Professional™ 300 series

DATATRIEVE Handbook

digital

digital

Digital Equipment bv
P.O. Box 9064 - 3506 GB Utrecht
The Netherlands

M. SIEBERT

Preface

BEFORE YOU START

This book explains how to use PRO/DATATRIEVE to perform advanced data management tasks and contains reference information on PRO/DATATRIEVE commands, statements, and clauses. If you are not familiar with the basic concepts of data management, or with PRO/DATATRIEVE, or if you have not installed and used PRO/DATATRIEVE, you should read *PRO/DATATRIEVE for Beginners* before reading this book.

You should also be familiar with your Professional 350 computer and with the PROSE editor. If you are not, read your *Professional 300 Series User's Guide for Hard Disk System.*

WHAT'S INSIDE THIS BOOK

This book consists of two parts and three appendixes:

Part I — Chapters 1 through 5 contain information on using PRO/DATATRIEVE to perform advanced data management tasks. Chapter 1 explains how to form and use PRO/DATATRIEVE expressions and chapters 2 through 5 explain how to design efficient records, establish context for PRO/DATATRIEVE statements, retrieve information from list fields, and restructure a data base.

Part II — Chapters 6 through 17 contain reference information on PRO/DATATRIEVE commands, statements, and field definition clauses and many explanatory examples. Language elements are grouped in chapters by function. To use this section, determine what you want to do with PRO/DATATRIEVE, then turn to the appropriate chapter to find information and examples.

Appendix A Contains a listing of record definitions for most records used in examples in this manual, procedure definitions used in examples, a comprehensive report specification, and the resulting report.

Appendix B Lists PRO/DATATRIEVE keywords.

Appendix C Lists the PRO/DATATRIEVE character set used to sort fields.

CONVENTIONS USED IN THIS BOOK

This book uses the following conventions:

- Red Print and Black Print

  In examples, red print indicates what you type. PRO/DATATRIEVE responses and prompts appear in black print.

- DO and RETURN

  You must end all lines that you type by pressing either the DO key or the RETURN key, but examples do not show either key. In text, this book refers to ending lines with the DO key rather than the RETURN key. You can use whichever key is more convenient for you.

- (EXIT)

  This symbol in examples indicates that you should press the EXIT key to exit from a particular PRO/DATATRIEVE process, like the Editor's insert mode or from the Editor itself.

- UPPERCASE/lowercase

  PRO/DATATRIEVE commands and statements appear in text in uppercase letters. In reference section formats, words in uppercase indicate exactly what you must type. You supply names or values for lowercase items.

- [Brackets] and {Braces}

  In reference section syntax formats, brackets ([ ]) and braces ({ }) group portions of a format as a unit. Brackets enclose optional portions of a format and braces enclose required portions. You must include all punctuation as it appears within the brackets or braces. When braces surround a list of items, you can choose only one of the items in the list.

- Ellipsis

  In reference section formats, a horizontal ellipsis (...) indicates that you can repeat the immediately preceding portion of the format. If the ellipsis follows a format unit enclosed in brackets or braces, you can repeat the entire unit. A comma preceding the ellipsis (,...) indicates that you must separate repeated units with a comma.

  In reference section formats, a vertical ellipsis beneath a format line means that you can repeat the immediately preceding line or lines, as explained in the explanation section. In examples in this manual, a vertical ellipsis indicates that lines actually displayed on the screen have been left out.

The reference section of this book describes PRO/DATATRIEVE language elements in the following manner:

- *Format* - Within a box, shows the order of keywords and other components of the command, statement, or clause. The upper portion of the box, if the box is divided, shows the general format of complex statements and commands. The lower portion of the box, if present, shows the format for components of the general format. Circled numbers next to lines in multiline formats serve as reference points for the explanation section.
- *Explanation* - Explains the format, indicates when you can repeat components or choose multiple options, and tells you what PRO/DATATRIEVE does when you use the command, statement, or clause and its components.
- *Comments* - Indicate what you can and cannot do with the command, statement, or clause and suggest how and when to use it.
- *Examples* - Show typical and explanatory examples. Appendix A contains a listing of record definitions used in the examples, as well as a comprehensive report specification and the resulting report.

IF YOU WOULD LIKE TO KNOW MORE

PRO/DATATRIEVE is a subset of DATATRIEVE-11 and this book is a distillation of the DATATRIEVE-11 documentation set. If you want additional information on the topics covered in this book, the following books are available to you:

- *Introduction to DATATRIEVE-11* (AA-X025A-TC)
- *DATATRIEVE-11 User's Guide* (AA-X023A-TC)
- *DATATRIEVE-11 Guide to Writing Reports* (AA-U051A-TC)
- *DATATRIEVE-11 Reference Manual* (AA-U049-TC)
- *DATATRIEVE-11 Pocket Guide* (AA-X103A-TC)

Contents

PART II - FUNCTIONAL REFERENCE SECTION

CHAPTER 6 GETTING HELP AND MONITORING PRO/DATATRIEVE

CHAPTER 7 DEFINING A DATABASE

CHAPTER 8 USING FIELD DEFINITION CLAUSES

CHAPTER 9 READYING AND FINISHING DOMAINS AND RELEASING TABLES

(continued on next page)

TABLES

Part 1

Concepts and Usage Section

This section contains information on PRO/DATATRIEVE concepts and on using PRO/DATATRIEVE to perform advanced data management tasks:

The chapters in this section contain information and many examples to help you use PRO/DATATRIEVE more effectively and efficiently. Most examples are fully explained. Others briefly introduce new commands and statements but do not fully explain them. The Summary section in each chapter provides references to other chapters for more information.

See Appendix A for a listing of the record definitions used in this section.

1

Designing a Useful Database

Chapter 1

Designing a Useful Database

You can design a database in many ways, but not all databases are as useful or as easy to access as others. This chapter introduces some of the techniques you can use to design a database to suit your needs:

- You can create records with *group fields*. A group field contains elementary fields or other group fields. When you refer to a group field, you refer to all the elementary or group fields contained in the group field.
- You can create *COMPUTED BY fields* to save space in records and to keep values up-to-date. PRO/DATATRIEVE does not store values in COMPUTED BY fields. When you refer to a COMPUTED BY field, PRO/DATATRIEVE calculates a current value for the field based on the values specified in the computation.
- You can create *REDEFINES fields* to provide alternate definitions for another field in the record. This lets you access information in a different way.
- You can create *dictionary tables* to validate data, to make storing data easier, and to use as little storage space as possible.
- You can create *list fields* that allow you to store more than one item of information in a field. This chapter explains how to create a list field and then discusses alternatives to list fields. Chapter 4 contains information on working with list fields.

- You can create *view domains* that let you see information in one or more domains in a different way. With views, you can work with subsets of information from one domain or with combined subsets from more than one domain. Views help you access and relate selected information quickly and efficiently.

CREATING AND USING GROUP FIELDS

Records created with ADT contain only elementary fields. Many times, you will find that fields you want to include in a record are related, so you can create a group field to contain those related elementary fields. A group field can even contain another group field:

```
01 CUST-REC.
   15 COMPANY-NAME PIC IS X(20)
      QUERY-NAME IS COMPANY.
   15 ADDRESS.
      20 STREET PIC IS X(15).
      20 CITY PIC IS X(10).
      20 S-Z.
         25 STATE PIC IS X(2).
         25 ZIP-CODE PIC IS X(5)
            QUERY-NAME IS ZIP.
;
```

The CUST-REC record contains COMPANY-NAME, an elementary field, and two group fields, ADDRESS and S-Z. The ADDRESS group field contains the STREET and CITY elementary fields and the S-Z group field. The S-Z group field contains the STATE and ZIP-CODE elementary fields.

When you refer to the ADDRESS field, you refer to all the fields it contains:

```
DTR> READY CUST
DTR> PRINT ADDRESS OF FIRST 2 CUST

                                    ZIP
    STREET          CITY     STATE CODE

43 HIGHLAND AVE BOSTON        MA    13532
9 REDACTOR      DUNKIRK       MA

DTR> PRINT S-Z OF FIRST 2 CUST

       ZIP
STATE CODE

 MA    13532
 MA

DTR>
```

You create group fields by using the DEFINE RECORD command as described in Chapter 7, or you can create the record with ADT and then edit the command file to include group fields.

When you create group fields, be aware of two things:

- The level number of a group field must be lower than the fields it contains. If the STREET field in CUST-REC, for example, had a level number of 18, it would be a group field containing the CITY and S-Z fields, which have level numbers of 20.
- The characteristics of fields within a group field determine the characteristics of the group field. You cannot include any field definition clauses except OCCURS, REDEFINES, and QUERY-NAME in a group field definition. If you include an OCCURS clause, you define the field as a list field. If you include a REDEFINES clause, you define the group field as an alternate definition for another field in the record. List fields and alternate definition fields are described later in this chapter. A QUERY-NAME clause defines an alternate name for the field. A query name is usually shorter than the field name and can save you typing time.

USING COMPUTED BY FIELDS

You create a COMPUTED BY field by including a COMPUTED BY clause in the field definition:

```
15 ON-HAND PIC IS S9(5)
   EDIT-STRING IS -Z(5).
15 UNIT-PRICE PIC IS S9(3)V99
   EDIT-STRING IS $$$$.$$
   QUERY-NAME IS COST.
15 TOTAL-VALUE EDIT-STRING IS $$$,$$$.$$
   QUERY-NAME IS TOTAL
   COMPUTED BY ON-HAND * UNIT-PRICE.
```

This portion of the INVENTORY-REC record defines two elementary fields to contain the number of units on hand and the cost per unit for an inventory item. Because the total value of an inventory is dependent on the quantity on hand and the unit price, the TOTAL-VALUE field is defined as a COMPUTED BY field.

Each time you change an ON-HAND or UNIT-PRICE value for an inventory item, the TOTAL-VALUE for the field changes:

```
DTR> READY INVENTORY WRITE
DTR> FIND INVENTORY WITH ITEM-NAME EQ "WIDGETS"
[1 Record found]
DTR> SELECT
DTR> PRINT

   ITEM        ITEM        ON      UNIT      TOTAL
   NAME       NUMBER      HAND     PRICE     VALUE

WIDGETS     1182413601    2688     $1.11   $2,983.68

DTR> MODIFY UNIT-PRICE
Enter UNIT_PRICE: 1.23
DTR> PRINT

   ITEM        ITEM        ON      UNIT      TOTAL
   NAME       NUMBER      HAND     PRICE     VALUE

WIDGETS     1182413601    2688     $1.23   $3,306.24

DTR>
```

PRO/DATATRIEVE does not store values for a COMPUTED BY field in the physical record, so COMPUTED BY fields save storage space, while also ensuring accurate data.

You can also use COMPUTED BY fields to perform date calculations. Instead of storing a person's age in a field, for example, you can store the birthdate in a field and use that field to calculate the person's current age so that you do not need to continually update records:

```
05 BIRTH-DATE USAGE IS DATE.
05 AGE COMPUTED BY ("TODAY" - BIRTH-DATE)/365.25
   EDIT-STRING IS ZZ9.
```

The value "TODAY" always takes the value of the current date and the 365.25 in the calculation takes into account leap years. Chapter 2 contains more information on using the "TODAY" date value expression.

