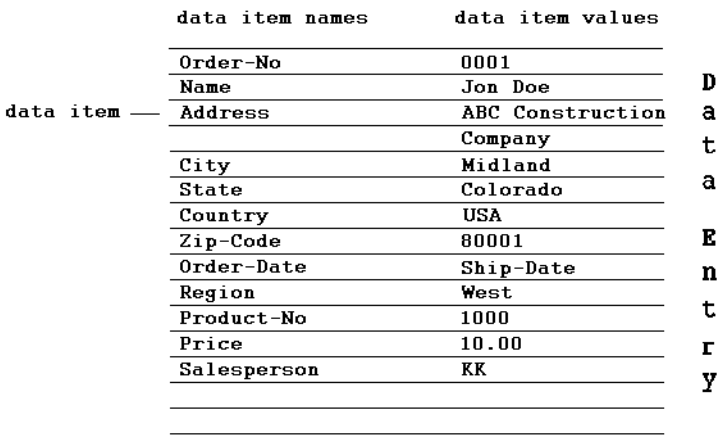
When it becomes necessary to find the card which contains the customer Jon Doe, you need a way to specify which line on each card is to be searched. Consequently, each data item is assigned a name and length. For example, the first data item might be named “customer” and have a length of one line. (When stored in the database, the length is actually specified as a number of characters or digits.)

Data Entry

The next level in the database system is the **data entry**. All the data items on a card might be defined as a data entry.

Figure 2 **Data Entry Example**

The figure shows a data entry that consists of 13 data items.



Data Set

A collection of similar data entries is known as a **data set**. The following figure shows three data entries which together make up a particular data set that will be referred to as the CUSTOMER data set. A data set is similar to a file.

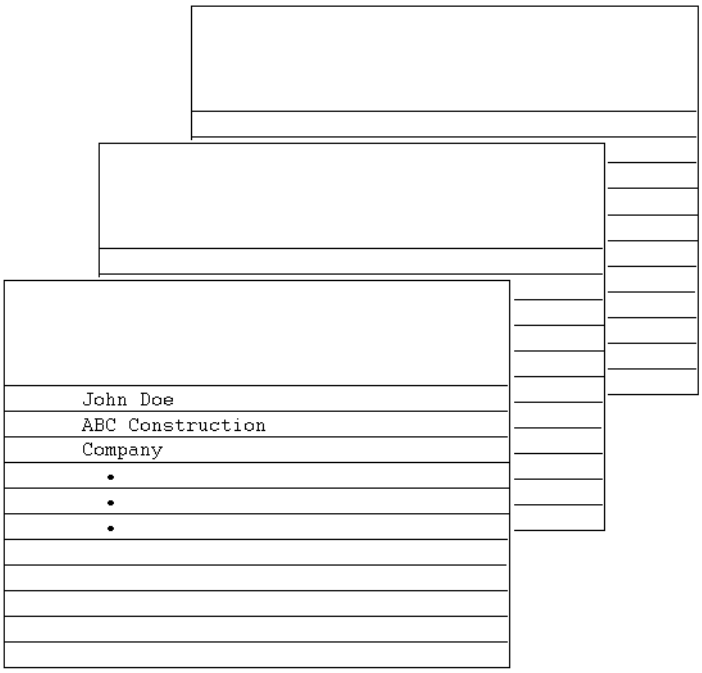


Figure 3 **Example Data Sets**

A collection of data sets is called a **database**. The data sets in a database are usually linked together. For example, a second data set in the Sales Analysis database contains the data items ORDER_NO, OPTION_DESC, OPTION_PRICE and OPTION_TYPE; this data set is referred to as the OPTION data set. Because the information in this data set may be wanted with the information in the customer data set, the two are linked together through a third data set, ORDER, which has one data item ORDER_NO. This link from one data set to another is called a **data path**.

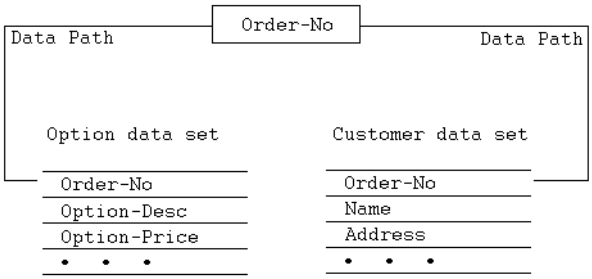


Figure 4 **Example Data Path**

The data item which is linked is called a **key item**.

Now assume the customer specifies two options on the same order. This would cause the option data set to have two entries with the same order number. These entries are also linked through the key item. This link, within a data set, is called a **data chain**.

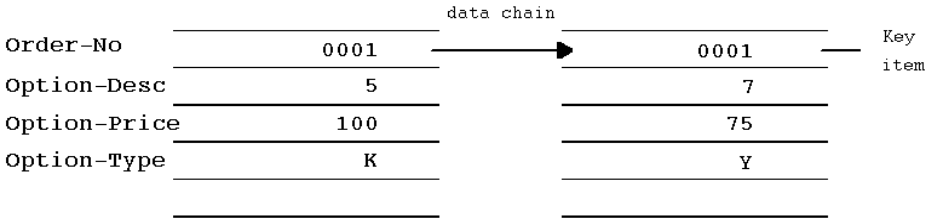


Figure 5 **Example Data Chain**

Types of Data Sets

There are two types of data sets, **master data sets** and **detail data sets**.

In a master data set, there is only one data entry for each key item value. There may be up to 16 data paths from this key item.

In a detail data set, there may be more than one data entry for each key item value. There may be up to sixteen key items in a data entry. For every value of a key item in a detail data set, there is a corresponding value in its associated master data set.

For example, the order data set has one data entry for each ORDER_NO value. The option data set has two data entries for ORDER_NO 0001. Since there is a value of 0001 in the option data set ORDER_NO data item, there must be a value of 0001 in the order data set ORDER_NO data item. However, there does not have to be the same value in the customer data set.

There are two types of master data sets. The **automatic master** data set contains one data item which is a key item. When a new value is added to 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 a particular key item value is deleted from the detail data set.

The **manual master** data set contains one or more data items, one of which is a 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 be first added to the manual master data set.

Database

The collection of multiple detail data sets and master data sets is collectively known as a **database**. As the information is stored into the database, no ordering is necessary since the links provide the ordering. All data chain maintenance is performed automatically.

The following is a diagram of the Sales Analysis database showing the data paths between master and detail data sets.

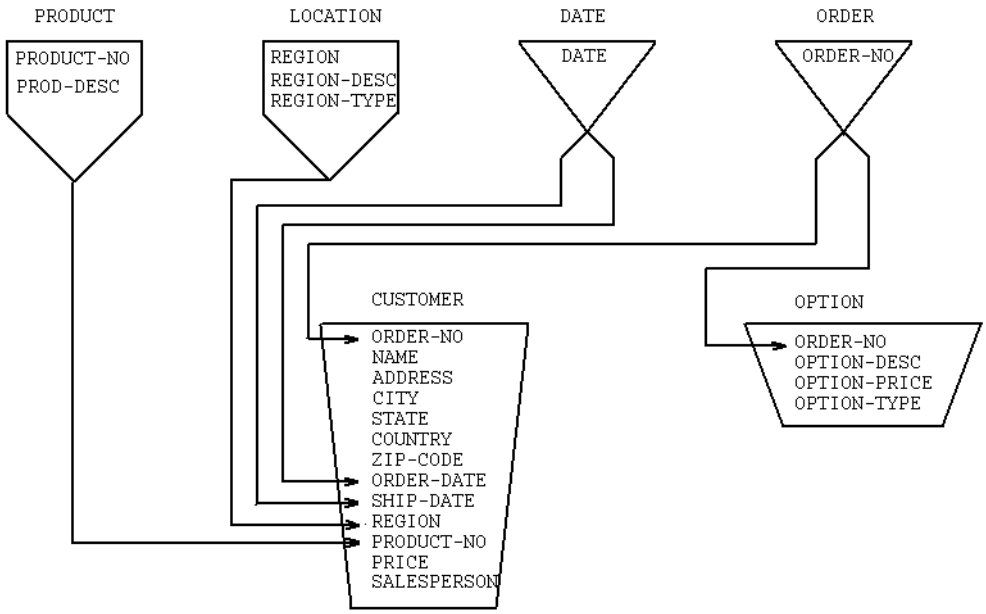


Figure 6 **Sales Analysis Database**

Database Schema

A **database schema** is used to define the database. There are three parts to a schema. Part one gives the name of the database and passwords used for access to the database. Part two lists each data item, specifying its size and type. (The type of a data item refers to the type of data that will be stored in it, such as integer data. For a full description, refer to the appendix titled “DBML Syntax” in the *Eloquence DBMS Manual*.) Part three describes the individual data sets and the data items they contain.

Now let’s look at the schema for the Sales Analysis database.

```
1  BEGIN DATA BASE     SAD;  <<CUSTOMER SALES ANALYSIS DATABASE>>
2
3  PASSWORDS:
4                10       SALESMAN;
5                15       MANAGER;
6                 3       SECRTARY;  <<WILL HAVE READ ACCESS ONLY>>
7
8  ITEMS:
9                ADDRESS,     2X30;  <<2 LINES OF ADDRESS ALLOWED>>
10               CITY,         X16;
11               COUNTRY,      X12;
12              DATE,        I;    <<PATH FOR ORDER-DATE, SHIP-DATE>>
13               NAME,         X30;
14               OPTION-DESC,  X10;
15               OPTION-PRICE, L;
16               OPTION-TYPE,  I;
17               ORDER-DATE,   I;    <<MUST BE YYMM>>
18               ORDER-NO,     X10;
19               PRICE,        L;
20               PRODUCT-NO,   I;
21               PROD-DESC,    X30;
22               REGION,       X6;
23               REGION-DESC,  X30;
24               REGION-TYPE,  I;
25               SALESPERSON,  X4;
26               SHIP-DATE,    I;    <<MUST BE YYMM>>
27               STATE,        X6;
28               ZIP-CODE,     X8;
29
30 SETS:
31
32    NAME:     DATE,AUTOMATIC(3/10,15);
33    ENTRY:    DATE(2);
34    CAPACITY: 50;   <<CAPACITY IS OPTIONAL>>
35
36    NAME:     ORDER,A(3/10,15);
37    ENTRY:    ORDER-NO(2);
38    CAPACITY: 100;
39
40    NAME:     PRODUCT,MANUAL(3,10/15);
41    ENTRY:    PRODUCT-NO(1),
42              PROD-DESC;
```

```
43    CAPACITY: 10;
44
45    NAME:     LOCATION,M(3,10/15);
46    ENTRY:    REGION(1),
47              REGION-DESC,
48              REGION-TYPE;
49    CAPACITY: 20;
50
51    NAME:     OPTION,D(3/10,15);
52    ENTRY:    ORDER-NO(ORDER),
53              OPTION-DESC,
54              OPTION-PRICE,
55              OPTION-TYPE;
56    CAPACITY: 300;
57
58    NAME:     CUSTOMER,DETAIL(3/10,15);
59    ENTRY:    ORDER-NO(ORDER),
60              NAME,
61              ADDRESS,
62              CITY,
63              STATE,
64              COUNTRY,
65              ZIP-CODE,
66              ORDER-DATE(DATE),
67              SHIP-DATE(DATE),
68              REGION(LOCATION),
69              PRODUCT-NO(PRODUCT),
70              PRICE,
71              SALESPERSON;
72    CAPACITY: 100;
73
74          END.
```

Line 1 The name of the database is given—SAD. When you use Query, you must give the database name.

Lines 3, 4, 5, and 6 The passwords are given. The numbers (10, 15, 3) are used later in the schema. You use the words SALESMAN, MANAGER, SECRTARY when you first begin using Query (as explained in page 23).

Lines 8 through 28 The name, size and type of each data item is given. **`2X30`** specifies a two item array, of which each item is 30 characters long. **`X16`** specifies a data item 16 characters long. **`I`** specifies that the data item contains an integer. **`L`** specifies that the data item contains a long precision number. If a data item is a number, then you can only enter a number for its value; otherwise an error will occur. If data item is a string (specified by **`X`**) then you may enter up to the specified number of characters. (Entering a value is described in page 57 .)

Lines 32, 33, and 34 The automatic data set DATE is described. Line 32 gives the

name (DATE), the type (AUTOMATIC), the read/write passwords (3/10, 15). The passwords (3/10, 15) mean that anyone who uses the password SECRTARY (3) can read and list the data in this data set, but cannot write new values, change values or delete values. Anyone who uses the password SALESMAN or MANAGER can read or write (add, delete, modify) the values in this data set. Line 33 gives the name of data item (DATE) and the number of data items in detail data sets to which are pointed (2). Line 34 gives the maximum number of data entries that are possible in this data set.

Lines 36 through 49 The other master data sets are described.

Lines 51 through 56 The detail data set OPTION is described.

Line 52 This specifies the name of the first data item (ORDER_NO) which is a key item pointed to from the master data set ORDER.

Lines 58 through 72 The detail data set CUSTOMER is described.

Line 74 This specifies the end of a schema.