You can also use COMPUTED BY fields to create edit strings for fields. The following field definitions format the LAST-NAME and FIRST-NAME so that you can display them in different ways:

```
01 WORK-REC.
   05 EMPLOYEE-NAME QUERY-NAME IS NAME.
      10 FIRST-NAME PIC IS X(10)
         QUERY-NAME IS F-NAME.
      10 LAST-NAME PIC IS X(10)
         QUERY-NAME IS L-NAME.
   05 NORMAL-NAME COMPUTED BY FIRST-NAME!!" "!LAST-NAME
      QUERY-NAME IS NORMAL.
   05 ALPHA-NAME COMPUTED BY LAST-NAME!!", "!FIRST-NAME
      QUERY-NAME IS ALPHA.
;
```

By using the NAME group field and the COMPUTED BY fields, you can change the way PRO/DATATRIEVE displays the FIRST-NAME and LAST-NAME fields:

```
DTR> READY WORK
DTR> PRINT NAME, NORMAL, ALPHA OF FIRST 2 WORK

  FIRST        LAST            NORMAL                    ALPHA
   NAME        NAME             NAME                      NAME

JOHN         BROWN       JOHN BROWN              BROWN, JOHN
JOANNE       WILSON      JOANNE WILSON           WILSON, JOANNE

DTR>
```

You can define COMPUTED BY fields by editing a command file created by ADT, or you can define the record yourself using the DEFINE RECORD command as described in Chapter 7. See Chapter 10 for information on editing and executing command files.

CREATING AND USING REDEFINES FIELDS

You create a REDEFINES field by including a REDEFINES clause in the field definition:

```
15 ITEM-NUMBER PIC IS 9(10)
   QUERY-NAME IS NUM.
15 ITEM-NUMBER-PARTS REDEFINES ITEM-NUMBER
   QUERY-NAME IS NUM-PARTS.
   17 PRODUCT-GROUP PIC 99.
   17 PRODUCT-YEAR PIC 99.
   17 ASSEMBLY-CODE PIC 9.
   17 SUP-ASSEMBLY PIC 9(5).
15 ITEM-NUMBER-GROUPS REDEFINES ITEM-NUMBER
   QUERY-NAME IS NUM-GROUPS.
   17 PRODUCT-GROUP-ID PIC 9(4).
   17 PART-DETAIL-ID PIC 9(6).
```

This portion of the INVENTORY-REC record defines the ITEM-NUMBER elementary field as a 10-digit number. The ITEM-NUMBER-PARTS and ITEM-NUMBER-GROUPS group fields redefine the ITEM-NUMBER fields, breaking the 10-digit number into different elementary fields.

You can refer to values stored in the field by the original name, ITEM-NUMBER, or by the REDEFINES field names:

```
DTR> READY INVENTORY
DTR> PRINT NUM, NUM-PARTS, NUM-GROUPS OF FIRST 2 INVENTORY

                                                    PRODUCT   PART
   ITEM     PRODUCT PRODUCT ASSEMBLY    SUP         GROUP   DETAIL
  NUMBER     GROUP    YEAR     CODE    ASSEMBLY      ID       ID

0981414899    09       81        4       14899      0981    414899
0881433332    08       81        4       33332      0881    433332

DTR>
```

You can use a REDEFINES field when storing records to break long numbers into smaller parts that are easier to type. The following procedure uses the ITEM-NUMBER-PARTS definition to prompt for numbers to store in the ITEM-NUMBER field, making it easier to type in correct information. The procedure uses prompting value expressions (*."prompt string") so that PRO/DATATRIEVE prompts with an explanatory message for values. The procedure also uses an ABORT statement in an IF-THEN-ELSE statement to make

sure the PRODUCT-YEAR value is 83. If you type a value other than 83, the ABORT statement stops the procedure and PRO/DATATRIEVE does not store the record:

```
DTR> SHOW INV-STORE
PROCEDURE INV_STORE
READY INVENTORY WRITE
STORE INVENTORY USING
BEGIN
   ITEM_NAME = *.ITEM_NAME
   PRODUCT_GROUP = *."Product group - 2 digits"
   IF *."Product year - 2 digits" NE 83 THEN
      ABORT "Product year must be 83"
   ASSEMBLY_CODE = *."assembly code - 1 digit"
   SUP_ASSEMBLY = *."supervisor ID - 5 digits"
   ON_HAND = *.ON_HAND
   UNIT_PRICE = *.UNIT_PRICE
END
END_PROCEDURE
DTR> :INV-STORE
Enter ITEM_NAME: TANDEMS
Enter Product group - 2 digits: 99
Enter product year - 2 digits: 83
Enter assembly code - 1 digit: 1
Enter supervisor ID - 5 digits: 55555
Enter ON_HAND: 800
Enter UNIT_PRICE: 2.13
DTR>
```

See the next chapter for information on using prompting value expressions in PRO/DATATRIEVE statements and Chapter 13 for information on using the IF-THEN-ELSE and ABORT statements.

CREATING AND USING TABLES

Tables save space in data files and can save time when you store or modify information. A PRO/DATATRIEVE table associates codes with corresponding translations — for example, area codes with states, products with price codes.

One advantage to such code and translation pairs is that you can substitute a short field value for a long one. For example, If you have a table with codes for long job titles, you can store a code, like "E01", in a field and display the longer table translation, like "Tax Assessor First Class."

To define a table, use the DEFINE TABLE command as described in Chapter 7. This example defines the table COMPANIES to contain abbreviations for company names:

```
DTR> DEFINE TABLE COMPANIES
DFN>        "BS"     : "BASIC SYSTEMS",
DFN>        "IA"     : "INFO ANALYSIS",
DFN>        "KS"     : "KEY SPECIALISTS",
DFN>        "LSC"    : "LOGIC SYSTEMS CO.",
DFN>        "PPI"    : "PAPER & PEN, INC.",
DFN>        "PU"     : "PRODUCTS UNLIMITED",
DFN>        "PP"     : "PROSE PROS",
DFN>        "SA"     : "STONE ASSOCIATES",
DFN>        "TA"     : "TACTICAL AID, INC.",
DFN>        "WRA"    : "WRITE RIGHT ASSOC.",
DFN>        ELSE "NOT A VALID CUSTOMER CODE"
DFN> END-TABLE
DTR>
```

The TRANS-REC record uses the COMPANIES table to validate the COMPANY-NAME field before storing data in a record. This is done by including a VALID IF field definition clause in the COMPANY-NAME definition:

```
15 COMPANY-NAME PIC IS X(3)
   VALID IF COMPANY-NAME IN COMPANIES
   QUERY-NAME IS COMPANY.
```

If you try to store a company code that is not in the table, PRO/DATATRIEVE displays an error message:

```
DTR> READY TRANS WRITE
DTR> STORE TRANS
Enter ITEM_NUMBER: 1182413601
Enter TRANS_TYPE: S
Enter TRANS_DATE: 6-6-83
Enter COMPANY_NAME: PAL
Validation error for COMPANY_NAME
Re-enter COMPANY_NAME: EXIT
Execution terminated by operator
DTR>
```

To display the table translations for the company codes stored in the TRANS domain, you need to include a VIA clause and the table name in the PRINT statement, like this:

```
DTR> READY TRANS
DTR> PRINT COMPANY-NAME OF FIRST 2 TRANS

COMPANY
 NAME

  TA
  PPI

DTR> PRINT COMPANY-NAME VIA COMPANIES OF FIRST 2 TRANS

 COMPANY
   NAME

TACTICAL A
PAPER & PE

DTR>
```

Note that PRO/DATATRIEVE displays only the first 10 characters of the translation string. To display more characters in the translation, you need to specify an edit string in a USING clause, like this:

```
DTR> PRINT COMPANY-NAME VIA COMPANIES USING X(20) OF FIRST 2 TRANS

       COMPANY
         NAME

TACTICAL AID, INC.
PAPER & PEN, INC.

DTR>
```

The X in the edit string tells PRO/DATATRIEVE to display the translations as characters. The 20 tells PRO/DATATRIEVE to display up to 20 characters of a translation. Since no translation has more than 20 characters, the X(20) edit string ensures that PRO/DATATRIEVE displays the whole translation string for a company. Specify an edit string that is as long as you need to accommodate the longest translation string in your table.

See Chapter 7 for more information on creating tables and for more examples.

CREATING LIST FIELDS

A list field can contain one or more items. The items in the list can contain one or more fields. A list lets you store more than one value for a field or group of fields (the list item) in one record. Because you can have more than one value in a record for the list element, lists are also called repeating fields.

Lists let you establish a one-to-many relationship between elements of fields. You might, for example, create lists for:

- One team with several players
- One project with several workers
- One employee with several previous jobs
- One supply item with several suppliers
- One supplier with several discount options

Records with lists are called *hierarchical records* because a list field and fields within the list have a one-to-many relationship. Records without list fields are called *flat records* because fields in the record have a one-to-one relationship to each other, based on the order of the record.

You create a list by using the OCCURS clause in the field definition. The SUPPLIES-REC record contains two list fields:

```
DTR> SHOW SUPPLIES-REC
RECORD SUPPLIES_REC
 USING
01 WHAT.
   05 ITEM PIC X(15).
   05 ORDER_UNIT PIC X(10).
   05 HOW_MANY PIC 9.
   05 VENDORS OCCURS 0 TO 10 TIMES DEPENDING ON HOW_MANY.
      07 COMPANY PIC X(10).
      07 PRICE PIC 99V99
         EDIT_STRING IS $$$.$$.
      07 DISCOUNT OCCURS 3 TIMES.
         09 MIN_ORDER PIC 9(5)
            EDIT_STRING IS Z(5).
         09 PERCENT PIC 99
            EDIT_STRING IS Z9%.
         09 WE_PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
            EDIT_STRING IS $$$.$$.
;
DTR>
```

The VENDORS field is a variable-length list that can have as many as 10 items in it. You specify how many items the list contains when you store a value in the HOW-MANY field. The VENDORS list contains two elementary fields, COMPANY and PRICE, and one fixed-length list field, DISCOUNT. The DISCOUNT list field occurs three times and contains three elementary fields, MIN-ORDER, PERCENT, and WE-PAY. See Chapter 8 for information on using the OCCURS clause to define fixed-length and variable-length lists.

When you display a SUPPLIES-REC record, it looks like this:

```
DTR> READY SUPPLIES
DTR> PRINT FIRST 1 SUPPLIES

                ORDER    HOW                          MIN              WE
     ITEM        UNIT    MANY  COMPANY    PRICE    ORDER  PERCENT    PAY

PAPER          REAM       2   ABC SUPPLY $13.45      100      4%   $12.91
                                                      500      5%   $12.77
                                                     1000      7%   $12.50
                                MORRIS     $12.98      10      2%   $12.72
                                                       50      4%   $12.46
                                                      100      6%   $12.20
DTR>
```

The value 2 in the HOW-MANY field indicates that the VENDORS list field can contain information for two suppliers. The DISCOUNT field for each supplier contains three items.

It is easier to work with data stored in flat records, so you should create flat records whenever possible. You can, for example, define a flat supplies record that looks like this:

```
DTR> SHOW FLAT-SUPPLIES-REC
RECORD FLAT_SUPPLIES_REC
 USING
01 WHAT.
   05 ITEM PIC X(15).
   05 ORDER_UNIT PIC X(10).
   05 VENDORS.
      07 COMPANY PIC X(10).
      07 PRICE PIC 99V99
         EDIT_STRING IS $$$.$$.
      07 DISCOUNT.
         09 MIN_ORDER PIC 9(5)
            EDIT_STRING IS Z(5).
         09 PERCENT PIC 99
            EDIT_STRING IS Z9%.
         09 WE_PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
            EDIT_STRING IS $$$.$$.
;
DTR>
```

The information stored in the first SUPPLIES-REC record looks like this when stored in FLAT-SUPPLIES-REC records:

```
DTR> READY FLAT-SUPPLIES
DTR> PRINT FLAT-SUPPLIES WITH ITEM EQ "PAPER"

                ORDER                          MIN              WE
     ITEM        UNIT      COMPANY    PRICE   ORDER  PERCENT   PAY

PAPER           REAM      ABC SUPPLY $13.45    100     4%    $12.91
PAPER           REAM      ABC SUPPLY $13.45    500     5%    $12.77
PAPER           REAM      ABC SUPPLY $13.45   1000     7%    $12.50
PAPER           REAM      MORRIS     $12.98     10     2%    $12.72
PAPER           REAM      MORRIS     $12.98     50     4%    $12.46
PAPER           REAM      MORRIS     $12.98    100     6%    $12.20

DTR>
```

Although it may take longer to store information in the flat record because of the redundancy of the ITEM, UNIT, COMPANY, and PRICE field, the FLAT-SUPPLIES-REC is easier to maintain. The next section in this chapter shows how you can create and use view domains to eliminate redundancy in flat records. With view domains, you can create lists from flat records.

It is fairly simple to modify a PERCENT or MIN-ORDER value in the flat record:

```
DTR> READY FLAT-SUPPLIES WRITE
DTR> MODIFY MIN-ORDER, PERCENT OF FIRST 1 FLAT-SUPPLIES WITH
[Looking for Boolean expression]
CON>      ITEM EQ "PAPER" AND COMPANY CONT "ABC"
Enter MIN_ORDER: 150
Enter PERCENT: 3
DTR>
```

To modify the same fields in the hierarchical record is not so simple. One way to modify fields in list fields involves forming collections and selecting records that contain the fields you want to modify:

```
DTR> READY SUPPLIES WRITE
DTR> FIND FIRST 1 SUPPLIES
[1 Record found]
DTR> SELECT
DTR> PRINT
```

```
                    ORDER     HOW                          MIN              WE
     ITEM            UNIT     MANY   COMPANY     PRICE   ORDER  PERCENT   PAY

PAPER              REAM         2    ABC SUPPLY $13.45     100     4%    $12.91
                                                           500     5%    $12.77
                                                          1000     7%    $12.50
                                     MORRIS       $12.98     10     2%    $12.72
                                                            50     4%    $12.46
                                                           100     6%    $12.20

DTR> FIND FIRST 1 VENDORS
[1 Record found]
DTR> SELECT
DTR> PRINT

                     MIN              WE
 COMPANY    PRICE  ORDER PERCENT   PAY

ABC SUPPLY $13.45    100     4%    $12.91
                     500     5%    $12.77
                    1000     7%    $12.50

DTR> FIND FIRST 1 DISCOUNT
[1 Record found]
DTR> SELECT
DTR> PRINT MIN-ORDER, PERCENT

 MIN
ORDER PERCENT

  100     4%

DTR> MODIFY MIN-ORDER, PERCENT
Enter MIN_ORDER: 150
Enter PERCENT: 3
DTR>
```

In this example, PRO/DATATRIEVE treats the VENDORS and DISCOUNT lists as records within the record, so successive FIND and SELECT statements refine the CURRENT collection until it contains the fields in the list that you want to modify. The MODIFY statement then operates on the specified fields in the selected "record." In this case, the selected record is really a selected list, DISCOUNT.

You can access list items in other ways, as described in Chapter 4. Working with hierarchical records is not as easy as working with flat records, so consider creating list fields carefully.

CREATING AND USING VIEW DOMAINS

A view domain is a special type of domain that is defined in terms of other domains. A view, much like a window, lets you "see" data in a different way. Depending on how you define a view, you can see data in one domain or you can relate information from one domain with information from another domain or domains. You can also specify whether a view uses all the records in a domain or only some of them.

You define a view domain with the DEFINE DOMAIN command, as in this definition of SALES-VIEW:

```
DTR> DEFINE DOMAIN SALES-VIEW
DFN>  OF CUSTOMERS USING
DFN> 01 CUSTOMER OCCURS FOR CUSTOMERS WITH SALES NE 0.
DFN>    03 COMPANY FROM CUSTOMERS.
DFN>    03 LAST-CONTACT FROM CUSTOMERS.
DFN>    03 SALES FROM CUSTOMERS.
DFN> ;
DTR>
```

This DEFINE DOMAIN command names the view (SALES-VIEW) and identifies the domain (CUSTOMERS) that contains the records the view uses. The next line names a view field (CUSTOMER) and identifies the records to include in the view. The next three lines identify the fields from the CUSTOMERS record to include in the view: COMPANY, LAST-CONTACT, and SALES. The semicolon ends the view definition.

You ready the view domain just as you would any other domain. Do not ready the domain the view uses:

```
DTR> READY SALES-VIEW
DTR> PRINT SALES-VIEW

                                    SALES
      COMPANY                LAST   THIS
        NAME                CONTACT YEAR

INFO ANALYSIS              2-Mar-83   $310.00
KEY SPECIALISTS           20-Jun-83 $1,025.00
PAPER & PEN, INC.          9-Jan-83 $2,031.00
PROSE PROS                15-Feb-83 $1,053.00
STONE ASSOCIATES          18-May-83 $2,091.00

DTR>
```

You can read and modify fields in a view. There is no data stored in a view, however, so you cannot store or erase records. A view is only a means of working with data in another domain. When you modify fields in a view, you also modify the record from which the fields came. A view domain can be useful for subsets of information you need to update or analyze frequently.

With views, you can work with subsets of records from one domain or several domains. SALES-VIEW, for example, lets you see readily the current relationship between sales and the last contact date so that you can keep up with your customer contacts.

View domains are also useful for pulling together information from two or more domains. Although PRO/DATATRIEVE lets you define very large records, you are usually better off defining several smaller related records. If you include all the records you need in one large record, you can access any portion of the data by readying only one domain. However, if you need information from only one field, PRO/DATATRIEVE still has to read the whole large record to extract the information you request.

The INVENTORY domain, for example, contains information on inventory stock — the item name and number, the quantity in stock, the item's price, and a COMPUTED BY field that calculates the total value for the quanity in stock. The TRANS domain contains information related to daily sales and orders. Among other things, the TRANS domain stores the item number, the type of transaction (sale or order), the transaction date, the company involved in the transaction, and the number of items sold or ordered.

To keep track of the relationship between the number of items on hand at a given time and the number sold, you can create a view domain to merge information from both domains:

```
DTR> DEFINE DOMAIN INV-VIEW
DFN>  OF INVENTORY, TRANS USING
DFN> 01 INV-DATA OCCURS FOR INVENTORY.
DFN>    03 ITEM-NAME FROM INVENTORY.
DFN>    03 ON-HAND FROM INVENTORY.
DFN>    03 SALES-DATA OCCURS FOR TRANS WITH
DFN>          (ITEM-NUMBER EQ INVENTORY-REC.ITEM-NUMBER) AND
DFN>          (TYPE EQ "S") SORTED BY TRANS-DATE.
DFN>       05 TRANS-DATE FROM TRANS.
DFN>       05 QUANTITY FROM TRANS.
DFN> ;
DTR>
```

This DEFINE DOMAIN command creates a hierarchical domain that lists sales information for each inventory item. You can then ready the view domain and display the merged data:

```
DTR> READY INV-VIEW
DTR> PRINT INV-VIEW

   ITEM       ON      TRANS
   NAME      HAND      DATE    QUANTITY

BAFFLES       2353   7/07/83       121
BALANCES       900
DO DADS        200   6/23/83       500
                     6/24/83       300
                     7/07/83      3000
                     7/11/83      1000
FIXITS        3912   7/18/83        34
GADGETS       1078
GLITCHES       284   3/03/83        95
                     3/04/83       231
HITCHES       1539
TANDEMS        800
THINGS        1000
WIDGETS       2688   3/04/83       344
                     7/01/83        50

DTR>
```

From this display, you can see that the DO DADS inventory is running low and that sales activity for the product has been particularly heavy recently. You can also see what items are not selling very well. Chapter 7 shows how to modify this view and write a procedure so that you can specify a starting date for records included in the view.

A view domain can provide you with the benefits of list fields without the complications involved in accessing list fields. The records used by the INV-VIEW view domain are flat records, so you can easily modify and change data in those records. Then, by using the view domain, you can display the information you want in list format.

If you want to work with list field values contained in a view domain, you must use one of the methods described in Chapter 4.

SUMMARY

This chapter introduced you to techniques you can use to design a useful and efficient database:

- You can use group and REDEFINES fields to help simplify the task of accessing data.
- You can use tables and COMPUTED BY fields to save storage space and typing time.
- You can use hierarchical records to store information in lists.
- You can use view domains to access subsets of information from one or more domains and to create list displays from flat records.

For more information on clauses and statements used in this chapter, refer to the following chapters:

- See Chapter 2 for information on forming and using concatenated expressions, prompting value expressions, and date value expressions.
- See Chapter 4 for information on working with list fields.
- See Chapter 7 for information on defining records, tables, and view domains.
- See Chapter 8 for information on field definition clauses and group fields.
- See Chapter 10 for information on editing and executing command files.
- See Chapter 13 for information on BEGIN-END, IF-THEN-ELSE, and ABORT statements.

2

Creating and Using PRO/DATATRIEVE Expressions

Chapter 2

Creating and Using PRO/DATATRIEVE Expressions

You use value expressions, Boolean expressions, and record selection expressions (RSEs) every time you tell PRO/DATATRIEVE what you want to print, store, modify, or find:

- A *value expression* consists of characters and symbols that specify a value for PRO/DATATRIEVE to use when executing statements. PRINT "HI", PRINT COMPANY-NAME, and PRINT TOTAL SALARY all contain value expressions to tell PRO/DATATRIEVE what to print.
- A *Boolean expression* consists of value expressions, relational operators, and Boolean operators and specifies a relationship between value expressions for PRO/DATATRIEVE to evaluate when executing statements. PRO/DATATRIEVE evaluates Boolean expressions as either true or false.
- A *record selection expression* (RSE) consists of Boolean expressions and other clauses that identify the records you want to include in a record stream or a collection.