Within the schema, comments are enclosed in signs. These comments aid you in understanding the schema. They also provide additional information about data item values. For example, line 12 has the following comment:

```
<<PATH FOR ORDER-DATE, SHIP-DATE>>
```

This tells the reader of the schema that this data item is used as the link between ORDER-DATE data item and SHIP-DATE data item.

Lines 17 and 26 both have the following comment:

```
<<MUST BE YYMM>>
```

This tells the reader that the values entered for the data must be formatted as YYMM. (YY means a two digit representation of the year. MM means a two digit representation of the month. For example, October 1986 represented in YYMM format would be 8610.) This is not required by the schema, but because reports are written expecting this format, you should use it.

In the schema, some data item names contain a dash (for example, ZIP-CODE). When you use this data item in Query, the dash must be changed to an underscore. Thus, ZIP-CODE would be written as ZIP_CODE.

2

Getting Started

Running Query

Run Query by typing the following HP-UX command:

```
$eloqcore QUERY,SYSTEM
```

Specifying a Database

As Query begins execution it displays the following prompt, asking you to enter the name of the database and to identify the volume on which it is located:

```
Enter the database name and volume:
```

When this prompt is displayed, enter in the database name and the volume specifier of the storage device, in the following form:

database name[*volume spec*]

If you do not provide a *volume specifier* when entering the name of the database, Query assumes that the database resides on the default MSI.

Query next finds the database you specified and prepares it for use. If for some reason Query cannot find the database you specified, it displays the following error message:

```
DATABASE NOT FOUND OR IN USE
```

If this should happen, check that you have entered the correct database name and volume specifier. If you have entered the information correctly and Query still displays the error message, contact your System Administrator for assistance.

Note that all error messages Query displays are described in Appendix B.

NOTE: QUERY will use the directory **`/tmp`** for temporary file storage. If this is not available, the System Administator should create **`/tmp`**, and grant read/write access to QUERY.

Examples

Assume that when starting Query, you executed the following commands:

```
MSI "DBASE" RETURN
RUN "QUERY,SYSTEM" RETURN
```

Query then displays a prompt, asking you to enter the name of the database and to identify the volume on which the database resides. If the database is named SAD and is stored on the volume whose label is DBASE, you need only type the following:

```
SAD RETURN
```

If the SAD database is stored in a directory whose volume name is FILES (not the default MSI), you would enter the following:

```
SAD, FILES RETURN
```

Specifying a Password

Once you have supplied the name of the database, Query asks you to enter your password by displaying the following prompt:

```
Enter the password:
```

Type your password and then press RETURN; the characters you type in response to this prompt *are not displayed*.

For example, if your password is OPERATOR, then you would type the following:

```
OPERATOR RETURN
```

After you press RETURN, the password prompt is cleared from the display; you are now ready to use the database.

NOTE: The database name, volume, and password must be specified before you use any Query command, except the DO command. (The DO command is described in page 34 .)

Communicating With Query

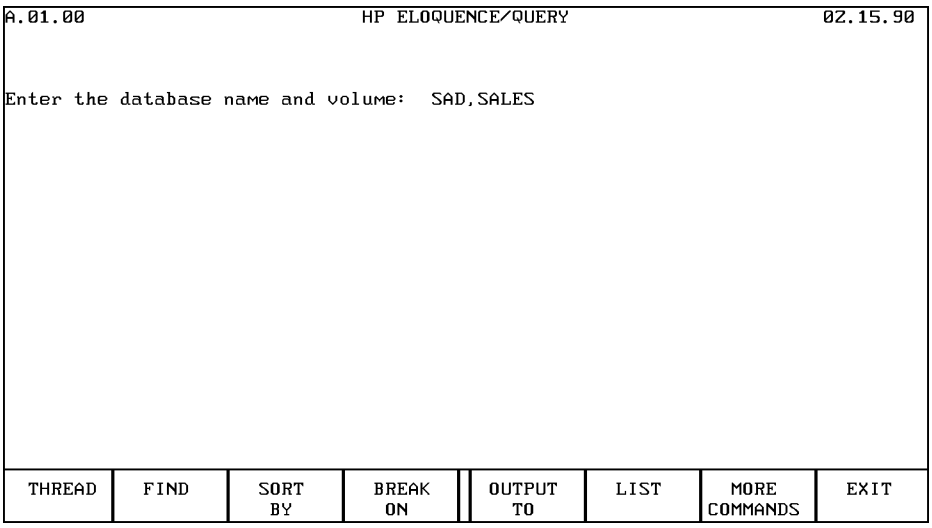
Query provides access to an Eloquence database through the execution of simple commands. Query commands can be executed in one of the following ways:

- Using your terminal's softkeys. This procedure is explained in page 26 .
- Typing in the command name and any information needed by the command (such as database name) and then pressing RETURN. The syntax for Query commands is discussed throughout this manual and is summarized in Appendix A.
- Storing a list of commands in a file; then execute Query's DO command by either of the preceding methods. The DO command is discussed in page 34 .

Use the method of command execution that is most convenient for you.

Using Softkeys

Each terminal keyboard is equipped with eight softkeys. These keys are labeled "f1" through "f8" on the keyboard. (If you cannot locate the softkeys, refer to your terminal manual.) Query defines each softkey to perform a certain function when that key is pressed. Query indicates the current function of each softkey by displaying a label for each key. For example, once you successfully supply the database name and password, Query defines the softkeys, displaying the following on your terminal:



The position of a label on the screen corresponds with the location of a function key on the keyboard. For example, the left-most label on the screen (THREAD) defines the function of the left-most function key (key f1). With these softkey definitions, pressing f1 executes Query's THREAD command. Similarly, the right-most label on the screen (EXIT) defines the function of the right-most function key (key f8). With each different screen displayed, Query can modify the definition and the label of the function key, providing only those responses that are relevant for the information displayed.

Examples

Notice that the preceding display is identical to the display on your terminal once the database name and password are entered. The softkey label second from the right on this display is MORE COMMANDS; this corresponds to function key f7.

Press f7. Notice that the displayed labels change; the softkeys have new definitions, as shown below:

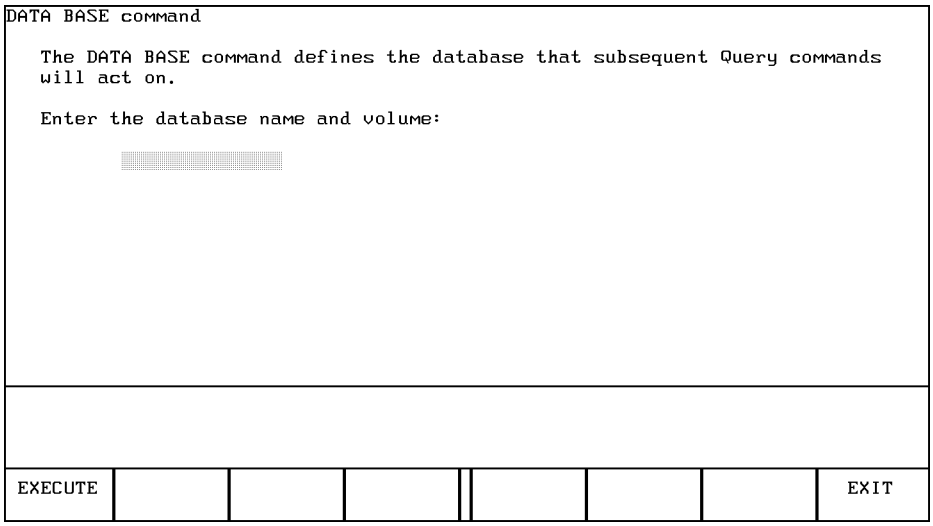
ADD	DELETE	REPLACE			LINEAR LIST	MORE COMMANDS	EXIT

Press f7 again. Once more the displayed labels change; the softkeys have new definitions, as shown below:

DATA BASE	PASSWORD	DO	RUN	WORKFILE	INFO	MORE COMMANDS	EXIT

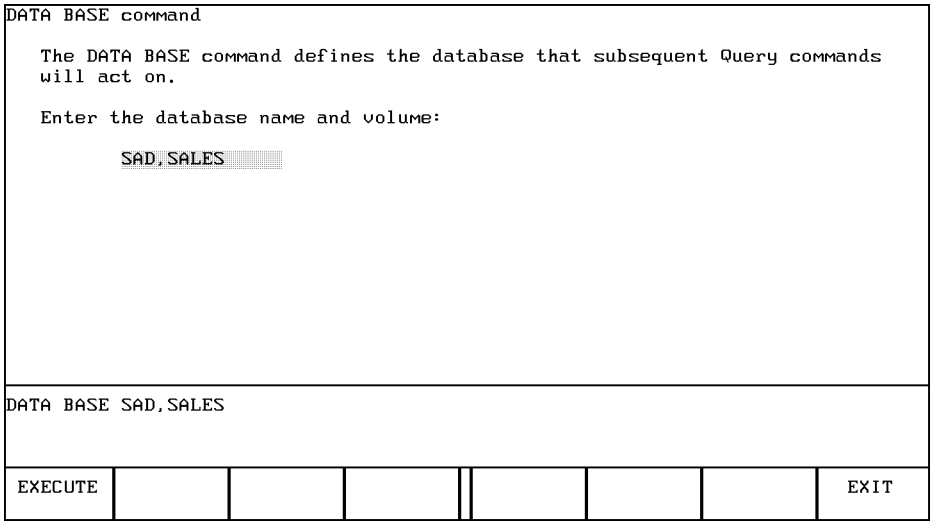
Press f7 once more to change the softkey definitions back to their original values. All possible Query commands are represented on the softkey labels and are accessed by using the MORE COMMANDS softkey. A Query command is accessed by first using the MORE COMMANDS softkey until the command name is displayed as a softkey label; then press the softkey that corresponds with the label identifying the desired Query command.

For example, to execute Query's DATABASE command, press the MORE COMMANDS softkey until the DATABASE softkey is displayed. Press the DATABASE softkey, and the following information is displayed:



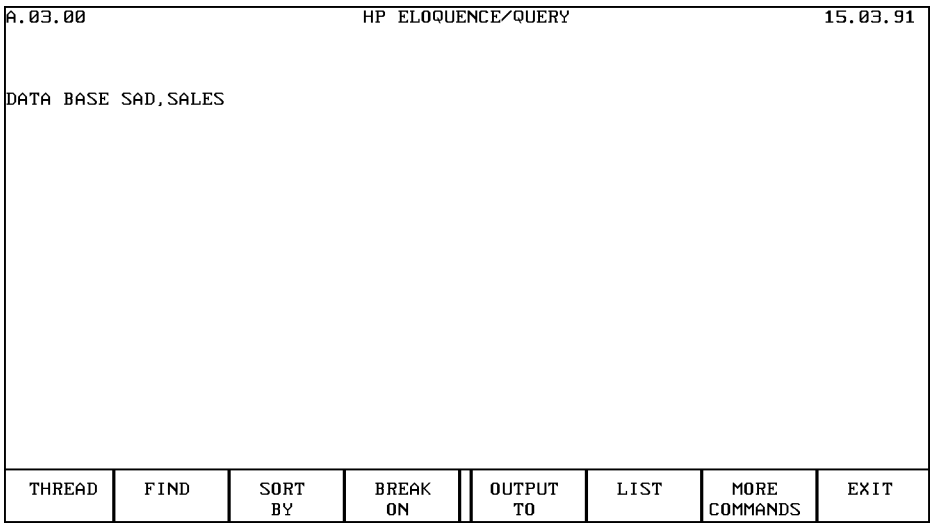
The command name and an explanation of the command are displayed. (If, after reading the explanation, you decide not to use this command, press the EXIT softkey to return to the first command menu.) The cursor is located in the inverse video field; this indicates that the computer is waiting for you to type in the name of the new database. Type in the name of the new database; if you are just practicing for now, type in any name; then press RETURN.

Upon pressing RETURN, a screen similar to the following is displayed:



Notice that the command, complete with the database name, is written on the lower part of the display; the cursor remains in the inverse video field. If the database name displayed with the command name is not correct, simply type the correct name in the inverse video field and press RETURN again.

When the command is printed exactly as you want it, press the EXECUTE softkey. The command menu is displayed again, while the command to be executed is displayed at the top of the screen; Query then executes the command.



In many of the command displays, there is more than one field. Query makes it easy to move the cursor from one field to another. To move the cursor from its current field to the next field, press either the TAB or RETURN key. To move the cursor from its current field to the previous field, press SHIFTTAB. Additionally, you can use the four arrowkeys to move the cursor to the desired field. Only the RETURN key, however, will cause the command to be displayed.

To change what you have entered, position the cursor in the field containing the incorrect information and type the correct information in the field; then press RETURN to display the changed command.

Changing Databases

When you want to work with a different database, execute Query's DATABASE command to tell Query the new name of the database. To execute this command, use your terminal's softkeys or type the following (where name is the name of the new database):

DATA BASE *name* [*volume spec*]

The words in upper case must be entered exactly as shown. Remember, anything shown in brackets is optional. For example, if the name of the new database is SAD and it is stored on the volume whose label is FILES, you would type the following:

```
DATA BASE SAD, FILES RETURN
```

Changing Passwords

Frequently, a database has several passwords. Each different password can provide a different set of capabilities. For example, a user who provides one password might only be allowed to read certain records in the database; a user providing a different password might have the ability to read and modify all records in the database. Changing your password can provide you with different access capabilities.