This chapter provides details on forming and using these expressions.

CREATING AND USING VALUE EXPRESSIONS

PRO/DATATRIEVE recognizes the following types of value expressions:

- Literal values
- Variables
- Field names
- Date value expressions
- Prompting value expressions
- Table value expressions
- Statistical expressions
- Arithmetic expressions
- Concatenated expressions

Literal Values

The simplest way to specify a value in PRO/DATATRIEVE is with a *literal*. There are two types of literal:

- A character string literal can contain up to 130 printing characters or spaces. It must be enclosed in quotation marks. Printing characters consist of uppercase and lowercase letters, numbers, and the following special characters:

  ! @ # $ % ^ & * ( ) - _ = + ‘ [

  {] } ~ ; : ’ ” \ | , < . > / ?

- A numeric literal can consist of up to 18 digits and an optional decimal point. A numeric literal cannot begin or end with a decimal point. For example, .5 and 123. are not valid numeric literals, but 0.5 and 123.0 are valid. The data type of a field or variable determines the maximum numeric literal you can assign to the field or variable.

The following example uses both a character string literal and a numeric literal:

```
DTR> PRINT "Hello", 21
Hello  21

DTR>
```

You can continue a character string literal on another line by typing a hyphen (-) and pressing DO or RETURN. PRO/DATATRIEVE discards the hyphen and waits for you to complete the literal by typing the closing quotation mark on a subsequent line. As long as the total number of characters in the literal does not exceed 130, you can use any number of continuation characters between the quotation marks.

You can create a character string literal longer than 130 characters by forming a concatenated expression, as described in a later section of this chapter.

To include quotation marks in a character string literal, type two consecutive quotation marks for every one you want to include in the literal. For example:

```
DTR> PRINT "They said, ""We're going."""
They said, "We're going."

DTR>
```

Although PRO/DATATRIEVE usually converts all lowercase letters that you type to uppercase, lowercase letters in character string literals remain lowercase. Because all PRO/DATATRIEVE relational operators except CONTAINING are case-sensitive, you should use uppercase letters in RSEs so that PRO/DATATRIEVE accurately identifies the records you want. For example:

```
DTR> READY EMPLOYEES
DTR> FIND EMPLOYEES WITH LAST-NAME EQ "BOOLE"
[4 records found]
DTR> FIND EMPLOYEES WITH LAST-NAME EQ "Boole"
[0 records found]
DTR> FIND EMPLOYEES WITH LAST-NAME CONT "Boole"
[4 records found]
DTR>
```

Variable Values

A variable is a type of field. You name and define a variable with the DECLARE statement. Then you can use the variable as a value expression. When you want to change the value of a variable, you simply assign another value to it. For example:

```
DTR> DECLARE A PIC XX.
DTR> DECLARE B PIC XX.
DTR> A = 15
DTR> B = 25
DTR> PRINT A

A

15
```

(continued on next page)

```
DTR> PRINT B

B

25

DTR> PRINT A * B
  375

DTR>
```

Unlike a field, a variable exists only temporarily. A variable declared within a BEGIN-END block or a THEN statement (a *local* variable) disappears when the BEGIN-END block or THEN statement finishes executing. A variable not declared in a BEGIN-END block or THEN statement (a *global* variable) and not defined as a COMPUTED BY variable retains its value until you assign it a new value, release it with the RELEASE command, or exit from PRO/DATATRIEVE.

The value of a variable defined with a COMPUTED BY clause depends on the value expression that controls the computation. The following example defines a global variable to calculate a 15% salary increase:

```
DTR> DECLARE NEW-SAL COMPUTED BY SALARY * 1.15
CON>     EDIT-STRING IS $$$,$$$,$$$.
DTR> PRINT SALARY, NEW-SAL OF EMPLOYEES WITH DEPT CONT "LIT"

              NEW
SALARY        SAL

$29,908       $34,394
$32,918       $37,855
$32,000       $36,800
$54,000       $62,100
$23,908       $27,494
$55,407       $63,718
$34,125       $39,243

DTR>
```

A variable declared in a BEGIN-END or THEN statement is local to that statement. A BEGIN-END statement contains other statements and forms a BEGIN-END block. PRO/DATATRIEVE treats statements in a BEGIN-END block as one statement and executes each statement in sequential order. The THEN statement joins two or more statements into a compound statement that executes as a single statement.

Because BEGIN-END and THEN statements form a single unit consisting of other statements, a local variable has no meaning for any statements outside the BEGIN-END or THEN statement that declares the variable. The SET NO PROMPT command tells PRO/DATATRIEVE not to display helpful messages like "[Looking for statement]" for continuation lines so that you can see the structure of the BEGIN-END statements more clearly:

```
DTR> SET NO PROMPT
DTR> BEGIN
CON>    DECLARE X PIC XXX.
CON>    X = "TOP"
CON>    PRINT X
CON>    BEGIN
CON>       DECLARE X PIC 9.99.
CON>       X = 1.23
CON>       PRINT X
CON>    END
CON>    PRINT X
CON> END

 X

TOP

 X

1.23

 X

TOP

DTR> PRINT X
Field "X" is undefined or used out of context
DTR>
```

In this example, the variable X declared in the inner BEGIN statement does not affect the variable X declared in the outer BEGIN statement, even though the inner X has a different data type and value. Neither local variable exists when PRO/DATATRIEVE completes the execution of the compound statements containing them.

See Chapter 13 for information on forming compound statements with BEGIN-END and THEN statements and Chapter 15 for information on creating and using variables.

Field Name Values

When you use the name or query name of an elementary or group field in an expression, PRO/DATATRIEVE uses the value stored in that field:

```
DTR> SET NO PROMPT
DTR> READY INVENTORY
DTR> PRINT ITEM-NUMBER OF INVENTORY WITH
CON>      ITEM-NAME EQ "BAFFLES"

   ITEM
  NUMBER

0981414899

DTR> PRINT NAME, NUM-PARTS OF INVENTORY WITH
CON>      NUM-PARTS CONT 81

   ITEM     PRODUCT PRODUCT ASSEMBLY    SUB
   NAME      GROUP    YEAR     CODE    ASSEMBLY

BAFFLES       09       81        4       14899
BALANCES      08       81        4       33332
HITCHES       11       81        4       13602

DTR>
```

In the first PRINT statement, ITEM-NAME identifies an elementary field. In the second PRINT statement, NUM-PARTS is the query name for the ITEM-NUMBER-PARTS group fields that redefines the ITEM-NUMBER field. The record definition for these fields looks like this:

```
15 ITEM-NAME PIC IS X(10)
   QUERY-NAME IS NAME.
15 ITEM-NUMBER PIC 9(10)
   QUERY-NAME IS NUM.
15 ITEM-NUMBER-PARTS REDEFINES ITEM-NUMBER
   QUERY-NAME IS NUM-PARTS.
   17 PRODUCT-GROUP PIC 99.
   17 PRODUCT-YEAR PIC 99.
   17 ASSEMBLY-CODE PIC 9.
   17 SUB-ASSEMBLY PIC 9(5).
```

See Chapter 8 for more information on using the REDEFINES clause.

If you specify the name of a field defined with a COMPUTED BY clause, PRO/DATATRIEVE calculates the value for the field and uses that value in the statement.

You can also use context variables (names for record streams), collection names, and domain names to qualify field names in value expressions. See Chapter 3 for information on using context variables and qualifying field names.

Date Value Expressions

You can assign the current date to a date field with the PRO/DATATRIEVE date value expression, "TODAY":

```
DTR> DECLARE X USAGE DATE.
DTR> X = "TODAY"
DTR> PRINT X

     X

 23-Jul-83

DTR>
```

You can also use the date value expression to assign the current date to a field or variable by typing TODAY in response to the prompt for a date value:

```
DTR> SHOW SALES-REC
RECORD SALES_REC
 USING
01 SALES_REC.
   03 SALE_DATE USAGE IS DATE.
   03 ITEM_NAME PIC X(10).
   03 QUANTITY PIC 9(5).
;
DTR> READY SALES WRITE
DTR> STORE SALES
Enter SALE_DATE: TODAY
Enter ITEM_NAME: WIDGETS
Enter QUANTITY: 50

DTR> PRINT SALES WITH SALE-DATE EQ "TODAY"

   SALE          ITEM
   DATE          NAME     QUANTITY

23-Jul-83     WIDGETS       00050

DTR>
```

You can also subtract dates. You might, for example, want to know how many days you have to complete a project. Define a variable for the project due date, then subtract today's date from the project due date, like this:

```
DTR> DECLARE PROJECT-DUE USAGE IS DATE.
DTR> PROJECT-DUE = "21-AUG-83"
DTR> PRINT (PROJECT-DUE - "TODAY")
                        29

DTR>
```

Prompting Value Expressions

To make your procedures more versatile, you can have PRO/DATATRIEVE prompt you for a value to use when executing statements. Do this by using a prompting value expression:

*."prompt-string"

or

**."prompt-string"

Prompt-string is a character string literal. If the prompt string contains no blanks and conforms to the rules for naming PRO/DATATRIEVE fields, you do not have to enclose it in quotation marks. Otherwise, you must enclose it in quotation marks.

If you put a *.prompt prompting value expression in a REPEAT loop or a FOR loop, PRO/DATATRIEVE prompts you for a value each time it executes the loop.

If you put a **.prompt prompting value expression in a REPEAT loop or a FOR loop, PRO/DATATRIEVE prompts you for a value only once, the first time it executes the loop. If the **.prompt prompting value expression assigns a value to a variable or a field, PRO/DATATRIEVE uses the same value each time through the loop. This feature is especially useful when storing or modifying a group of records that have a common value in one or more fields. See Chapter 13 for information on using REPEAT and FOR statements.

The following example shows the difference between the *.prompt and **.prompt:

```
DTR> SET NO PROMPT
DTR> READY SALES WRITE
DTR> REPEAT 3 STORE SALES USING
CON> BEGIN
CON>   SALE-DATE = **."SALE DATE"
CON>   ITEM-NAME = *."ITEM NAME"
CON>   QUANTITY = *.QUANTITY
CON> END
Enter SALE DATE: TODAY
Enter ITEM NAME: BAFFLES
Enter QUANTITY: 50
Enter ITEM NAME: GLITCHES
Enter QUANTITY: 100
Enter ITEM NAME: HATCHES
Enter QUANTITY: 2000
DTR> PRINT SALES WITH SALE-DATE EQ "TODAY"

   SALE          ITEM
   DATE          NAME      QUANTITY

5-Jul-83     GLITCHES      00100
5-Jul-83     HATCHES       02000
5-Jul-83     WIDGETS       00050

DTR>
```

The first time PRO/DATATRIEVE executes the BEGIN statement, it prompts for all three values. The second time, it prompts only for the ITEM-NAME and QUANTITY values and assigns the current date to all three records. Note that the SET NO PROMPT command does not affect the prompting value expressions. SET NO PROMPT suppresses only PRO/DATATRIEVE's helpful prompts, such as "[Looking for statement]".

The *.prompt prompting value expression is particularly useful in the PRINT and REPORT statements. You can include a prompting value expression such as *."TI: for screen or a file name" to have PRO/DATATRIEVE prompt you for a file name or the device name (TI:) of your Professional video screen. If you respond to the prompt with TI:, PRO/DATATRIEVE displays information on your video screen. To print information to a file, respond with a file name.

Table Value Expressions

To use a value stored in a table, specify the table name in a VIA clause after the value you want translated. The value expression that precedes the VIA clause determines the value of the table value expression. For example:

```
DTR> SHOW COMPANIES
TABLE COMPANIES
    "BS"     : "BASIC SYSTEMS",
    "IA"     : "INFO ANALYSIS",
    "KS"     : "KEY SPECIALISTS",
    "LSC"    : "LOGIC SYSTEMS CO.",
    "PPI"    : "PAPER & PEN, INC.",
    "PU"     : "PRODUCTS UNLIMITED",
    "PP"     : "PROSE PROS",
    "SA"     : "STONE ASSOCIATES",
    "TA"     : "TACTICAL AID, INC.",
    "WRA"    : "WRITE RIGHT ASSOC.",
    ELSE "NOT A VALID CUSTOMER CODE"
END_TABLE
DTR> READY TRANS
DTR> PRINT COMPANY OF FIRST 1 TRANS

COMPANY
 NAME

  TA

DTR> PRINT COMPANY VIA COMPANIES USING X(20) OF FIRST 1 TRANS

      COMPANY
        NAME

TACTICAL AID, INC.

DTR>
```

This example includes a USING clause and an edit string clause that tell PRO/DATATRIEVE to display up to 20 characters of the table translation. If you do not specify an edit string, PRO/DATATRIEVE displays only the first 10 characters of a table translation.

PRO/DATATRIEVE displays the table translation if the specified value expression is stored as a code in the table. If the table contains an ELSE clause and

the value expression you specify does not match any code in the table, PRO/DATATRIEVE displays the value of the ELSE clause:

```
DTR> SHOW COMPANIES
TABLE COMPANIES
    "BS"     : "BASIC SYSTEMS",
    "IA"     : "INFO ANALYSIS",
    "KS"     : "KEY SPECIALISTS",
    "LSC"    : "LOGIC SYSTEMS CO.",
    "PPI"    : "PAPER & PEN, INC.",
    "PU"     : "PRODUCTS UNLIMITED",
    "PP"     : "PROSE PROS",
    "SA"     : "STONE ASSOCIATES",
    "TA"     : "TACTICAL AID, INC.",
    "WRA"    : "WRITE RIGHT ASSOC.",
    ELSE "NOT A VALID CUSTOMER CODE"
END_TABLE
DTR> PRINT "WW" VIA COMPANIES USING X(30)
NOT A VALID CUSTOMER CODE
DTR>
```

If the table does not contain an ELSE clause and the value expression you specify does not match any code in the table, PRO/DATATRIEVE tells you the value was not found:

```
DTR> SHOW AREA-TABLE
TABLE AREA_TABLE
"603" : "NEW HAMPSHIRE",
"617" : "MASSACHUSETTS",
"802" : "VERMONT"
END_TABLE
DTR> PRINT "999" VIA AREA-TABLE
Value not found in table "AREA_TABLE"
Execution failed
DTR>
```

See Chapter 7 for more information on creating and using tables.

Statistical Expressions

Statistical expressions consist of PRO/DATATRIEVE statistical function keywords and value expressions. Table 2-1 lists PRO/DATATRIEVE statistical functions and the values they compute.

Table 2-1: Statistical Functions and Results

Function	Result
AVERAGE	Averages the values of the expression
COUNT	Counts the number of records in a collection or record stream
MAX	Identifies the largest value of the expression
MIN	Identifies the smallest value of the expression
TOTAL	Totals the values of the expression

Except for TOTAL values, PRO/DATATRIEVE uses the edit string specified in the EDIT-STRING clause of a field definition when printing a statistical value for a field. To specify an edit string for a TOTAL value, include a USING clause in the PRINT statement:

```
DTR> SET NO PROMPT
DTR> PRINT TOTAL SALARY OF EMPLOYEES WITH
CON>    DEPT CONT "MATH" USING $$$,$$$,$$$

  SALARY

   $219,718

DTR>
```

With the exception of the COUNT function, you must specify a value expression for the statistical function. The value expression in a statistical expression is usually a field name or the name of a variable:

```
DTR> READY EMPLOYEES
DTR> PRINT MAX SALARY OF EMPLOYEES

SALARY

$75,892

DTR>
```

When the value expression is more complex than a field name or variable name, enclose the expression in parentheses to avoid unexpected results.

The COUNT function counts the number of records in a record stream. It does not operate on a value expression, such as a field name, as other statistical functions do:

```
DTR> READY EMPLOYEES
DTR> PRINT COUNT OF EMPLOYEES WITH DEPT CONT "MATH"
    8

DTR> PRINT AVERAGE SALARY OF EMPLOYEES WITH DEPT CONT "MATH"

SALARY

$27,464

DTR>
```

If you do not specify an RSE in a statistical expression, the function keyword operates on all records in the CURRENT collection. If you do not specify an RSE and have no CURRENT collection, PRO/DATATRIEVE displays the message "A current collection has not been established".

When you use two or more statistical expressions in the same statement, specify an RSE in each expression to identify the record stream on which the function is to operate. If you do not, your results may be misleading. In the following example, PRO/DATATRIEVE displays the average SALARY for the whole CURRENT collection, and the total SALARY for employees in the mathematics department. The RSE for the AVERAGE function in the second PRINT statement tells PRO/DATATRIEVE to compute the average salary for only employees in the mathematics department, thus producing a different average:

```
DTR> SET NO PROMPT
DTR> FIND EMPLOYEES
[34 records found]
DTR> PRINT AVERAGE SALARY, MAX SALARY OF EMPLOYEES WITH
CON>    DEPT CONT "MATH"

SALARY     SALARY

$35,281    $59,594

DTR> PRINT AVERAGE SALARY OF EMPLOYEES WITH DEPT CONT "MATH",
CON>    MAX SALARY OF EMPLOYEES WITH DEPT CONT "MATH"

SALARY     SALARY

$27,464    $59,594

DTR>
```

Arithmetic Expressions

An arithmetic expression consists of numeric value expressions and arithmetic operators. PRO/DATATRIEVE recognizes four arithmetic operators. Table 2–2 shows these arithmetic operators and the operation each performs.

Table 2-2: Arithmetic Operators

Operator	Operation
+	addition
-	subtraction
*	multiplication
/	division

You must separate the minus operator from its operands with spaces so that PRO/DATATRIEVE does not interpret the minus sign as a hyphen. If one or both the operands is a field or variable name and you do not include spaces, PRO/DATATRIEVE interprets the minus sign as a hyphen and converts it to an underscore:

```
DTR> DECLARE X PIC 99.
DTR> X = 8
DTR> !
DTR> ! Print X -1 with a space before the minus sign
DTR> !
DTR> PRINT X -1
   7

DTR> !
DTR> ! Print 1-X without a space before the minus sign
DTR> !
DTR> PRINT 1-X
 -7

DTR> !
DTR> ! Print X-1 without a space before the minus sign
DTR> !
DTR> PRINT X-1
"Field X_1" is undefined or used out of context
DTR>
```

You can control the order in which PRO/DATATRIEVE performs arithmetic operations by using parentheses. PRO/DATATRIEVE evaluates arithmetic expressions in the following order:

1. PRO/DATATRIEVE evaluates any value expressions in parentheses.
2. PRO/DATATRIEVE performs multiplications and divisions from left to right in the arithmetic expression.
3. PRO/DATATRIEVE performs additions and subtractions from left to right in the arithmetic expression.

The following examples show how parentheses affect the evaluation of arithmetic expressions:

```
DTR> PRINT (6 * 7) + 5
  47

DTR> PRINT 6 * (7 + 5)
  72

DTR> PRINT 12 - 6 / 2
     9.000

DTR> PRINT (12 - 6) / 2
      3.000

DTR>
```

Concatenated Expressions

Concatenated expressions are value expressions joined together with one or two vertical bars (| or | |), called *concatenation symbols*:

```
DTR> PRINT LAST-NAME|:", "|FIRST-NAME OF FIRST 2 EMPLOYEES
BOOLE, GEORGE
SPIVA, CHARLOTTE

DTR>
```

When PRO/DATATRIEVE encounters a concatenation symbol, it converts the value expressions on either side of the symbol to character string literals. PRO/DATATRIEVE then joins the literals to form a longer literal.

The type of symbol you use determines how PRO/DATATRIEVE treats trailing spaces in the first literal:

- When you use the single bar (|), PRO/DATATRIEVE retains trailing spaces in the first literal:

```
DTR> PRINT "ABC   "|"DEF"
ABC   DEF

DTR>
```

- When you use the double bar (| |), PRO/DATATRIEVE suppresses trailing spaces in the first literal:

```
DTR> PRINT "ABC   "||"DEF"
ABCDEF

DTR>
```

Both the single bar (|) and the double bar (| |) retain leading spaces in the second literal:

```
DTR> PRINT "ABC"|"   DEF"
ABC   DEF

DTR> PRINT "ABC"||"   DEF"
ABC   DEF

DTR>
```

You can use concatenated expressions to form neat displays, as in the following example:

```
DTR> SET NO PROMPT
DTR> READY CUSTOMERS
DTR> PRINT SKIP, COMPANY(-), SKIP, CONTACT(-), SKIP, STREET(-),
CON>     CITY||", "|STATE|", "|ZIP OF FIRST 2 CUSTOMERS

        BASIC SYSTEMS
        ANNE DUGGAN
        43 HIGHLAND AVE
        BOSTON, MA, 13532

        INFO ANALYSIS
        WILMA SWAYER
        9 REDACTOR
        DUNKIRK, MA,

DTR>
```

This example uses SKIP keywords to put the COMPANY, CONTACT, and STREET fields on separate lines and concatenation symbols to format the CITY, STATE, and ZIP-CODE fields as one line. Hyphens in parentheses, (-), suppress column headers for the COMPANY, CONTACT, and STREET fields. You do not have to suppress column headers for concatenated fields, as PRO/DATATRIEVE suppresses them automatically.

You can also use concatenated expressions to create character strings longer than 130 characters:

```
DTR> DECLARE STR PIC X(300) EDIT-STRING IS T(50).
DTR> STR = *.L1!*.L2!*.L3!*.L4!*.L5
Enter L1: This string contains the first part of a long character string.
Enter L2: This string is so long that you can't use just one character
Enter L3: string literal to assign a value to it.  You need concatenated
Enter L4: expressions to increase the length of this string beyond the
Enter L5: limit of 132 characters.
DTR> PRINT STR

                          STR

This string contains the first part of a long
character string.  This string is so long that you
can't use just one character string literal to
assign a value to it.  You need concatenated
expressions to increase the length of this string
beyond the limit of 132 characters.

DTR>
```

This example combines the T edit string (to indicate that a line of text can contain 50 characters), prompting value expressions, and character string literals to assign a value to a long variable. You can use similar assignment statements in the USING clauses of the STORE and MODIFY statements. You must type a space at the end of L1, L2, L3, L4, and L5 before pressing DO or RETURN. The T edit string prints up to 50 characters on a line, but does not break up words from one line to another unless you type a space to separate the last word on a line from the first word on the next line.