Additionally, different databases often use different passwords to provide better data security. After changing databases with Query's DATABASE command, you may also have to change your password before you can work with the new database.

To enter a new password, use your terminal's softkeys to execute the PASSWORD command, or type the following (where *password* is the new password):

PASSWORD *password* RETURN

Database Information

If you want information about the database you are using, execute Query's INFO command by using your terminal's softkeys or by typing the following:

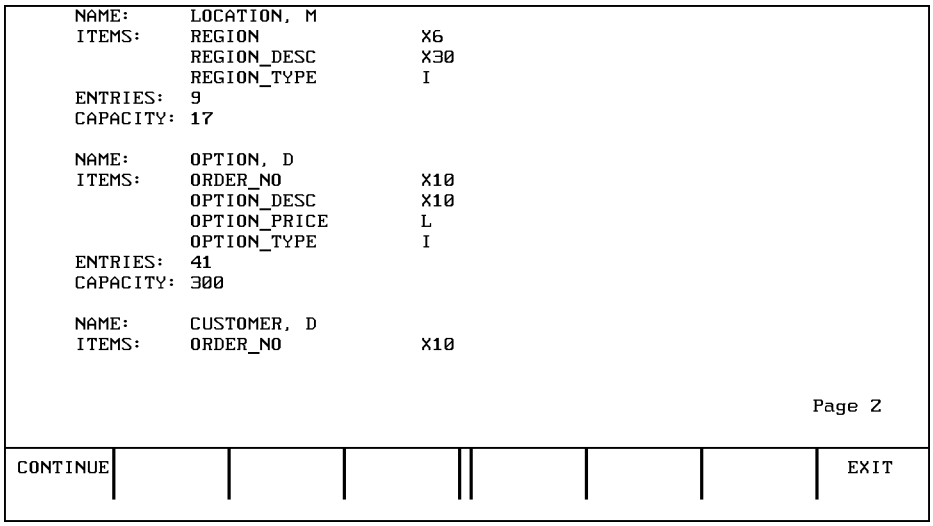
INFO RETURN

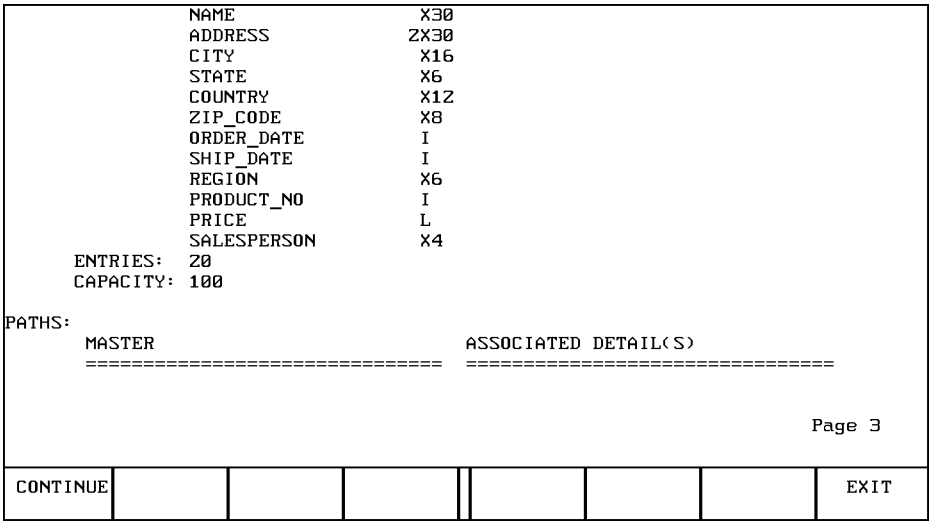
The function of the INFO command is to list a modified schema. The information is listed to the current output device (the device specified with the OUTPUT TO command); if no output device has yet been specified, the information is listed on the display. For example, the listing for the database SAD would appear as follows:

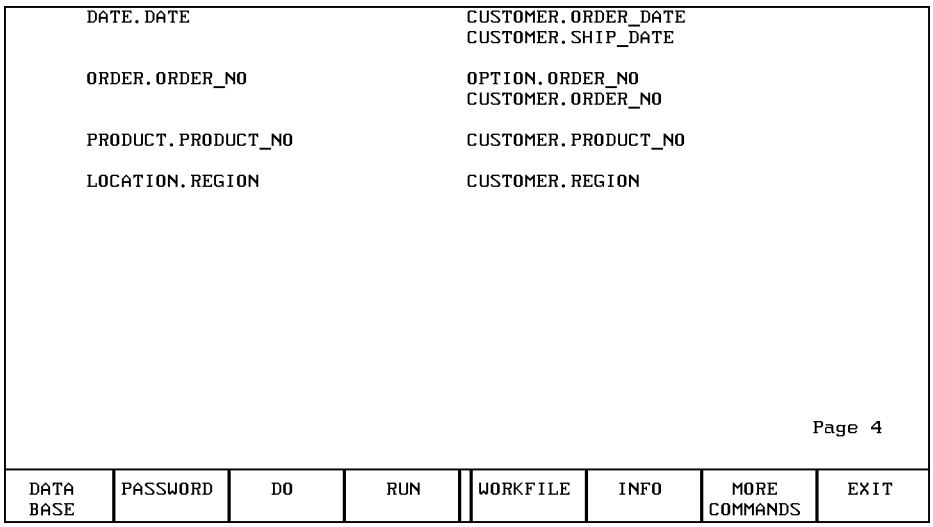
```
DATABASE: SAD,SALES

SETS:
      NAME:      DATE, A
      ITEMS:     DATE                 I
      ENTRIES:   14
      CAPACITY:  51

      NAME:      ORDER, A
      ITEMS:     ORDER_NO             X10
      ENTRIES:   20
      CAPACITY:  101

      NAME:      PRODUCT, M
      ITEMS:     PRODUCT_NO           I
                 PROD_DESC            X30
      ENTRIES:   5
      CAPACITY:  11


                                                          Page 1
```

CONTINUE							EXIT

Note the CONTINUE softkey. Since the entire listing won't fit on one display page. Query pauses after each page. Press the CONTINUE softkey to display the next page of information.


NAME: LOCATION, M
ITEMS: REGION X6
REGION_DESC X30
REGION_TYPE I
ENTRIES: 9
CAPACITY: 17
NAME: OPTION, D
ITEMS: ORDER_NO X10
OPTION_DESC X10
OPTION_PRICE L
OPTION_TYPE I
ENTRIES: 41
CAPACITY: 300
NAME: CUSTOMER, D
ITEMS: ORDER_NO X10
Page 2
CONTINUE
EXIT


NAME X30
ADDRESS ZX30
CITY X16
STATE X6
COUNTRY X12
ZIP_CODE X8
ORDER_DATE I
SHIP_DATE I
REGION X6
PRODUCT_NO I
PRICE L
SALESPERSON X4
ENTRIES: 20
CAPACITY: 100
PATHS:
MASTER
ASSOCIATED DETAIL(S)
Page 3
CONTINUE
EXIT



This first line displays the name of the database and the volume on which it is located. The next set of lines gives you the name and type of each data set. The item names, the number of entries currently in the data set and the maximum capacity of the data set are listed. The third set of lines lists the data paths between data sets. In the database SAD, six data paths exist.

You can compare this listing with the SAD database schema listed in page 9 . Note that the INFO listing does not show the database passwords. It does show the number of entries that have been made to the data set and shows the paths between master and detail data sets.

Prewritten Procedures

Sometimes a sequence of commands is done repeatedly. This sequence of commands is called a **procedure** and can be put into a data file (with extension name .DATA) and then executed with a single command. Once the procedure is in a file you can access it at any time.

NOTE: The DATA file containing the QUERY commands might be a “special DATA file”, i.e. a DATA file created with Eloquence and containing additional header information, or a plain HP-UX ascii file created, for example, by vi editor.

To execute the procedure, use your terminal’s softkeys or type the following to execute Query’s DO command:

DO "*filename* [*volume spec*]" RETURN

filename is the name of the file in which the procedure is stored, without the extension.

Query looks at the file on the volume specified and expects to find a procedure. For example, a procedure might consist of the following (the commands used are explained in the next section):

```
FIND OPTION FOR OPTION_PRICE>”100”
SORT BY OPTION_DESC, ORDER_NO
OUTPUT TO PRINTER
TOTAL OPTION_PRICE
LIST
```

Query will use the commands to find certain options, sort them, and list them on the printer, giving a total of OPTION_PRICE. While Query is busy, you will not be able to type commands.

Each command is displayed as it is processed (except the password command). The cursor reappears when Query has finished the procedure, allowing you to enter the next command.

Exit

When you have finished using Query, use your terminal's softkeys to execute the EXIT command or type the following:

EXIT RETURN

Executing EXIT logically terminates Query and closes the database.

3

Creating Reports

This chapter shows you how to list or report data in many forms and on different devices.

In order to report any data you must tell Query the following:

- How to find the data items and data entries that you want to list.
- Where the data items and data entries are to be listed (for example, to your terminal or to a specific printer).
- How the data items and data entries are to appear when listed.

Finding Entries

When telling Query how to find data items and data entries, you must first specify a location (called a workfile) where Query can store the information it finds; this is done with the WORKFILE command.

If the data you want to report is contained in one data set, you can then use Query's FIND command to find the data and copy it into the workfile. If the data is stored in more than one data set, you must first use the THREAD command to tell Query how the data sets are connected and in which order they are to be searched; then use the FIND command to find the data items and copy them into the workfile.

Specifying a Workfile

When finding data items and data entries for you, Query needs a place to temporarily store the information it finds; this location is called a **workfile**. Query automatically creates a workfile when told to find the first data item and then deletes this workfile when you exit the Query program.

You can tell Query to use a specific workfile by executing the following command:

WORKFILE ["*file name* [*volume spec*]"]

Notice that the file name is optional. If you do not supply a file name, Query creates its own workfile and then removes that file when you exit the Query program.

For example, to tell Query to use the file TEMP, located on the volume labeled FILES, as the workfile you would enter the following:

```
WORKFILE "TEMP, FILES" RETURN
```

Finding Entries in One Data Set

Once you have specified the workfile (or decided to let Query create its own), you are ready to find entries and copy them into the workfile. To find entries, you must know the names of the data sets and data items. If you do not know them, use the INFO command as described in page 31 .

To find entries, execute the FIND command:

FIND *item list* **FOR** *search expression*

The parameter *item list* specifies which data items are to be found; the parameter *search expression* specifies what conditions the data items must satisfy. A maximum of 60 data items may be specified in the item list and search expression combined.

For example, using the example database, assume you wanted to find the name of all customers who ordered product number 92640. You would enter the following command:

```
FIND NAME FOR CUSTOMER.PRODUCT_NO = "92640" RETURN
```

Query would find all the values of the NAME data item that have a corresponding value of 92640 for the PRODUCT_NO data item in the CUSTOMER data set. Then Query copies all these values into the workfile.

The syntax of the item list and search expression parameters and a description of their use is provided in the following paragraphs.

Item List

The item list parameter is a list of data items and data sets that you want Query to find. Query finds each data entry that satisfies the search expression. It then copies the values of all data items specified in the item list for that entry into the workfile. If only a data set is specified in the item list, the entire data entry is copied in the workfile.

There are two ways to specify a data item. If a data item's name is in only one data set, you need enter only the data item name. If the data item name is in two or more data sets, precede the item name with the set name and a period: *set name.item name*

For example, suppose that you wanted to know the customer order number for every customer that ordered product number 92640. In other words, you want to copy into the workfile the ORDER_NO and PRODUCT_NO data items from the CUSTOMER data set each time that the PRODUCT_NO data item in the CUSTOMER data set equals "92640". To do this you would type the following:

```
FIND CUSTOMER.ORDER_NO FOR CUSTOMER.PRODUCT_NO = ″92640″ RETURN
```

Alternately, if you want to copy the entire data entry into the workfile each time the CUSTOMER data set's PRODUCT_NO data item equals "92640", you would type the following:

```
FIND CUSTOMER FOR CUSTOMER.PRODUCT_NO = ″92640″ RETURN
```

As a final example, if you want to copy the values for the data items ORDER_NO, NAME, and PRODUCT_NO into the workfile each time that the CUSTOMER data set's PRODUCT_NO data item equals "92640", you would type the following:

```
FIND CUSTOMER.ORDER_NO,NAME FOR
      CUSTOMER.PRODUCT_NO = "92640" RETURN
```

If you fail to specify the data set and the requested data item exists in more than one data set, Query asks you from which data set you want the data item value. For example, using the example database, suppose that you entered the following:

```
FIND ORDER_NO FOR PRODUCT_NO = ”92640” RETURN
```

Query would then display the following message:

```
ORDER_NO is a member of these sets:
       1) ORDER
       2) OPTION
       3) CUSTOMER

Enter the number of the set you wish to use:
```

You would then type the number identifying the data set to be used and press RETURN.

If a data set and a data item have the same name and you want Query to copy the entire data set into the workfile, add a period to the data set name:

FIND *set name.* **FOR** *search expression*

Otherwise, in this situation Query interprets a name without a period as a data item name.