CREATING AND USING BOOLEAN EXPRESSIONS

A Boolean expression specifies a relationship between value expressions and consists of value expressions, relational operators, and Boolean operators:

- *Value expressions* specify the values you want to compare.
- *Relational operators* specify how you want to compare value expressions.

(continued on next page)

- *Boolean operators* join two or more Boolean expressions together or reverse the value of a Boolean expression.

All Boolean expressions contain value expressions and relational operators, and some contain Boolean operators. PRO/DATATRIEVE evaluates a Boolean expression as either true or false. You can use Boolean expressions in the following PRO/DATATRIEVE clauses and statements:

- The WITH clause in a record selection expression
- The IF clause of an IF-THEN-ELSE statement
- The VALID IF clause in a record definition
- The WHILE statement

See Chapter 13 for information on using IF-THEN-ELSE and WHILE statements and Chapter 8 for information on using the VALID IF clause.

Relational Operators

Relational operators allow you to compare value expressions, check whether a code string is contained in a table, or check whether a record stream is empty or not. Most Boolean expressions contain a field name, a relational operator, and a value expression.

To save typing time, you can use the following abbreviations and symbols for relational operators:

EQUAL ——→ EQ or =
NOT-EQUAL ——→ NE
GREATER-THAN ——→ GT or >
GREATER-EQUAL ——→ GE
LESS-THAN ——→ LT or <
LESS-EQUAL ——→ LE
BETWEEN ——→ BT
CONTAINING ——→ CONT

Table 2–3 shows relational operators and describes their meaning. The characters a, b, and c in the table stand for value expressions used in the comparison.

Table 2-3: Relational Operators

Operator	Meaning
a EQUAL b	True if a and b are equal
a NOT-EQUAL b	True if a and b are not equal
a GREATER-THAN b	True if a is greater than b
a GREATER-EQUAL b	True if a is greater than or equal to b
a LESS-THAN b	True if a is less than b
a LESS-EQUAL b	True if a is less than or equal to b
a BETWEEN b AND c	True if a is between b and c or if a is equal to b or c
a NOT BETWEEN b AND c	True if a is not between b and c or not equal to b or c
a CONTAINING b	True if a contains b
a NOT CONTAINING b	True if a does not contain b
a IN table-name	True if a is found in the specified table
a NOT IN table-name	True if a is not found in the specified table
ANY rse	True if the record stream formed by the RSE is not empty

You can specify more than one value expression only after the =, EQ, EQUAL, and NOT EQUAL operators. PRO/DATATRIEVE displays an error message if you specify more than one value expression after any other operator:

```
DTR> READY EMPLOYEES
DTR> FIND EMPLOYEES WITH LAST-NAME EQ "BOOLE", "EINSTEIN"
[5 records found]
DTR> FIND EMPLOYEES WITH LAST-NAME NE "BOOLE", "EINSTEIN"
Expected end of statement, encountered ","
DTR>
```

Notice that, although you can use NOT EQUAL with multiple value expressions, you cannot use NOT-EQUAL or NE in the same way.

You can use a variable name in place of a field name in a Boolean expression. You cannot, however, use any other value expressions in place of the field name. Generally, you use relational operators to compare field values to value expressions.

When you use relational operators that compare for lesser or greater values, PRO/DATATRIEVE treats lowercase letters in character string literals as "greater than" uppercase letters.

All operators except CONTAINING are case-sensitive. Thus, EQ "BOOLE" and EQ "Boole" do not perform the same comparison. The CONTAINING operator, however, is not sensitive to case. Uppercase letters and lowercase letters are treated the same, whether in quotation marks or not:

```
DTR> READY EMPLOYEES
DTR> PRINT EMPLOYEES WITH LAST-NAME EQ "EINSTEIN"

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

78375 EXPERIENCED ALBERT     EINSTEIN   MATHEMATICS      $40,095

DTR> PRINT EMPLOYEES WITH LAST-NAME EQ "Einstein"
DTR> PRINT EMPLOYEES WITH LAST-NAME CONT "Einstein"

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

78375 EXPERIENCED ALBERT     EINSTEIN   MATHEMATICS      $40,095

DTR>
```

Use the IN relational operator to compare the contents of a field with the code strings in a dictionary table. You can, for example, write a record definition that uses a table to validate the data before it is stored:

```
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>    03 NAME PIC X(20).
DFN>    03 NUMBER PIC 9(7)
DFN>       EDIT-STRING IS XXX-XXXX.
DFN>    03 AREA-CODE PIC XXX VALID IF AREA-CODE IN AREA-TABLE.
DFN> ;
DTR>
```

Use the ANY relational operator to determine whether a record stream is empty. This operator is particularly useful with lists. The RSE following the ANY operator generally specifies the name of a list or sublist:

```
DTR> SET NO PROMPT
DTR> PRINT FAMILIES WITH ANY KIDS WITH
CON>     KID-NAME CONT "RAL"
```

```
                          NUMBER     KID
  FATHER       MOTHER      KIDS      NAME      AGE

JIM            ANN           2     URSULA       7
                                   RALPH        3

DTR>
```

Boolean Operators

You use the AND and OR Boolean operators to join Boolean expressions and the NOT Boolean operator to reverse the value of a Boolean expression:

- When you join Boolean expressions with AND, the resulting expression is true only if both expressions are true:

```
DTR> PRINT EMPLOYEES WITH (LAST-NAME EQ "BOOLE") AND
[Looking for Boolean expression]
CON>        (STATUS EQ "EXPERIENCED")

                          FIRST        LAST
 ID      STATUS            NAME        NAME          DEPT         SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000
83771 EXPERIENCED GERALD      BOOLE       ASTRONOMY        $21,000
```

- When you link Boolean expressions with OR, the resulting expression is true if either one of the expressions is true:

```
DTR> PRINT EMPLOYEES WITH (LAST-NAME EQ "EINSTEIN") OR
[Looking for Boolean expression]
CON>       (SALARY LE 2000)

                          FIRST        LAST
 ID      STATUS            NAME        NAME          DEPT         SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000
78375 EXPERIENCED ALBERT      EINSTEIN    MATHEMATICS      $40,095
```

- When you precede a Boolean expression with NOT, the resulting expression is true if the expression following NOT is false:

```
DTR> PRINT EMPLOYEES WITH SALARY NOT GT 2000

                          FIRST        LAST
 ID      STATUS            NAME        NAME          DEPT         SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000
```

Use parentheses to group Boolean expressions into compound Boolean expressions and to ensure that your Boolean expressions identify the records you want. For example:

```
DTR> SET NO PROMPT
DTR> FIND EMPLOYEES WITH DEPT CONT "MATH" OR DEPT CONT "ASTR" AND
CON>     STATUS EQ "EXPERIENCED"
[11 records found]
DTR> FIND EMPLOYEES WITH (DEPT CONT "MATH" OR DEPT CONT "ASTR") AND
CON>     (STATUS EQ "EXPERIENCED")
[7 records found]
DTR>
```

The Boolean expression without parentheses first identifies all employees in the math or astronomy department (the OR expressions). Then it identifies experienced employees only for the astronomy department (the AND expression). The Boolean expression with parentheses identifies all employees in the math or astronomy department. It then identifies only experienced employees for both those departments.

PRO/DATATRIEVE evaluates Boolean expressions in parentheses before evaluating other Boolean expressions. If a Boolean expression contains Boolean operators as well as parentheses, PRO/DATATRIEVE evaluates the expression in this order:

1. Expressions enclosed in parentheses
2. Expressions preceded by NOT
3. Expressions combined with AND
4. Expressions combined with OR

CREATING AND USING RECORD SELECTION EXPRESSIONS

A record selection expression (RSE) identifies the records you want to include in a record stream and consists of the following components:

- An optional *restriction clause* (FIRST or ALL)
- An optional *name clause*
- A required *record source* (a domain, collection, or list field)
- An optional *condition clause* (WITH)
- An optional *sort clause* (SORTED BY)

A complete RSE looks like this:

```
DTR> SET NO PROMPT
DTR> PRINT ALL MATH-PROFS IN EMPLOYEES WITH
CON>      DEPT CONT "MATH" SORTED BY LAST-NAME
```

This RSE tells PRO/DATATRIEVE to include all records with the DEPT field containing "MATH" in a record stream named MATH-PROFS, then to sort records in the record stream by the LAST-NAME field. The EMPLOYEES domain is the source for the record stream.

To use a collection as the record source, form the collection and specify the collection name in the RSE. To use a list field as the record source, use the name of the list field as the record source in the RSE.

By using a list name to specify the source of records for an RSE, you can form a record stream from the list items in a single record or in multiple records. In the following example, the SELECT statement picks out the first record in the current collection. The RSE in the PRINT statement identifies the first item in the KIDS list in the selected record as the record stream:

```
DTR> READY FAMILIES
DTR> FIND FAMILIES
[14 records found]
DTR> SELECT
DTR> PRINT AGE OF FIRST 1 KIDS

AGE

 7

DTR>
```

See Chapter 4 for more information on working with list fields. For more information about specifying the record source and using the restriction, condition, and sort clauses, see Chapter 4 in *PRO/DATATRIEVE for Beginners*.

You can give a record stream a name by including a name clause in the RSE. When you use a name clause in an RSE in a FIND statement, the name becomes the name of the collection. Until you form another collection or release the one just formed, the collection has both the name you gave it and the name CURRENT:

```
DTR> READY EMPLOYEES
DTR> FIND MATH-PROFS WITH DEPT CONT "MATH"
[8 records found]
DTR> SHOW COLLECTIONS
Collections:
        MATH_PROFS (also CURRENT)
DTR>
```

When you use a name clause in an RSE in a statement other than FIND, the name exists only as long as the record stream it names exists. When the statement containing the RSE finishes executing, the name disppears and you can no longer use it to refer to the record stream. When used to name a record stream, the name is called a *context variable* because it exists only within the context of a statement.

By using context variables to name record streams, you can refer to fields from different record streams in the same statement. The following procedure, for example, uses two FOR statements to form and name record streams from two domains. The procedure then finds prices in the INVENTORY domain and writes those prices to records with matching item numbers and no prices in the TRANS domain:

```
DTR> SHOW WRITE-PRICE
PROCEDURE WRITE_PRICE
    FINISH
    READY TRANS WRITE
    READY INVENTORY
    FOR A IN INVENTORY
    FOR B IN TRANS WITH (B.NUM = A.NUM) AND
            (TOTAL_TRANS = 0)
    MODIFY USING B.UNIT_PRICE = A.UNIT_PRICE
END_PROCEDURE
DTR>
```

The first FOR statement creates a record stream named A from the TRANS domain and the second FOR statement creates a record stream named B from the INVENTORY domain. Both domains have fields named NUM and UNIT-PRICE, so the context variables are necessary to distinguish between the fields. B.NUM, A.NUM, B.UNIT-PRICE, and A.UNIT-PRICE are *qualified field names* that tell PRO/DATATRIEVE which field to find values in and which fields to copy values to.

See Chapter 3 for more information on using context variables and qualified field names and Chapter 13 for information on using the FOR statement.

SUMMARY

This chapter described how to create and use PRO/DATATRIEVE expressions:

- A value expression specifies a value for PRO/DATATRIEVE to use when executing statements.
- A Boolean expression specifies a relationship between value expressions for PRO/DATATRIEVE to evaluate when executing statements.
- A record selection expression (RSE) identifies the records you want to include in a record stream or a collection.

For more information on statements, commands, and clauses used in this chapter, refer to the following chapters:

- See Chapter 3 for information on using context variables and qualified field names.
- See Chapter 4 for information on list fields.
- See Chapter 7 for information on creating and using tables.
- See Chapter 8 for information on field definition clauses.
- See Chapter 13 for information on BEGIN-END, THEN, REPEAT, FOR, IF-THEN-ELSE, and WHILE statements.
- See Chapter 15 for information on creating and using variables.

3

Establishing Context for PRO/DATATRIEVE Statements

Chapter 3

Establishing Context for PRO/DATATRIEVE Statements

When you use a field name as a value expression or when you display, modify, or erase one or more records, PRO/DATATRIEVE determines which field, record, or records to use based on the current *context*. Context is the set of conditions that determines how PRO/DATATRIEVE recognizes field names and identifies target records for statements.

PRO/DATATRIEVE always has to have a context within which to operate. If, for example, you try to display records from a domain you have not readied, PRO/DATATRIEVE displays a message telling you that the domain name is "undefined or used out of context." If you try to display a field that does not exist for the domain you specify in the RSE, PRO/DATATRIEVE displays the same error message. PRO/DATATRIEVE does not know what records or field to display because you have not established a context for the field.

You can help PRO/DATATRIEVE identify the context for field names and records by using context variables to form qualified field names. You can also establish context by using the FIND, SELECT, and FOR statements, or by including an OF rse clause in PRINT, STORE, MODIFY, and ERASE statements.

The PRINT, MODIFY, and ERASE statements can act on one record at a time (single-record context) or on an entire record stream or collection (multiple-record context). In a single-record context, PRINT, MODIFY, and ERASE statements operate on a selected record or on each record in a record stream. In a multiple-record context, these statements operate on all records in a collection or in a record stream.

This chapter explains how to use context variables and qualified field names to help PRO/DATATRIEVE correctly identify fields in records. It also explains how to establish single-record and multiple-record context. Chapter 4 explains how to establish context for records with list fields.

USING CONTEXT VARIABLES AND QUALIFIED FIELD NAMES

A *context variable* is a dummy variable specified in an RSE. Context variables help PRO/DATATRIEVE recognize field names. You can then use the context variable as a prefix to a field name to make the field name unique.

Copying Records from One Domain to Another Domain

The following procedure defines a new file for the WORK-EMP domain and then copies records from the EMPLOYEES domain to the WORK-EMP domain. Because the WORK-EMP domain uses the same record definition as the EMPLOYEES domain, you have to use context variables to differentiate between the the records that contain EMPLOYEES data and the WORK-EMP records the procedure creates:

```
DTR> SHOW WORK-EMP
DOMAIN WORK_EMP
 USING EMPLOYEES_REC ON WORK;
DTR> SHOW CLEAN-DATA
PROCEDURE CLEAN_DATA
   DEFINE FILE FOR WORK_EMP KEY = ID
   READY WORK_EMP WRITE
   READY EMPLOYEES
   FOR A IN EMPLOYEES
      STORE WORK_EMP USING EMPLOYEES_REC = A.EMPLOYEES_REC
END_PROCEDURE
DTR>
```

The FOR statement creates a record stream from the EMPLOYEES domain and uses a context variable, A, to name that record stream. The STORE statement then uses that context variable to create a *qualified field name*, A.EMPLOYEES-REC. This field name distinguishes the records that contain EMPLOYEES data from the records that contain WORK-EMP data.

PRO/DATATRIEVE assigns values from records in the EMPLOYEES domain, identified as A.EMPLOYEES-REC, to records in the WORK-EMP domain, identified as EMPLOYEES-REC. You do not need to specify a context variable for the WORK-EMP domain because the STORE statement establishes context for the STORE operation.

If you plan on doing the examples in this chapter, you may want to define the CLEAN-DATA procedure and execute it periodically to write fresh data from the EMPLOYEES domain to the WORK-EMP domain.

Modifying and Updating Values with Context Variables

By using context variables and nested FOR statements, you can create and work with multiple record streams. This allows you to use information in one record stream to modify or update fields in another record stream.

You can, for example, put a MODIFY statement inside two FOR statements to copy prices from the INVENTORY domain to transaction records in the TRANS domain:

```
DTR> SHOW WRITE-PRICE
PROCEDURE WRITE_PRICE
FINISH
READY TRANS WRITE
READY INVENTORY
FOR A IN INVENTORY
FOR B IN TRANS WITH (B.NUM EQ A.NUM) AND
        (TOTAL_TRANS EQ 0)
MODIFY USING B.UNIT_PRICE = A.UNIT_PRICE
END_PROCEDURE
DTR>
```

The first FOR statement creates a record stream named A that contains all records in the INVENTORY domain. The second FOR statement creates a record stream named B from the TRANS domain. By using the context variables to distinguish between fields with the same name, the record stream formed from the TRANS domain contains only those records with item numbers (B.NUM) that match item numbers in the INVENTORY domain (A.NUM). The MODIFY statement then changes the UNIT-PRICE field in the TRANS domain (B.UNIT-PRICE) to the value stored in the UNIT-PRICE field in the INVENTORY domain (A.UNIT-PRICE).

See Appendix A for a listing of the INVENTORY and TRANS record definitions.

You can also create nested FOR statements in which PRO/DATATRIEVE executes a statement or series of statements at each level of nesting. The following procedure updates the INVENTORY domain based on records in the TRANS domain that have not been posted (FLAG = 0) in the inner FOR statement. Then

it changes the FLAG in the TRANS record used for the update to 1 so that it will not be used again before ending the outer FOR statement:

```
DTR> SHOW UPDATE-MASTER
PROCEDURE UPDATE_MASTER
FINISH
READY TRANS WRITE
READY INVENTORY WRITE
FOR A IN TRANS WITH FLAG EQ 0
    BEGIN
    FOR B IN INVENTORY WITH B.NUM EQ A.NUM
        BEGIN
        IF TYPE EQ "S" THEN
            MODIFY USING B.ON_HAND = B.ON_HAND - A.QUANTITY ELSE
            MODIFY USING B.ON_HAND = B.ON_HAND + A.QUANTITY
        END
    MODIFY USING A.FLAG = 1
    END
END_PROCEDURE
DTR>
```

The inner FOR statement uses an IF statement to determine when to subtract and when to add items to the ON-HAND field in the INVENTORY domain ("S" indicates a sale). Then the MODIFY statement changes the ON-HAND field in the INVENTORY domain (B.ON-HAND) accordingly. A.QUANTITY establishes context for the QUANTITY field so that PRO/DATATRIEVE knows it comes from the record stream named A. If you do not specify A.QUANTITY, PRO/DATATRIEVE does not have a context for the field and displays an error message.

The UPDATE-MASTER procedure works as follows:

```
DTR> PRINT TRANS WITH FLAG EQ 0

   ITEM     TRANS   TRANS    COMPANY                    UNIT       TOTAL
  NUMBER    TYPE     DATE     NAME    FLAG QUANTITY    PRICE      TRANS

0879432876    S     7/18/83    PP                34       $2.37      $80.58
0981414899    S     7/07/83    SA               121       $2.19     $264.99
1182413601    S     7/01/83    BS                50       $1.11      $55.50
```

```
DTR> PRINT INVENTORY WITH ITEM-NUMBER EQ 0879432876,
[Looking for next element in list]
CON>     0981414899, 1182413601

   ITEM          ITEM        ON      UNIT       TOTAL
   NAME        NUMBER      HAND     PRICE      VALUE

BAFFLES     0981414899     2353     $2.19   $5,153.07
FIXITS      0879432876     3912     $2.37   $9,271.44
WIDGETS     1182413601     2688     $1.11   $2,983.68

DTR> :UPDATE-MASTER
DTR> PRINT TRANS WITH ITEM-NUMBER EQ 0879432876,
[Looking for next element in list]
CON>     0981414899, 1182413601

   ITEM     TRANS   TRANS    COMPANY                      UNIT        TOTAL
  NUMBER    TYPE     DATE     NAME    FLAG QUANTITY     PRICE       TRANS

0879432876    S     7/18/83    PP       1        34        $2.37      $80.58
0981414899    S     7/07/83    SA       1       121        $2.19     $264.99
1182413601    S     7/01/83    BS       1        50        $1.11      $55.50

DTR> PRINT INVENTORY WITH ITEM-NUMBER EQ 0879432876,
[Looking for next element in list]
CON>     0981414899, 1182413601

   ITEM          ITEM        ON      UNIT       TOTAL
   NAME        NUMBER      HAND     PRICE      VALUE

BAFFLES     0981414899     2232     $2.19   $4,888.08
FIXITS      0879432876     3878     $2.37   $9,190.86
WIDGETS     1182413601     2638     $1.11   $2,928.18

DTR>
```

See Chapter 13 for more information on using FOR, BEGIN-END, and IF statements. See Chapter 12 for more information on storing, modifying, and deleting information.

ESTABLISHING SINGLE-RECORD CONTEXT

You can establish single-record context in two ways:

- By selecting a record from a collection with the SELECT statement
- By putting the PRINT, MODIFY, or ERASE statement in a FOR statement

Operating on a Selected Record

Because you can select records only from collections, a short review of facts about collections is in order.

You use the FIND statement to form a collection. PRO/DATATRIEVE names that collection CURRENT, in addition to any other name you give it, and puts the collection at the top of a collection list. The most recently formed collection, the "youngest," is always at the top of the list, while the "oldest" collection is at the bottom of the list.

To see the collection list and the order in which you formed collections, use the SHOW COLLECTIONS command:

```
DTR> READY EMPLOYEES
DTR> FIND MATH IN EMPLOYEES WITH DEPT CONT "MATH"
[8 records found]
DTR> SHOW COLLECTIONS
Collections:
    MATH (also CURRENT)
DTR> FIND BIOL IN EMPLOYEES WITH DEPT CONT "BIOL"
[4 records found]
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    MATH
DTR> FIND EMPLOYEES WITH DEPT CONT "ASTRON"
[5 records found]
DTR> SHOW COLLECTIONS
Collections:
    CURRENT
    BIOL
    MATH
DTR>
```

Because this example did not name the last collection, the collection of astronomists has only one name, CURRENT. This collection disappears when you form a new collection because it has no other name. When you name a collection, it does not disappear when you form a new collection.