Search Expression

The search expression is a mathematical expression that is evaluated true or false. Data item names are used as variables. These data items cannot be arrays (such as the ADDRESS data item in the CUSTOMER data set).

Some simple examples of search expressions are:

```
SALESPERSON="SAM"

NAME > "A" AND NAME < "B"

NAME > "A" AND CUSTOMER.PART_NO < "999"
```

Notice that the values (such as SAM, A, B, and 999) to which data items are compared *must* be enclosed in quotes.

Query looks at a data entry and inserts the value of the data items specified into the search expression. If the expression evaluates true, Query copies all the data items specified in the item list and in the search expression from that data entry into the workfile. If the expression evaluates false, Query looks at the next entry. All entries are tested. A maximum of 1024 data items may be contained in the item list and search expression combined.

The search expression can be any expression using the following operators:

```
+   –   /   *   AND   OR   POS   >   <   =   #
```

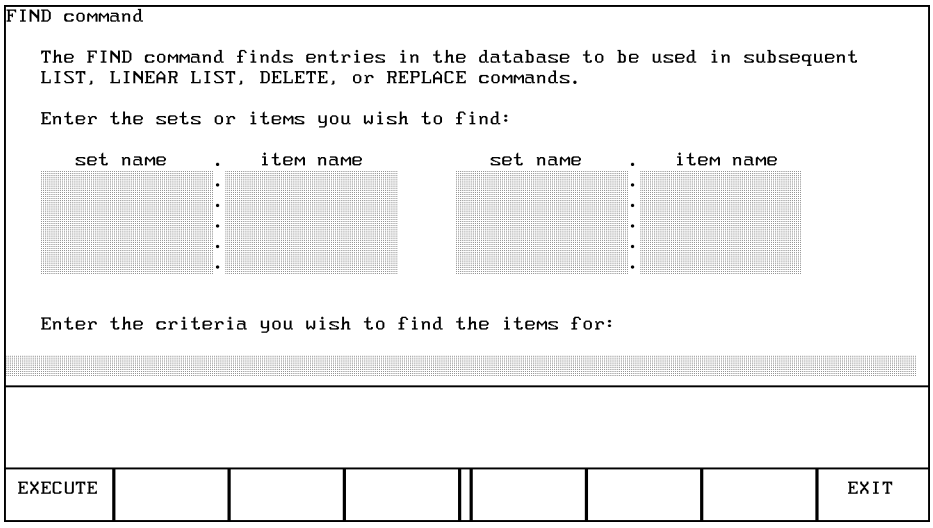
These operators are explained in Appendix C page 97 .

If you want to find all the entries for the item list, use the word **ALL** as the search expression. For example, to copy the name of every customer into the workfile you would type the following:

```
FIND NAME FOR ALL
```

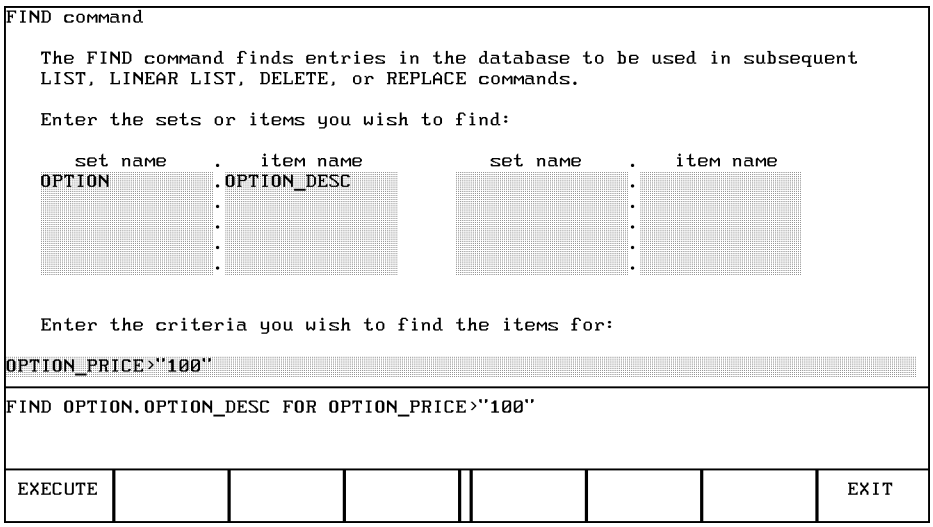
Using the FIND Softkey

If you press the FIND softkey to execute the FIND command, the following is displayed:



Enter list item values in the columns labeled “set name” and “item name”; enter the search expression in the line labeled “Enter the criteria you wish to find the items for:”.

For example, your display would resemble the following if you used the softkeys to enter this sample FIND command: FIND OPTION.OPTION_DESC FOR OPTION_PRICE>”100”



Once the item list and search expression fields are filled in, the command is shown above the softkey definitions. If it is not correct, use the editing key to move the cursor to the incorrect item in the inverse video boxes and retype the item. When the command is displayed correctly, press the EXECUTE softkey.

Finding Entries in Multiple Data Sets

In the previous paragraphs all data items specified were in the same data set. Before you can use multiple data sets, you must tell Query how and in what order the data sets are connected.

Threading Data Sets

Use the THREAD command to specify paths between data sets.

The data sets must be connected by a key item. For example, the PRODUCT data set is connected to the CUSTOMER data set with the key item PRODUCT_NO.

```
THREAD PRODUCT, CUSTOMER
```

The data set CUSTOMER is also connected to the ORDER data set by the key item ORDER_NO.

```
THREAD PRODUCT, CUSTOMER; CUSTOMER, ORDER
```

If you do not know how the data sets are connected, use the INFO command. The data paths are listed.

Note that you can only thread data sets from master to detail and detail to master.

You can extend the thread through 16 data sets as long as there are no breaks. For example, assume you had sets A, B, C, D, and E, with A connected to B, B connected to C, and D connected to E. You can thread A, B, and C together, but D and E cannot be on the thread. If you give a second THREAD command it cancels the first.

The syntax is as follows:

THREAD *set1, set2* [;*set2, set3* [;*set3, set4* ...]]

If one master data set has paths to more than one item in a detail data set, the data item to be threaded must be specified.

THREAD *set1.item1, set2.item2*; ...

If you do not give the item name, Query issues an error message.

The THREAD command stays in effect until the next THREAD command is given or until a single set FIND or ADD command is executed.

The FIND Command with Threaded Data Sets

After the THREAD command is given, you can find entries in more than one data set. For example, assume you want to know what options were ordered with a specific product. You would type the following:

```
THREAD CUSTOMER, ORDER; ORDER, OPTION RETURN

FIND OPTION FOR CUSTOMER.PRODUCT_NO=”100” RETURN
```

Query looks at the first data entry in the data set CUSTOMER and tests the search expression. If the expression is true, Query chains to the OPTION data set through the ORDER data set and copies all the values into the workfile. If the expression is false, or after the chain operation, Query tests the next entry, in CUSTOMER. If the chain length is zero (there are no entries in the OPTION data set), then no values are copied into the workfile even if the search expression (CUSTOMER<PRODUCT_NO = “100”) is satisfied.

In any multiple data set find, Query searches the first data set sequentially and all other data sets on the thread via the data paths and data chains.

Listing the Workfile

Once Query has found entries and copied them into the workfile, the contents of the workfile can be listed to a display, to a printer, or to a spool file. When listing the contents of the workfile, you must first tell Query to which device the information is to be listed; this is done with the OUTPUT TO command. Once Query knows which device to use, you can execute the LIST command to list data items in a columnar format or the LINEAR LIST command to list data items in a linear format.

Specifying the Output Device

To specify the device on which reports and the output from the INFO command are written, execute the OUTPUT TO command:

`OUTPUT TO` *device* [,"*width*"[,"*lenght*"]]

The parameter *device* tells Query the physical location of the output device. The following are acceptable values for *device*:

Value	Description
DISPLAY	Specifies that your terminal is the output device.
PRINTER	Specifies that printer defined with printer no. 0 is the output device.
STDOUT	Specifies that HP-UX stdout is the output device.
STDERR	Specifies that HP-UX stderr is the output device.
CONSOLE	Specifies that the console of your HP9000 system is the output device.
"spool file name"	Specifies that the listing is to be stored in a spool file to be used later.
"printer number"	An integer number specifying the printer number of the output device. (For more information on printer number, refer to the chapter “Output Operations” in the *Eloquence Manual*.)

The device specified by an OUTPUT TO command remains the designated output device until another OUTPUT TO command is executed.

The parameter *width* specifies the number of characters per line; valid values for *width* range from 20 through 264. If you do not specify it, Query chooses a default width depending on the output device specified: 80 characters for the display, 130 characters for the printer, 130 characters for spool files, and 80 characters for a printer number.

The parameter *length* is used to specify the number of lines per page on the output device. If you do not specify it, Query chooses a default device page length depending on the device specified: 20 lines for the display, 66 lines for the printer or spool file, and 66 lines for a printer number.

For example, assume you want to list the workfile to a printer whose printer number is 0; the printer contains narrow paper that holds 80 characters per line. You would type the following:

```
OUTPUT TO PRINTER, ”80”
```

Listing Data Items in Columnar Format

Once you have found the data items you want to list, you can copy them from the workfile to the output device with the LIST command. This command prints the data items in columnar format and allows you to specify a page heading, which is printed at the top of each page of the report. The syntax of the LIST command is:

LIST ["*heading*"] [,*item list*]

The *heading* is printed at the top of each page in the report. It can consists of at most 80 characters.

The *item list* is a list of the data sets and/or data items to be listed from the workfile. If you do not supply an item list, Query assumes that you want all values in the workfile listed.

The values copied from the workfile are listed in columns. The head of the column lists the data set name and data item name.

For example, in the previous FIND command, values for the OPTION data set with corresponding PRODUCT_NO value of 100 were placed in the workfile. Using the following OUTPUT TO and LIST commands, the sample listing below could be generated:

```
OUTPUT TO DISPLAY RETURN

LIST RETURN
```

```
OPTION      OPTION              OPTION         OPTION        CUSTOMER
ORDER_NO    OPTION_DESC         OPTION_PRICE   OPTION_TYPE   PRODUCT_NO
==========  ==========  ==================  ==========  ==========

101                                     75           0         100
101         Horn                       2.5         569         100
103                                     75           0         100
103         Light                        5         552         100
103         Mud Flaps                 7.25         589         100
103         Horn                        10        6987         100
103         Stripes                    2.5          44         100
103         Fan                         10         459         100
108                                     75           0         100
108         Horn                         5         589         100
113                                     75           0         100
113         Basket                      10           5         100

 THREAD | FIND | SORT BY | BREAK ON | OUTPUT TO | LIST | MORE COMMANDS | EXIT
```

The data items are listed in the same order that they are specified in the *item list* parameter. Or, if a data set is specified (or all items are to be listed), the order of each data item listed in the FIND command or in the schema determines the order they are listed with the LIST command.

If there are more columns than will fit on the output device width, Query wraps the columns around. For example:

```
OPTION      OPTION              OPTION         OPTION        CUSTOMER
ORDER_NO    OPTION_DESC         OPTION_PRICE   OPTION_TYPE   PRODUCT_NO
==========  ==========  ==================  ==========  ==========

CUSTOMER                        CUSTOMER
NAME                            SALESPERSON
==============================  ===========

101                                     75           0         100
Noname, Joseph                  12

101         Horn                       2.5         569         100
Noname, Joseph                  12

103                                     75           0         100
Hernandes, Jose                 56




                                                        Page  1

 CONTINUE |  |  |  |  |  |  | EXIT
```

Another list command, LINEAR LIST, can be used to avoid wrap around.

Listing Data Items in a Linear Format

Query's LINEAR LIST command allows you to copy items from the workfile and list them, one item per line, on the output device. Optionally, you can also supply a page heading with this command, causing the heading to be printed at the top of each page of the report. The syntax for the LINEAR LIST command is:

LINEAR LIST ["*heading*"] [,*item list*]

The *heading* is printed at the top of each page in the report. It can consist of at most 80 characters.

The *item list* is a list of the data sets and/or data items to be listed from the workfile. If you do not supply an item list, Query assumes that you want all values in the workfile listed.

The LINEAR LIST command lists the data items and data sets specified in *item list* in a linear format (one data item per line). For example:

```
OPTION.ORDER_NO = 101
OPTION.OPTION_DESC =
OPTION.OPTION_PRICE = 75
OPTION.OPTION_TYPE = 0
CUSTOMER.PRODUCT_NO = 100
CUSTOMER.NAME = Noname, Joseph
CUSTOMER.SALESPERSON = 12

OPTION.ORDER_NO = 101
OPTION.OPTION_DESC = Horn
OPTION.OPTION_PRICE = 2.5
OPTION.OPTION_TYPE = 569
CUSTOMER.PRODUCT_NO = 100
CUSTOMER.NAME = Noname, Joseph
CUSTOMER.SALESPERSON = 12

OPTION.ORDER_NO = 103
OPTION.OPTION_DESC =

                                                    Page  1
CONTINUE                                                  EXIT
```

The data items are listed in the same order that they are specified in the *item list* parameter. Or if a data set is specified (or all items are to be listed), the order each data item was listed in the FIND command or in the schema determines the order they are listed with the LIST command.

Formatting the Listing

In addition to page headings and a choice between columnar or linear listings, Query provides other commands for controlling the format of a listing:

- The SORT BY command sorts the data to be listed alphabetically or numerically, in either ascending or descending order.
- The TOTAL command computes a total for each numeric data item specified in its parameter list. The total is listed with the data item when the LIST command is executed.
- The BREAK ON command allows you to break a listing into subgroups based on the value of a data item. This command can be used together with the TOTAL command to create subtotals of data items.

Sorting Data Items

Once items have been placed in the workfile with the FIND command, you can sort the items alphabetically or numerically with the SORT BY command. Its syntax is as follows:

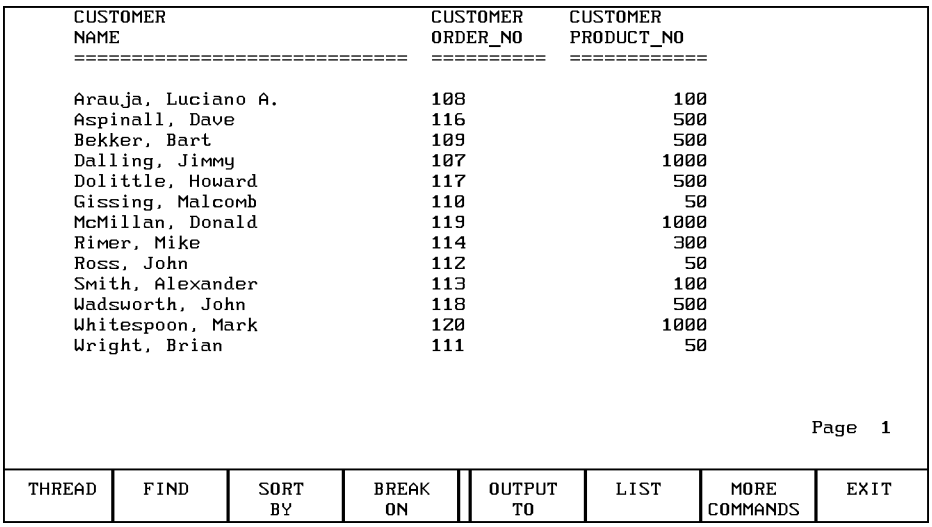
SORT BY *sort list*

The *sort list* is a list of data items to be sorted; items in the list are separated by commas. Items are sorted in ascending order unless the name of an item is followed by a D (specifying to sort in descending order). A maximum of 10 items may be sorted. An array cannot be sorted.

For example, using the sample SAD database, assume that you want to list the CUSTOMER data set in ascending order by NAME, for every customer whose order number is larger than "106"; if a customer has placed more than one order, you want to list these orders in ascending order based on order number. To do this, you would execute the following commands:

```
FIND CUSTOMER FOR CUSTOMER.ORDER_NO>"106"
SORT BY NAME,CUSTOMER.ORDER_NO
LIST NAME,CUSTOMER.ORDER_NO,CUSTOMER.PRODUCT_NO
```

Executing these commands causes the following to be displayed:

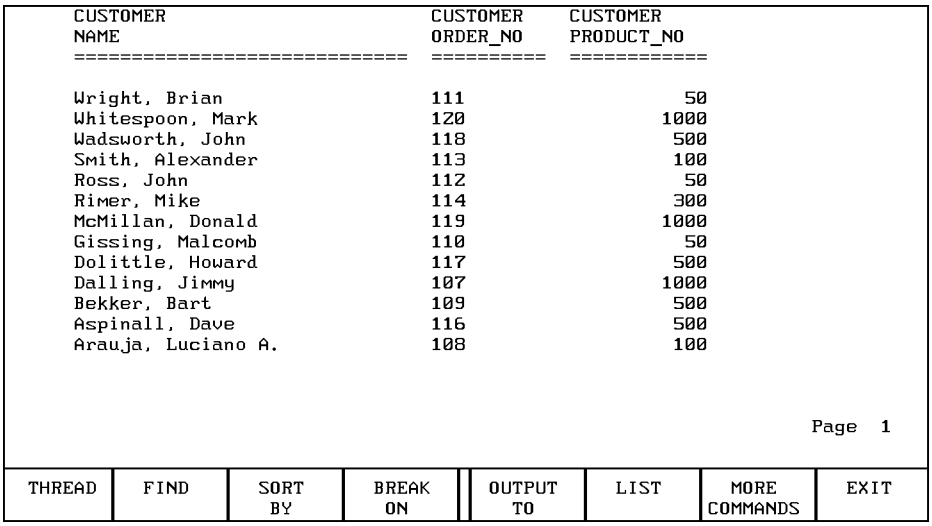


CUSTOMER NAME	CUSTOMER ORDER_NO	CUSTOMER PRODUCT_NO
Arauja, Luciano A.	108	100
Aspinall, Dave	116	500
Bekker, Bart	109	500
Dalling, Jimmy	107	1000
Dolittle, Howard	117	500
Gissing, Malcomb	110	50
McMillan, Donald	119	1000
Rimer, Mike	114	300
Ross, John	112	50
Smith, Alexander	113	100
Wadsworth, John	118	500
Whitespoon, Mark	120	1000
Wright, Brian	111	50

Alternately, assume that you wanted to list the CUSTOMER data set in reverse alphabetic order; like before, if a customer has placed more than one order, list those orders in ascending order based on order number. To do this, you would execute the following commands:

```
FIND CUSTOMER FOR CUSTOMER.ORDER_NO>"106"

SORT BY NAME D,CUSTOMER.ORDER_NO

LIST NAME,CUSTOMER.ORDER_NO,CUSTOMER.PRODUCT_NO
```

This would cause the following to be displayed:



```
     CUSTOMER                       CUSTOMER    CUSTOMER
     NAME                           ORDER_NO    PRODUCT_NO
     ============================   =========   ============

     Wright, Brian                  111                  50
     Whitespoon, Mark               120                1000
     Wadsworth, John                118                 500
     Smith, Alexander               113                 100
     Ross, John                     112                  50
     Rimer, Mike                    114                 300
     McMillan, Donald               119                1000
     Gissing, Malcomb               110                  50
     Dolittle, Howard               117                 500
     Dalling, Jimmy                 107                1000
     Bekker, Bart                   109                 500
     Aspinall, Dave                 116                 500
     Araujo, Luciano A.             108                 100


                                                      Page  1
```

THREAD	FIND	SORT BY	BREAK ON	OUTPUT TO	LIST	MORE COMMANDS	EXIT

Computing with Numeric Data Items

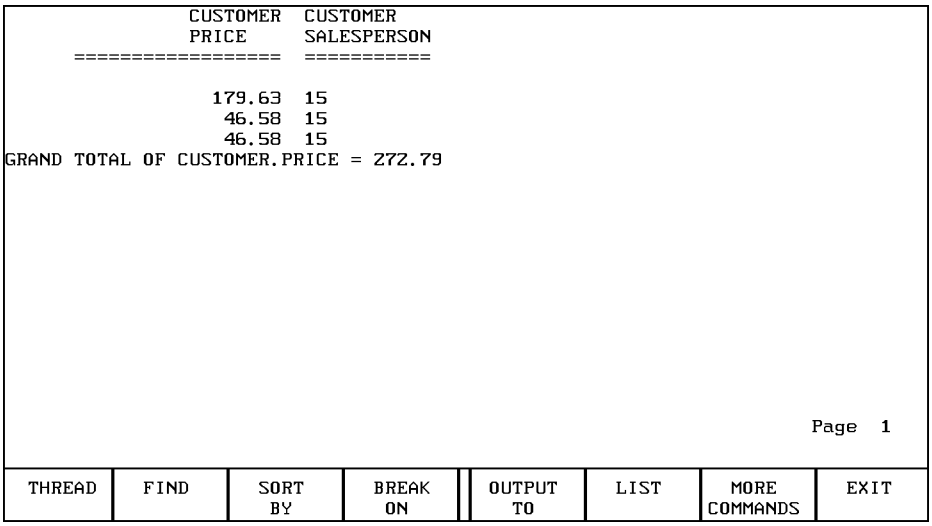
To compute and list the total of a numeric data item, use the TOTAL command. Syntax is as follows:

TOTAL *item list*

The *item list* is a list of the data items you want to be totaled. The items must be a numeric. A maximum of ten items may be totaled, and no arrays may be totaled.

For example, assume you want to know the total dollar amount that one salesperson sold in a certain month. Using the SAD database, you would type the following:

```
FIND PRICE FOR SALESPERSON = ”15”

TOTAL PRICE

LIST
```



While the SORT BY command remains in effect until the next SORT BY command or FIND command is executed, the TOTAL command is in effect only until the next LIST command. Thus, you must re-enter the TOTAL command for every list.

Breaking a Listing into Groups

To break up a list by a change in the value of a single data item, use the BREAK ON command. Syntax is as follows:

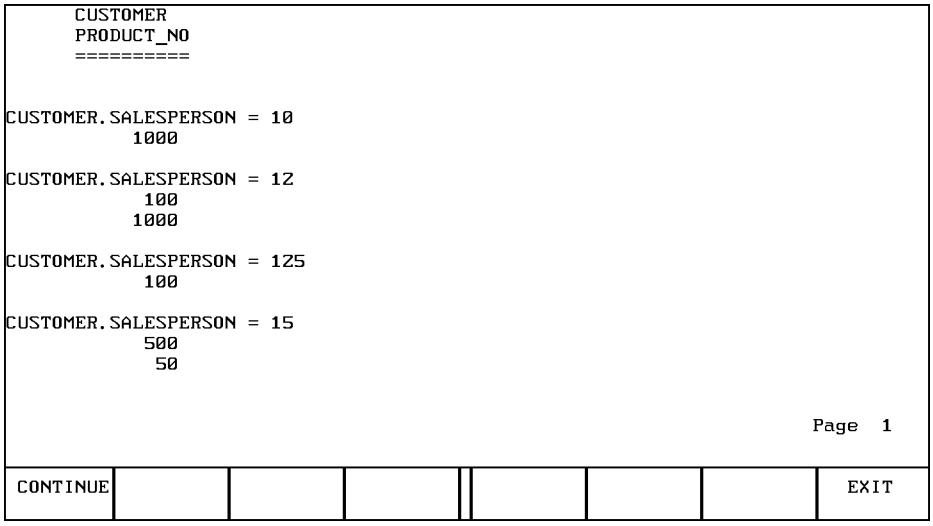
BREAK ON *item*

The data item must be in the workfile.

For example, assume you want a list of the products each salesperson has sold and you want to list the salespeople in groups. You would execute the following commands:

```
SORT BY SALESPERSON

BREAK ON SALESPERSON

LIST CUSTOMER.PRODUCT_NO
```

The following would be displayed:



Even though SALESPERSON was not specified in the LIST command, the breaks still occur. Also note that a sort was done prior to the LIST. If a sort had not been done, the values of SALESPERSON would not have been in order and the breaks would not be what you wanted.

Creating Subtotals within a List

Use of the TOTAL command will produce a grand total. You can also use TOTAL to produce subtotals. Syntax is as follows:

BREAK ON *item* **TOTAL** *item list*

For example, to list the products each salesperson has sold and the total amount used, type the following:

```
SORT BY SALESPERSON

BREAK ON SALESPERSON TOTAL PRICE

LIST CUSTOMER.PRODUCT_NO
```

This would cause the following to be displayed:

```
      CUSTOMER
      PRODUCT_NO
      ==========


CUSTOMER.SALESPERSON = 10
           1000
TOTAL OF CUSTOMER.PRICE = 157.5


CUSTOMER.SALESPERSON = 12
            100
           1000
TOTAL OF CUSTOMER.PRICE = 234.74


CUSTOMER.SALESPERSON = 125
            100


                                                          Page  1
```

CONTINUE							EXIT

As with the BREAK ON item, the data items being totaled do not have to be listed.

Running a Report Program

Using SORT BY and BREAK ON allow some formatting of the lists. If you want editing capabilities and more complex formatting, however, the REPORT WRITER software can be used.

A subprogram can be written using REPORT WRITER statements and accessing the workfile. You run this subprogram from Query with the RUN command. Syntax is as follows:

RUN "*subprogram name* [*volume spec*]" RETURN

NOTE: If you have used the DATABASE command to change databases, you must first execute one of the following commands before running a subprogram with the RUN command: ADD, FIND or INFO. These commands open the database, ensuring that the subprogram can access it.

For example, assume you want to run the subprogram REPORT which is located in the directory whose volume label is “PROGRAMS”. You type the following:

```
RUN ”REPORT, PROGRAMS” RETURN
```

4

Modifying the Database

Query provides several commands that allow you to add new data items, delete data items, and replace data items. You can use these commands *only if* you have supplied a database password that gives you permission to modify the database.

Adding Entries

To add entries to a data set, use Query's ADD command. Its syntax is as follows:

ADD *item list*

If you specify a data set in *item list*, then Query will ask you for each data item in the data set. Entries are added to one data set at a time. For example, suppose that you want to add a new entry to the OPTION data set, which consists of four data items: ORDER_NO, OPTION_DESC, OPTION_PRICE, and OPTION_TYPE. The following shows the initial command executed to add the new entry; it also shows Query's prompts and the supplied responses:

```
ADD OPTION

OPTION
ORDER_NO ?
190
OPTION_DESC ?
Super Tire
OPTION_PRICE ?
18
OPTION_TYPE ?
33

OPTION
ORDER_NO ?
_
```

Query continues to prompt for new entries until the EXIT softkey is pressed. If you press EXIT before you have entered a value for every data in the list, then Query will not add any of the data item values to the database. For example:

```
ADD OPTION

OPTION
ORDER_NO ?
190
OPTION_DESC ?
Super Tire
OPTION_PRICE ?
18
OPTION_TYPE ?
33

OPTION
ORDER_NO ?
200
OPTION_DESC ?
EXIT
_
```

Order number 190 and its associated data items are added; however, order 200 is not added because the EXIT softkey was pressed before values for all four data items were supplied.

You can add specific data items in a data entry. The items not entered are given a null value (0 for numeric items and blank for alphabetic items). For example, the following shows a new addition to the OPTION data set. Notice that while the value “250” is supplied for ORDER_NO, nulls are supplied for the three remaining data items (a null is entered by simply pressing RETURN without typing in a value). The example also shows the result of executing FIND and LIST for this data entry:

```
ADD OPTION

OPTION
ORDER_NO ?
250
OPTION_DESC ?

OPTION_PRICE ?

OPTION_TYPE ?

OPTION
ORDER_NO ?

FIND OPTION FOR OPTION.ORDER_NO="250"
**  1 entry found
LIST

OPTION          OPTION            OPTION             OPTION
ORDER_NO        OPTION_DES        OPTION_PRICE       OPTION_TYPE
========        ==========        ============       ===========

250                               0                  0
```

Later, the REPLACE command can be executed to give OPTION_DESC, OPTION_PRICE, and OPTION_TYPE values. Note that the REPLACE command does not operate on key items. Therefore, you should not enter a null value for a key item.

You can add entries to detail data sets and manual master data sets but entries are added to an automatic master data set for you. Further, if you are adding a new key item to a detail data set and that item is pointed to by a manual master, then you must first add the new key item value to the manual master.

If you make an error in entering a value, you can either press EXIT to terminate the command (if the error was not made on the last data item in the list), or use the REPLACE command to change the value.

Using Forms to Add Entries

If you frequently add the same data items to a database, you may find it more convenient to enter the data items into a standard form. A version of Query's ADD command makes it easy to display a form and then update the database with the data you type on the form:

ADD *item list* **FROM** "*form name* [*volume spec*]"

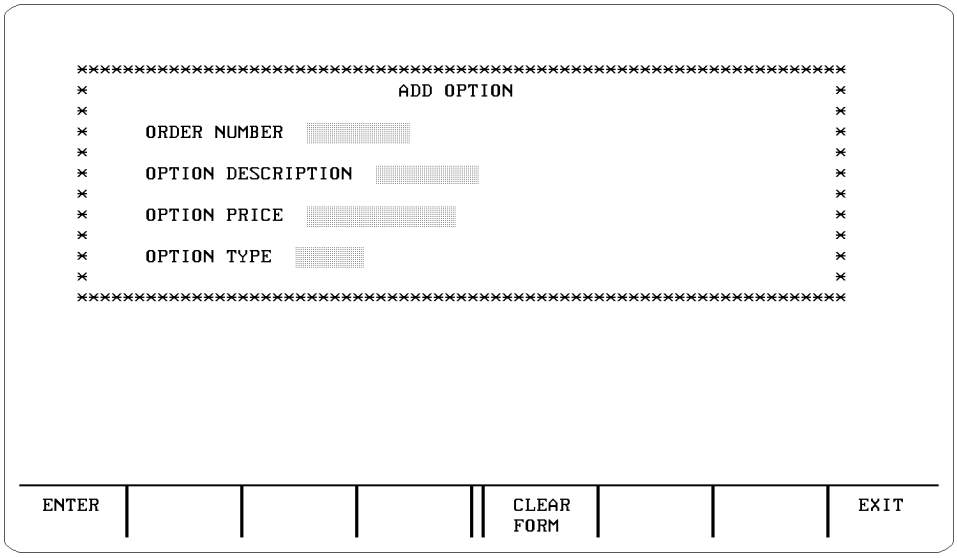
The *item list* is a list of specific data items or data sets to which new values will be added.

NOTE: When using a form to add entries to a database, the number of items in the *item list* must equal the number of input fields on the form. (Arrays require one field for each element.) The form input order must correspond to the order of the data items in the input list.

NOTE: If a data set name is supplied in the *item list*, Query expects an input field on the form for each data item in the data set; again, the form input field order must correspond to the data item order in the data set.

form name is the name of a form file, stored on the volume specified by *volume spec*. The form is created using the Forms software. For information on creating forms, refer to the *Eloquence Forms Manual*.

When the ADD command is used in this way, the specified form is displayed and the softkeys take a new definition. For example:



When the form is displayed, the cursor is placed in the first input field. Type in the correct information for that field; then press RETURN or TAB to move the cursor to the next input field. If you press RETURN at the last input field, Query prompts you to verify that the information typed in is correct. If it is, enter Y (yes). If not, enter N (no) and then re-enter any incorrect value. Once the form is correct and you press RETURN and Y, Query reads the form and adds the data item values to the database. Query then erases the entries, leaving a blank form so that you can continue to update the database. If you make a mistake and want to start over, press the CLEAR FORM softkey.

Pressing the ENTER softkey, at any input field, enters the data into the database without prompting you.

When you have made all the entries, press the EXIT softkey.

Replacing Entries

If you would like to change the value of one or more data items in the database, you must first copy the data item(s) into the workfile using the FIND command. Then, you can execute Query's REPLACE command to modify the value(s):

REPLACE *item list*

The *item list* is a list of data items whose values are to be changed. All the data items must be in the same data set. If a data set name is supplied in *item list*, Query assumes that every data item in the data set is to be modified.

NOTE:

The REPLACE command cannot modify the value of a key item. If the *item list* contains a key item, Query redisplays the command and item list, positioning an error message and pointer under the key item.

One data item at a time, Query displays the name of each data item and that item's current value. To change the data item's value, type the new value and then press RETURN; if you want the data item to retain its current value, simply press RETURN. If while replacing entries you make a mistake and want to start again, press the EXIT softkey. Pressing the EXIT softkey while the REPLACE command is executing causes Query to ignore all modifications to the data entry currently being accessed.

For example, when introducing the ADD command earlier in this section, ORDER_NO 250 was added to the OPTION data set; the values for OPTION_DESC, OPTION_PRICE and OPTION_TYPE were not entered. To change the values from null, first find the entry and then use the REPLACE command, as shown in the following example:

```
FIND OPTION FOR OPTION.ORDER_NO="250"
**  1 entry found.
REPLACE OPTION_DESC,OPTION_PRICE,OPTION_TYPE

OPTION
OPTION_DESC =
Light
OPTION_PRICE =
75
OPTION_TYPE =
0
```

In this example, only one entry was found. Query then prompted for each data item specified with the REPLACE command. If more than one entry would have been found in the workfile, Query would have prompted for each data item of each entry until either all the entries were modified or the EXIT softkey was pressed.

NOTE: If there is more than one data entry in the workfile, the order in which Query prompts for the items is the same as the order in which the data entries were placed in the workfile.

Deleting Entries

If you would like to delete one or more data items, you must first use the FIND command to copy the data item(s) into the workfile. Then you can execute Query's DELETE command to delete the item(s):

DELETE *item list*

The *item list* is a list of data items to be deleted. All data items must be in the same data set. When a data item is deleted, its value is changed to null character(s). If all the data items in a data set are specified in the item list, the entire data entry is deleted; the values of the data items are not simply changed to nulls.

For example, assume the ORDER_NO 25 was cancelled. The following shows how the entry is deleted from the database:

```
FIND CUSTOMER FOR CUSTOMER.ORDER_NO=”25”
**  1 entry found
DELETE CUSTOMER

FIND OPTION FOR OPTION.ORDER_NO=”25”
**  1 entry found
DELETE OPTION
```

NOTE: If one of the entries in a list cannot be deleted, Query issues an error message; it then goes to the next entry and continues deleting.

Deleting Key Items

It is not possible to delete just the key item in a detail data set; the entire data entry must be deleted.

The entries in a manual master data set are deleted in the same manner as deleting an entry from a detail data set. To delete a key item from a manual master data set you must first delete all entries in associated detail data sets that have the same key item value as that of the manual master you wish to delete.

Entries are deleted from an automatic master data set for you.

5

How to Create Files

The QUERY commands DO, RUN and ADD... FROM require a file to be written prior to their execution. The process of creating these files is described here.

Creating a DO File

You create a DO file with the use of a text editor such as vi. Each line in the DO file must be a QUERY command. There is no required first or last line. When control is transferred to the DO file, QUERY executes each command. When an end of file is reached, control transfers back to the operator.

Creating a REPORT Subprogram

The QUERY RUN command is used to start a REPORT subprogram. This section describes how to build them. The REPORT WRITER statements are described in the REPORT WRITER Programming Manual.

The entry point of the subprogram must be:

SUB REPORT (#1,A$, INTEGER B(*),C)

where:

- #1 is the mass memory file number of the workfile.
- A$ is the data base name.
- B is a one-dimension integer array containing the set numbers for the workfile pointers.
- C is the number of sets in B

The subprogram cannot use COM statements. The IMAGE statements necessary to access the data base are described in the IMAGE Programming Manual.

The name of the file where the subprogram is stored is used in the RUN command.

NOTE: No command is shown in the following program for selecting the printer to take the output from the report program. It is recommended that you assign the printer using the OUTPUT command.

Example subprogram

```
! This is a report program to list customer names and addresses
! from the sales analysis data base. Any QUERY THREAD, FIND, or
! SORT (as long as the customer data set is found) may precede
! this program.
!
! variables:
!  Buf$ - a buffer for a customer data set record
!  Db$  - the data base reference number and name
!  i      a loop index
!  Rptr - an array of record pointers from the workfile
!  Sets - an array of sets in the thread used with the FIND
!         command that filled the workfile
!  Slen - the length of the sets array
!  Spos - the position of the customer data set in the thread
!  Sts  - the status array for the DBGET call
!  #l   - the file number for the workfile
!
        SUB Report(#1,Db$,INTEGER Sets(*),Slen)
!
```

```
            INTEGER I,Spos,Sts(1:10)
            SHORT Rptr(1:10)
            DIM Buf$[250]
!
! This is the report description section.
Rpt1:       REPORT HEADER USING Fmt1
Fmt1:          IMAGE 27X, "SALES ANALYSIS DATA BASE"
               PAGE LENGTH 66,2,3
         REPORT TRAILER WITH 2 LINES USING Fmt2;"---- TOTAL CUSTOMERS
PRINTED",VAL$(WFLEN(1))
Fmt2:          IMAGE 10X,K,2X,K
            PAGE HEADER WITH 5 LINES USING Fmt3
Fmt3:          IMAGE 33X, "CUSTOMER LIST", 4/
            PAGE TRAILER WITH 2 LINES USING Fmt4;VAL$(NUMPAGE)
Fmt4:          IMAGE /,70X,"PAGE",K
            END REPORT DESCRIPTION
!
! If no entries in the workfile, error out.
            IF WFLEN(1)<1 THEN
               DISP "--- ERROR...NO ENTRIES HAVE BEEN FOUND"
               GOTO End
            END IF
!
! Set the length of the record pointer array to the number
! of sets in the thread (and hence the number of pointers per
! record in the workfile).
            REDIM Rptr(1:Slen)
!
! Scan the thread for the customer data set (set number 6).
!           Spos=0
            FOR I=1 TO Slen
               IF Sets(I)=6 THEN Spos=I
            NEXT I
!
! If the customer set is not in the thread (and thus not  in the
! workfile), error out.
            IF Spos=0 THEN
               DISP "--- ERROR...CUSTOMER DATA SET NOT FOUND"
               GOTO End
            END IF
!
!  Set an exit softkey and output to the printer.
            ON KEY #8:"EXIT" GOTO End
!
! Rewind the workfile
            READ #1,1
!
!
! This is the report execution section
            BEGIN REPORT Rpt1
!
! This loop prints customer data for every record in the workfile.
            FOR I=1 TO WFLEN(L)
!
! Get a record from the workfile and customer set
               READ #1;Rptr(*)
               DBGET (Db$,6,4,Sts(*),"@ ",Buf$,Rptr(Spos))
!
! If a data base error, error out.
               IF Sts(1) THEN
                  DISP "--- ERROR...DATA BASE ERROR",Sts(1)
```

```
                GOTO End
              END IF
!
! Print the customer name.
              DETAIL LINE 1 WITH 6 LINES
              DETAIL LINE 0 USING "SX,30A";Buf$[11,40]
!
! Print the customer addresses.
              IF TRIM$(Buf$[41,70])<>"" THEN DETAIL LINE 0 USING
 "5X,30A";Buf$[41,70]
              IF TRIM$(Buf$[71,100])<>"" THEN DETAIL LINE 0 USING
 "5X,3OA";Buf$[71,100]
!
! Print the city, state, and zip.
              DETAIL LINE 0 USING "5X,33A";TRIM$(Buf$[101,116])&",
 "&TRIM$(Buf$[117,122])&" "&Buf$[135,142]
!
! Print the customer country and space.
              DETAIL LINE 0 USING "5X,12A";Buf$[123,134]
              DETAIL LINE 0 USING "X"
            NEXT I
!
! End the report
            END REPORT
!
! Done, stop the report and exit
End:        STOP REPORT
         SUBEND
```

Creating a Form

The FORMS Programming Manual describes how to create a form. When QUERY uses the form for adding items, it expects that the number of items in the item list is equal to the number of input fields (arrays require one field for each element) and that the form input order corresponds to the order of the data items in the input list.

The length of the input field must be equal to or less than the length of the data item if the data item is a string. If the data item is numeric, any length field is accepted.

Creating a Workfile

When QUERY creates its own workfile it purges this file before it terminates. The file name is QRwf1x, where x is the user number.

To create a workfile which QUERY will use instead of creating its own, use the CREATE command.

```
CREATE filespec, number of defined records[, record length ]
```

You can create the file any size; however, if QUERY runs out of room when using it an error will occur.

For further information of the size of the workfile refer to Appendix C of the SORT Programming Manual.

Each time QUERY is run it creates two scratchfiles, with names of the following type: QRsc1X and QRsc2X where X 's a letter between A and O.

lf QUERY is allowed to create its own workfile, the name of this workfile will be of the following type: QRwf 1X where X is a letter between A and O.

NOTE

Each time QUERY is exited normally, it will purge all files of either of the above types. Therefore, users must not create files with names beginning with QRwf or QRsc.

6

Control Number

The QUERY control number is specified in the data base schema and is used by QUERY to determine whether an item may be modified and to determine the list format for the item.

Control Number

The QUERY control number is specified in the data base schema and is used by QUERY to determine whether an item may be modified and to determine the list format for the item.

Option	Choice	Value
Write Inhibit	No Inhibit	0
	Write Inhibit	1
Item Type	Default	0
	Date type	2
	Currency Type	4
	Undefined Type	6
Numeric Spacing	Default	0
	Comma Every 3 Digits	8
Digits Right Of Decimal Point	Default	0
	0 digits	16
	1 digits	32
	2 digits	48
	3 digits	64
	4 digits	80
	5 digits	96
	6 digits	112

To find the control number, one value from each option is selected and the values are summed.

For example the data base item TEST-HOURS may be specified as:

2 Digits Right Of Decimal Point	+48
Comma Every 3 Digits	+8
Default Type	+0
QUERY Write Inhibit	+1
Control Number	57

Then in the schema this item is: TEST_HOURS L(57);

WRITE INHIBIT specifies whether or not an operator can make changes to this data item using QUERY-

ITEM TYPE specifies a particular format.

- Date Type - The data item values are input and output as MM/DD/YY, (or DD/MM/YY in the UK and Australia), but are stored differently (refer to next Chapter).
- Currency Type - The data item values are output with a preceding dollar sign, $XXXX. The UK currency is preceded with a pound sign.
- Undefined Type - QUERY ignores the spacing and digits specification for this data item.

Numeric Spacing specifies whether or not commas are inserted when the data item values are output. If commas are specified, they are output after every third digit, counting from the decimal point to the left, x,xxx,xxx.

Digits Right of Decimal Point specifies the number of digits to the right of decimal point to be output. If 5 digits are specified, the number 3.45 is output as 3.45000, and the number 3.451236 is output as 3.45124.

Control Number

7

Date Conversion

Date Conversion

lf a data item has a control number which specifies it to be a date, QUERY uses an algorithm to convert a date format (MM/DD/YY) into an integer for storage. The inverse algorithm is used when the value is output. These algorithms are used to save storage space in the data base and to allow sorting by date. The algorithms are listed below.

```
! This function converts a date string of the form mm/dd/yy, m/dd/yy,
! mm/d/yy, or m/d/yy to an integer value.  The date must be from
! 1/1/72 to 12/31/99 inclusive.  The integer value returned will be
! in the range 1 to 10227.  If an invalid date string is input,
! the return integer value will be 0.
!
! variables:
!  D      - the day value
!  Dt$    - the date string
!  Dval   - the numeric value of the date
!  I      - a loop index
!  M      - the month value
!  Y      - the year value
!
       DEF FNDate(Dt$)
       INTEGER D,Dval,I,M,Y
!
! Check for proper date string length.
         IF (LEN(Dt$)<=8) AND (LEN(Dt$)=6) THEN Cont1
         Dval=0
         GOTO End
!
! Insert leading zeros in month and day, if necessary,
! to get mm/dd/yy.
Cont1:   IF LEN(Dt$)=6 THEN Dt$="O"&Dt$
         IF (LEN(Dt$)=7) AND (Dt$[2,2]="/") THEN Dt$="O"&Dt$
         IF LEN(Dt$)=7 THEN Dt$=Dt$[1,3]&"O"&Dt$[4]
!
! Check for slashes between month, day and year.
         IF (Dt$[3,3]="/") AND (Dt$[6,6]="/") THEN Cont2
         Dval=0
         GOTO End
!
! Check for only digits In month,day and year.
Cont2:   FOR I=1 TO 8
            IF (I=3) OR (I=6) THEN Cont3
            IF POS("1234567890",Dt$[I,I])>0 THEN Cont3
            Dval=0
            GOTO End
!
Cont3:   NEXT I
!
! Convert month, ciay and year to numeric.
         M=VAL(Dt$[1,2])
         D=VAL(Dt$[4,5])
         Y=VAL(Dt$[7,8])+1900
!
```

```
! Compute the numeric value of the date.
          Dval=INT(365.2500*(Y-1972)+0.75)+INT(30.550*M-29.950)
-2*(M>2)+D+((M>2) AND (Y/4=INT(Y/4)))
!
! Convert the numeric value of the date back to a string
! to check for valid date.
Cont4:    IF Dt$<>FNDate$(Dval) THEN Dval=0
End:      RETURN Dva1
       FNEND
!
!
! This function converts a date integer vlaue in the range 1 to 10227
! to a date string of the form mm/dd/yy.  If an invalid date integer
! value is input, the date string will be set to null.
!
! variables:
!  D     - the day value
!  Dt$   - the date string
!  Dval  - the numeric value of the date
!  M     - the month value
!  N     - an intermediate value
!  Y     - the year value
!
       DEF FNDate$(INTEGER Dval)
          INTEGER D,M,N,Y
          DIM Dt$[8]
!
! Check the numeric value for valid range.
          IF (Dval>0) AND (Dval<10228) THEN Cont1
          Dt$=" "
          GOTO End
!
! Convert numeric value of the date to month, day and year.
Cont1:    Y=INT((Dval-L)/365.2500)+1972
          N=Dval-INT(365.2500-(Y-1972)+0.75)
          M=INT((N+31)/30)
         IF INT(30.550*M-29.950)-2*(M>2) AND ((Y/4=INT(Y/4))>N) THEN
 M=M-1
       D=N-INT(30.550*M-29.950)+2*(M>2)-((M>2) AND (Y/4=INT(Y/4)))
!
! Assemble the date string.
!       Dt$=VAL$(M)&"/"&VAL$(D)&"/"&VAL$(Y-1900)
!
! Insert leading zeros in ..nth and day, if "ecessary,
! to get mm/dd/yy.
          IF LEN(Dt$)=6 THEN Dt$="0"&Dt$
          IF (LEN(Dt$)=7) AND (Dt$[2,2]="/") THEN Dt$="0"&Dt$
          IF LEN(Dt$)=7 THEN Dt$=Dt$[1,3]&"0"&Dt$[4]
          RETURN Dt$
       FNEND
```

Alternate Date Conversion

The United Kingdom and Australien date format (DD/MM/YY) is converted using the folowing algorithms.

```
! This function converts a date string of the form dd/mm/yy, d/mm/yy
! dd/m/yy, or d/m/yy to an integer value.  The date must be
! from 1/1/72 to 31/12/99 inclusive.  The integer value returned
! will be in the range 1 to 10227.
! If an invalid date string is input,the return integer value
! will k>e 0.
! variables:
!      D      the day value
!      Dt$    the date string
!      Dval   the numeric value of the date
!      I      a loop index
!      M      the month value
!      Y      the year value
!
       DEF FNDate(Dt$)
          INTEGER D,Dval,I,M,Y
!
! Check for proper date string length.
          IF (LEN(Dt$)<=8) AND (LEN(Dt$)=6) THEN Contl
          Dval=0
          GOTO End
! Insert leading zeros in month and day, if necessary, to
! get mm/dd/yy.
Cont1:    IF LEN(Dt$)=6 THEN Dt$="O"&Dt$
          IF (LEN(Dt$)=7) AND (Dt$[2,2]="/") THEN Dt$="0"&Dt$
          IF LEN(Dt$)=7 THEN Dt$=Dt$[1,3]&"0"&Dt$[4]
!
! Check for slashes between month, day and year.
          IF (Dt$[3,3]="/") AND (Dt$[6,6]="/") THEN Cont2
          Dval=0
          GOTO End
!
! Check for only digits in month,day and year.
Cont2:    FOR I=1 TO 8
             IF (I=3) OR (I=6) THEN Cont3
             IF POS("1234567890",Dt$[I,I])>0 THEN Cont3
             Dval=0
             GOTO End
Cont3:    NEXT I
!
! Convert month, day and year to numeric.
          M=VAL(Dt$[1,2])
          D=VAL(Dt$[4,5])
          Y=VAL(Dt$[7,8])+1900
!
! Compute the numeric value of the date.
          Dval=INT(365.2500*(Y-1972)+0.75)+INT(30.550*M-29.950)
-2*(M>2)+D+((M>2)AND (Y/4=INT(Y/4)))
!
! Convert the numeric value of the date back to a string eck for
! valid date.
```

```
Cont4:    IF Dt$<>FNDate$(Dval) THEN Dval=0
End:      RETURN Dval
       FNEND
!
!
! This function converts a date integer vlaue in the range 1 to 10227
! to a date string of the form mm/dd/yy.  If an invalid date integer
! value is input, the date string will be set to null.
!
! variables:
!  D    - the day value
!  Dt$  - the date string
!  Dval -the numeric value of the date
!  M    - the month value
!  N    - an intermediate value
!  Y    - the year value
!
       DEF FNDate$(INTEGER Dval)
          INTEGER D,M,N,Y
          DIM Dt$[8]
!
! Check the numeric value for valid range.
          IF (Dval>0) AND (Dval<10228) THEN Contl
          Dt$=" "
          GOTO End
!
! Convert numeric value of the date to month, day and year.
Cont1:    Y=INT((Dval-L)/365.2500)+1972
          N=Dval-INT(365.2500-(Y-1972)+0.75)
          M=INT((N+31)/30)
       IF INT(30.550*M-29.950)-2*(M>2) AND ((Y/4=INT(Y/4))>=N) THEN
M=M-L
        D=N-INT(30.550*M-29.950)+2*(M>2)-((M>2) AND (Y/4=INT(Y/4)))
!
! Assemble the date string.
          Dt$=VAL$(D)&"/"&VAL$(M)&"/"&VAL$(Y-1900)
!
! Insert leading zeros in month and day, if necessary, to
! get mm/dd/yy.
          IF LEN(Dt$)=6 THEN Dt$="O"&Dt$
          IF (LEN(Dt$)=7) AND (Dt$[2,2]="/") THEN Dt$="O"&Dt$
          IF LEN(Dt$)=7 THEN Dt$=Dt$[1,3]&"O"&Dt$[4]
End:      RETURN Dt$
       FNEND
```

A

Syntax

RUN "QUERY [*volume spec*]"

Begins Query operation.

ADD *item list* [**FROM** "*form name* [*volume spec*]"]

Adds values to the data items or data set listed. A form can be used to input the values.

[**BREAK ON** *item*] [**TOTAL** *item list*]

Sets report breaks and their associated totals for the LIST or LINEAR LIST commands. TOTAL alone causes a grand total to be printed. A maximum of ten items can be totaled.

DATA BASE *database* [*volume spec*]

Causes future commands to operate on the specified database.

DELETE *item list*

Deletes data item values or entries from the data set. The data items must have been found by the previous FIND command.

DO "*file name* [*volume spec*]"

Transfers control to a file containing Query commands. At the end of the file, control transfers back to operator.

EXIT

Terminates Query.

FIND *item list* **FOR** *search expression*

Finds entries which satisfy the search expression and copies the data items listed in the item list or the entire data entry (if the set name is in the item list) into the workfile.

INFO

Prints information about the database on the current output device.

LINEAR LIST ["*heading*"] [,*item list*]

Lists data items from the workfile in a linear format (one item per line) on the current output device using all BREAKS and TOTALS specified.

LIST ["*heading*"] [,*item list*]

Lists data items from the workfile in columnar format (one data entry per line) on the current output device using all BREAKS and TOTALS specified.

OUTPUT TO *device* [,"*width*" [,"*length*"]]

Changes the output device or spool file for future LIST, LINEAR LIST and INFO commands.

PASSWORD *password*

Defines the password to be used for subsequent commands.

REPLACE *item list*

Replaces values for the data items specified which are in the workfile.

RUN "*subprogram name* [*volume spec*]"

Causes a report subprogram to be run.

SORT BY *item list*

Sorts the entries in the workfile by the data items listed. Data items are sorted in ascending order unless a D follows the data item name.

THREAD *set1, set2* [;*set2, set3 ...*]

Defines the order in which data sets are accessed during the FIND command.

WORKFILE "*file name* [*volume spec*]"

Specifies the workfile to be used for subsequent commands.

B
Error Messages

If an error is made when entering a command, Query points to the error with a ^ sign and outputs a message.

For example:

```
OUTPUT PRINTER
       ^
EXPECTED 'TO'
```

The different error messages that Query might issue are listed alphabetically on the following pages, along with descriptions of the error messages. Where possible, the error message description includes suggestions for corrective actions.

A QUERY FILE IS MISSING FROM THE DIRECTORY

One of the programs or data files that Query uses is not in the /opt/eloquence/share/prog directory.

ABNORMAL QUERY TERMINATION

You have caused Query to abort operation. This may have caused the database to be destroyed. Consult your System Administrator for assistance.

ARITHMETIC OPERATOR IN STRING EXPRESSION

In the FIND search expression a string type data item is being evaluated to a numeric expression (e.g., NAME="1" + "1") or a string is being used in numeric expression (e.g., "ABC" + "3" = PRICE).

ATTEMPTED DIVISION BY ZERO

The divisor in an equation is zero. Check all the values for the data items used in the search expression.

ARRAY ITEM NOT ALLOWED HERE

The data item in the FIND search expression and sort list, as well as the data item in the item list of a BREAK ON or TOTAL command must be a simple item. The INFO command lists all the data items and their types.

CANNOT MODIFY KEY ITEM

A key item value cannot be changed with the REPLACE or DELETE command. You must delete the entire entry and then add it with the change.

CLOSING PARENTHESIS MISSING

You have omitted a closing parenthesis.

CLOSING QUOTE MISSING

You have omitted a closing quotation mark.

DATABASE CORRUPT

The database is not stored correctly. Do not attempt further use of the database. The database needs attention. Consult your system administrator or the *Eloquence DBMS Manual*.

DATABASE NOT CREATED

The database has not been prepared for use. The dbcreate utility must be run (refer to the *Eloquence DBMS Manual*).

DATABASE NOT FOUND OR IN USE

Query cannot open the database, either because it could not find it on the disk or because it is open to another user. Using the CATalogue command, check the disk directory to see if the database is on the disk. Also check that there is no other user.

DATABASE ROOT FILE NOT FOUND

Query cannot find the root (main) file of the database. Check the disk file directory with a CATalog command.

DATA SET IN USE

Someone or some program is using the data set you specified. Query must have exclusive use of the data set.

DISK DIRECTORY OVERFLOW

The file system on the disk is full.

'DO' COMMAND NOT ALLOWED IN A 'DO' FILE

The DO file has a DO command in it. The DO command cannot be stacked.

DUPLICATE KEY ITEM IN MANUAL MASTER

You are attempting to add a key item value which already exists in the manual data set.

EXCEEDS ITEM LIMIT OF 16

Only 16 items can be totaled. Enter the command again limiting the number of items.

EXCEEDS ITEM LIMIT OF 1024

The total number of data items in the FIND item list and search expression cannot exceed 1024.

EXPECTED A COMMA

Check the syntax of the command and then re-enter it correctly.

EXPECTED A DETAIL SET

The THREAD command links detail sets to master sets. Two master sets cannot be directly linked together.

EXPECTED A LOGIC OPERATOR

A logic operator (AND, OR) must be used between expressions in the FIND search expression.

EXPECTED A MASTER SET

The THREAD command links detail sets to master sets. Two detail sets cannot be directly linked together.

EXPECTED A RELATIONAL OPERATOR

A relational operator (>, %<, =, #), is required in each expression of the FIND search expression.

EXPECTED A SET OR ITEM NAME

The value entered for the name of the set or data item is not a valid set or item name.

EXPECTED ’BASE’

The word BASE was omitted from the DATABASE command.

EXPECTED ’BY’

The word BY was omitted from SORT BY command.

EXPECTED END OF LINE

The end of the command was read, but more characters followed. Check the syntax of the command.

EXPECTED ’FOR’

The word FOR was omitted from the FIND command.

EXPECTED ’LIST’

The word LIST was omitted from the LINEAR LIST command.

EXPECTED ’ON’

The word ON was omitted from the BREAK ON command.

EXPECTED ’TO’

The word TO was omitted from the OUTPUT TO command.

EXPECTED ’TOTAL’

Something other than the word TOTAL follows the break item in the BREAK ON command.

EXPRESSION MUST CONTAIN AN ITEM OR ’POS’

The search expression in the FIND command must contain a data item or a POS function.

FATAL COMMAND xxx ENCOUNTERED

An error was found during the execution of the command. The number -xxx- following the message is the error code. Consult your system administrator or - if no cause for the error can be detected - contact Marxmeier support.

FILE NOT FOUND, WRONG TYPE, OR BUSY

The DO file, workfile, RUN program file or FORM file cannot be found or the type is wrong, or some one else is using it. Using the CATalogue command, check the disk directory for file names and file types.

FIND COMMAND TOO LONG

There are too many characters in the command.

INCOMPATIBLE DATABASE VERSION

The database was created on another version of Eloquence and cannot be used on this version.

INPUT EXCEEDS 512 CHARACTERS

A command can consist of at most 512 characters.

INSUFFICIENT DISK SPACE FOR FILES

Query cannot create a workfile, scratch file or spool file because of insufficient room on the specified file system.

INSUFFICIENT MEMORY FOR DATA SET LOCK

The Common Block of memory is full and another data set cannot be locked at this time.

INSUFFICIENT NUMBER OF FIELDS IN FORM

There are not enough fields in the form to enter values for all the data items.

INTERMEDIATE RESULT OVERFLOW

While the search expression was being evaluated, an overflow occurred. The expression must be modified and the FIND command entered again.

INVALID COMMAND

You entered a command unknown to Query. Check that the command is spelled correctly.

INVALID DATABASE NAME

The name you entered cannot be a database name. Check that you have entered it correctly.

INVALID DATA IN 'DO' FILE

The entry in the file is not a valid Query command. This file may not be the DO file. Check the contents of the file.

INVALID DATE

The date must be entered in the format MM/DD/YY for the U.S. or DD/MM/YY for the U.K. The date may be from Jan 1, 1978 to Dec 31, 1999.

INVALID DEVICE SPECIFICATION

The printer number in the OUTPUT TO command has been specified incorrectly, or is not PRINTER, DISPLAY, STDOUT, STDERR or CONSOLE.

INVALID EXPRESSION

The expression in the FIND search expression cannot be used by Query. Refer to page 41 for more information.

INVALID FILE NAME

The name you specified for a file is not a valid disk file name.

INVALID INTEGER PRECISION NUMBER

The data item is integer precision and the value you entered for the data item is not an integer precision number. Integer precision is described in page 97 .

INVALID LENGTH SPECIFICATION

The page length specified in the OUTPUT TO command is not valid. Check that it is numeric and between 20 and 30000.

INVALID LONG PRECISION NUMBER

The data item is long precision and the value you entered for the data item is not a long precision number. Long precision is described in page 97 .

INVALID PASSWORD

The password you specified is not recognized by Query.

INVALID REPORT SUBPROGRAM

The subprogram has a name other then REPORT, the parameters are incorrect, the subprogram is binary, or there is no subprogram in the file specified.

INVALID SHORT PRECISION NUMBER

The data item is short precision and the value entered for the data item is not a short precision number.

INVALID WIDTH SPECIFICATION

The page width specified in the OUTPUT TO command cannot be used. Check that the value is an integer between 20 and 264.

ITEM NOT IN SPECIFIED SET

The data item specified is not in the specified data set.

KEY ITEM POINTS TO DETAIL ENTRY

You are attempting to delete an entry from a manual master which still points to an entry in a detail set.

LIST TITLE EXCEEDS 80 CHARACTERS

The optional heading cannot be more than 80 characters in length.

MORE THAN ONE PATH BETWEEN THESE SETS

You must specify the key data items to show which path to use.

MORE THAN ONE SET NOT ALLOWED HERE

You can only add, delete or replace items in one data set at a time.

NAME EXCEEDS 15 CHARACTERS

The data set or data item name exceeds 15 characters. Use the INFO command to check the schema for the correct name and then re-enter it.

NO ENTRIES HAVE BEEN FOUND

There are no data entries in the workfile.

NO MATCHING KEY ITEM IN MANUAL MASTER

You cannot add a new key item value to a detail data set unless the value already exists in the manual master set which points to it.

NO WRITE ACCESS WITH CURRENT PASSWORD

You cannot add, replace or delete entries in this data set with your password.

NOT A VALID PATH

The only paths are those connecting a detail set and a master set together. The key item must be in both sets.

NUMBER NOT IN QUOTES OR INVALID CHARACTER

You entered a character which Query cannot use (e.g., lowercase or line drawing characters) or there are no quotes around a number. Re-enter the command using only the standard character set (in upper-case).

NUMERIC ITEM OR CONSTANT REQUIRED

A numeric data item is being tested for a string value, or you are adding a string value to a numeric data item.

NUMERIC ITEM REQUIRED

Only numeric data items may be totaled.

OUTPUT DEVICE IS IN USE

Someone else is using the output device. Choose another printer, or use a spooled printer or a spool file for output.

OUTPUT DEVICE NOT AVAILABLE

The device specified is not currently ready for output.

QUERY DISK DIRECTORY OVERFLOW

The file directory is full. Include a different volume specifier with the WORK-FILE command.

REPORT BREAK PREVIOUSLY SET ON THIS ITEM

The data item in the current BREAK ON command was used in a previous BREAK ON command.

SET NAME ONLY NOT ALLOWED HERE

A set name cannot be used alone in the FIND search expression, the BREAK ON list, the TOTAL list, or the SORT BY list.

SET OR ITEM NOT FOUND

Query cannot find the data set or data item name in this database.

SET OR ITEM NOT IN PREVIOUS FIND

The set or item specified in the command is not in the workfile.

SPOOL FILE CANNOT BE EXPANDED

The spool file is full and there is no room on the disk to expand it. Use another for further output.

STRING ITEM OR STRING REQUIRED

A numeric data item or constant is being used where a string is required.

STRING TOO LONG

The string entered is too long for the data item.

THIS SET NOT IN THE SPECIFIED THREAD

One or more of the sets in the FIND command were not in the previous THREAD command.

VOLUME NOT FOUND OR WRITE PROTECTED

The specified volume could not be found or is write protected. Check that the volume is correctly specified.

WORKFILE OR SCRATCH FILE NAME CONFLICT

Query cannot create the workfile or scratch the file because the name it uses has already been used on another file.

WORK RECORD EXCEEDS 256 BYTES

A maximum of 16 data items may be sorted. Further, the total length of the data items plus 2 bytes times the length of the thread must be less than or equal to 256 bytes. How to calculate the length of data items (variables) is described in the section titled “Memory Usage”, in the *Eloquence Manual*.

WRITE TO AUTOMATIC MASTER NOT ALLOWED

You cannot use ADD, DELETE or REPLACE commands on data items in the automatic master.

WRITE TO THIS SET OR ITEM NOT ALLOWED

You cannot use the ADD, DELETE, or REPLACE commands on the item or set specified because the control number is set to write inhibit.

C
Math Operations

Mathematical operations are used in the FIND command’s search expression. The following describes valid operators for mathematical operations in a search expression:

+	addition	/	division
–	subtraction	*	multiplication
AND	logical AND	POS	position function
OR	logical OR	=	equal to
>	greater than	#	not equal to
<	less than	<>	not equal to

Examples

Suppose that you wanted to find data entries where the data item NAME equals "Hewlett-Packard". The search expression to be used with the FIND command would be:

```
NAME = ”Hewlett-Packard”
```

Now suppose that you want to find data entries where the data item NAME equals "Hewlett-Packard" and where the data item ORDER_NO equals "1000". The search expression would be:

```
NAME = ”Hewlett-Packard” AND ORDER_NO = ”1000”
```

If you wanted to find all data entries where the data item NAME does not equal "Hewlett-Packard" or where the data item ORDER_NO equals "1000", you would use the following search expression with the FIND command:

```
NAME # ”Hewlett-Packard” OR ORDER_NO = ”1000”
```

Suppose that you wanted to find all data entries where the data item PRICE is greater than or equal to "1000". The search expression would be:

```
PRICE >= ”1000”
```

Now suppose that have a large database of products and want to find all products where the profit is less than "100". You might use the following search expression:

```
(PRICE - DISCOUNT) < (COST + ”100”)
```

As a final example, suppose that you want to find all data entries where the data item NAME contains "John"; NAME may contain more than "John", but you only want to know if "John" is somewhere in NAME. The search expression would be:

```
POS(NAME,”John”) > ”0”
```

If the string "John" is located anywhere in the value of NAME, POS returns a numeric value indicating the location where "John" begins. Thus if "john" is part of the value of NAME, the value will be greater than 0 and the expression is true.

Index

D