You remove a collection from the collection list and from memory with the RELEASE command:

```
DTR> RELEASE BIOL
DTR> SHOW COLLECTIONS
Collections:
    CURRENT
    MATH
DTR>
```

If you release the CURRENT collection, the next collection becomes the CURRENT collection:

```
DTR> RELEASE CURRENT
DTR> SHOW COLLECTIONS
Collections:
    MATH (also CURRENT)
DTR>
```

The relative ages of collections or their positions in the collection list determine a collection context. Collection context then determines the selected record context. If you have a selected record in the CURRENT collection and type just PRINT, PRO/DATATRIEVE identifies that record as the target record:

```
DTR> READY WORK-EMP WRITE
DTR> FIND MATH IN WORK-EMP WITH DEPT CONT "MATH"
[8 records found]
DTR> SELECT 3
DTR> FIND ASTRON IN WORK-EMP WITH DEPT CONT "AST"
[5 records found]
DTR> SELECT 2
DTR> FIND BIOL IN WORK-EMP WITH DEPT CONT "BIOL"
[4 records found]
DTR> SELECT 1
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR> PRINT
                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT        SALARY

49001 EXPERIENCED DAN          ROBERTS    BIOLOGY          $41,395

DTR>
```

You can also display specific fields in the selected record by simply typing PRINT and the names of the fields:

```
DTR> PRINT LAST-NAME, SALARY

   LAST
   NAME     SALARY

ROBERTS     $41,395

DTR>
```

Because you did not provide a context for the LAST-NAME and SALARY fields, PRO/DATATRIEVE looks for the fields in the selected record.

When the CURRENT collection does not have any record selected, PRO/DATATRIEVE identifies as the target record the selected record in the next "youngest" collection that has a selected record. The next example uses the DROP statement to drop the selected record from the CURRENT collection and then prints the selected record from the ASTRON collection:

```
DTR> PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

49001 EXPERIENCED DAN         ROBERTS    BIOLOGY         $41,395

DTR> DROP
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR> PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

78923 EXPERIENCED LYDIA       HARRISON   ASTRONOMY       $40,747

DTR>
```

Because you have dropped the selected record in the CURRENT collection, the second PRINT statement displays the selected record in the ASTRON collection, the next collection with a selected record.

The DROP statement deletes a selected record from a collection, but not from a data file. If you type DROP and press DO, PRO/DATATRIEVE drops the selected record in the next collection with a selected record. If you want to drop a selected record in a particular collection, you can specify the collection name in the DROP statement. See Chapter 14 for information on using the DROP statement.

If you drop the selected record in the ASTRON collection, PRO/DATATRIEVE goes to the next collection in the list to find the selected record:

```
DTR> PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

78923 EXPERIENCED LYDIA       HARRISON   ASTRONOMY       $40,747
```

```
DTR> DROP
DTR> PRINT

                        FIRST        LAST
 ID      STATUS          NAME        NAME          DEPT         SALARY

03991 TRAINEE       JAMES       BOOLE       MATHEMATICS      $14,000

DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR>
```

If you type ERASE or MODIFY and press DO, PRO/DATATRIEVE deletes or changes the "youngest" selected record, just as it displays the "youngest" selected record when you type PRINT and press DO:

```
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR> PRINT

                        FIRST        LAST
 ID      STATUS          NAME        NAME          DEPT         SALARY

03991 TRAINEE       JAMES       BOOLE       MATHEMATICS      $14,000

DTR> ERASE
DTR> PRINT
No record selected, printing whole collection

                        FIRST        LAST
 ID      STATUS          NAME        NAME          DEPT         SALARY

87465 EXPERIENCED ANTHONY     IACOBONE    BIOLOGY          $58,462
90342 EXPERIENCED BRUNO       DONCHIKOV   BIOLOGY          $35,952
99029 EXPERIENCED RANDY       PODERESIAN BIOLOGY           $33,738

DTR>
```

Because you have dropped the selected records in the BIOL and ASTRON collections, the first PRINT statement in the example displays the selected record from the MATH collection. The ERASE statement then deletes that record from the data file, so the next PRINT statement has no selected record in any collection to display. Thus, PRO/DATATRIEVE displays a message and prints the whole CURRENT collection.

The MODIFY statement operates on selected records, just as the PRINT and ERASE statements do. The following example selects the first record in the ASTRON collection, displays it with PRINT to make sure it is the "nearest" selected record, and modifies the SALARY field:

```
DTR> SELECT 1 ASTRON
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR> PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY       $75,892

DTR> MODIFY SALARY
Enter SALARY: 50000
DTR> PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY       $50,000

DTR>
```

Because the context for the selected record you want to change, erase, or drop may not be what you think it is, use the PRINT statement before using a MODIFY, ERASE, or DROP statement.

Operating on Each Record in a Record Stream

By putting a PRINT, MODIFY, or ERASE statement in a FOR statement, you create a single-record context for individual records in a record stream. This technique has the advantage of letting you work with many records at one time, while letting you deal with each record individually.

The RSE in the FOR statement creates a record stream for the following PRINT, MODIFY, or ERASE statement:

```
DTR> READY WORK-EMP WRITE
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "BIOL"
```

```
                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

49001 EXPERIENCED DAN         ROBERTS     BIOLOGY         $41,395
87465 EXPERIENCED ANTHONY     IACOBONE    BIOLOGY         $58,462

DTR> FOR FIRST 2 WORK-EMP WITH DEPT CONT "BIOL"
[Looking for statement]
CON>     MODIFY SALARY
Enter SALARY: 42000
Enter SALARY: 60000
DTR> FOR FIRST 2 WORK-EMP WITH DEPT CONT "BIOL"
[Looking for statement]
CON>     PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

49001 EXPERIENCED DAN         ROBERTS     BIOLOGY         $42,000
87465 EXPERIENCED ANTHONY     IACOBONE    BIOLOGY         $60,000

DTR>
```

In this example, the FOR statements create record streams for the MODIFY and PRINT statements. The MODIFY statement then prompts for a unique value for the SALARY field for each record in the record stream and changes each record accordingly. As this example shows, the PRINT statement in the FOR statement produces the same display as the PRINT statement with the RSE produces. However, as the next section shows, a MODIFY statement in a FOR statement produces different results from a MODIFY statement with an RSE.

If you want to see a record before modifying a field in it, you can use the THEN statement to join the PRINT and MODIFY statement. If you do not want to modify a value, press the **TAB** key, as in this example, to keep the old value:

```
DTR> FOR FIRST 2 WORK-EMP WITH DEPT CONT "PHILOS"
[Looking for statement]
CON>     PRINT THEN MODIFY STATUS, SALARY

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

34456 TRAINEE     HANK        MORRISON    PHILOSOPHY      $30,000
Enter EMPLOYEE_STATUS: EXPERIENCED
Enter SALARY: 35000
39485 EXPERIENCED DEE         TERRICK     PHILOSOPHY      $55,829
Enter EMPLOYEE_STATUS: (TAB)
Enter SALARY: 56000
```

(continued on next page)

```
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "PHILOS"

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

34456 EXPERIENCED HANK        MORRISON   PHILOSOPHY      $35,000
39485 EXPERIENCED DEE         TERRICK    PHILOSOPHY      $56,000

DTR>
```

See Chapter 13 for more information on using the THEN statement.

If you include an ERASE statement in a FOR statement, PRO/DATATRIEVE erases each record in the record stream:

```
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "PHILO"

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

34456 EXPERIENCED HANK        MORRISON   PHILOSOPHY      $35,000
39485 EXPERIENCED DEE         TERRICK    PHILOSOPHY      $56,000

DTR> FOR FIRST 2 WORK-EMP WITH DEPT CONT "PHILO" ERASE
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "PHILO"

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

48573 TRAINEE      SY          KELLER     PHILOSOPHY      $31,546
49843 TRAINEE      BART        HAMMER     PHILOSOPHY      $26,392

DTR>
```

Note that the ERASE statement in this example is on the same line as the FOR statement. This is perfectly valid and stops PRO/DATATRIEVE from prompting you for a statement with the CON> prompt.

ESTABLISHING MULTIPLE-RECORD CONTEXT

You establish a multiple-record context for a collection by including the keyword ALL in PRINT, MODIFY, or ERASE statements. You establish a multiple-record context for a record stream by putting an OF rse clause in a PRINT, MODIFY, or ERASE statement.

Using the MODIFY statement in a multiple-record context is quite different from using it in a single-record context. In a single-record context, you can change fields in a record to unique values. In a multiple-record context, you change fields in all records to the same value, unless the new value derives from the old value, as explained in the following sections. You should use the MODIFY statement in a multiple-record context with care.

Operating on All Records in a Collection

The order of collections in the collection list, as described earlier in this chapter, determines the collection context. If you type PRINT and press DO with no selected record in any collection, PRO/DATATRIEVE displays a message and prints the CURRENT collection. If any collection has a selected record, PRINT displays a selected record.

By including the keyword ALL in a PRINT, MODIFY, or ERASE statement, you can make these statements operate on all records in a collection. If you do not include ALL, PRO/DATATRIEVE acts only on a selected record.

If you have a CURRENT collection and type PRINT ALL, PRO/DATATRIEVE displays the whole CURRENT collection. If you have no CURRENT collection, PRO/DATATRIEVE displays a message:

```
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR> RELEASE BIOL, ASTRON, MATH
DTR> PRINT ALL
A current collection has not been established
DTR>
```

PRO/DATATRIEVE displays the same message if you type ERASE ALL or MODIFY ALL when you you have no CURRENT collection.

When you specify ERASE ALL, PRO/DATATRIEVE deletes from the data file every record in the CURRENT collection. Although frequently useful, erasing all records in a collection can jeopardize valuable data if done carelessly. It is a good

idea to use the PRINT ALL statement to see which records the ERASE ALL statement will erase:

```
DTR> SHOW COLLECTIONS
Collections:
    BIOL (also CURRENT)
    ASTRON
    MATH
DTR> PRINT ALL

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

49001 EXPERIENCED DAN         ROBERTS     BIOLOGY         $41,395
87465 EXPERIENCED ANTHONY     IACOBONE    BIOLOGY         $58,462
90342 EXPERIENCED BRUNO       DONCHIKOV   BIOLOGY         $35,952
99029 EXPERIENCED RANDY       PODERESIAN  BIOLOGY         $33,738

DTR> ERASE ALL
DTR> PRINT ALL
DTR>
```

When you use a MODIFY ALL statement, PRO/DATATRIEVE changes field values in every record in the CURRENT collection to the same value, unless the new field value is calculated from the old field value. For example:

```
DTR> FIND FIRST 2 PHILOS IN WORK-EMP WITH DEPT CONT "PHILOS"
[2 records found]
DTR> PRINT ALL

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

34456 TRAINEE     HANK        MORRISON    PHILOSOPHY      $30,000
39485 EXPERIENCED DEE         TERRICK     PHILOSOPHY      $55,829

DTR> MODIFY ALL SALARY
Enter SALARY: 25000
DTR> PRINT ALL

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

34456 TRAINEE     HANK        MORRISON    PHILOSOPHY      $25,000
39485 EXPERIENCED DEE         TERRICK     PHILOSOPHY      $25,000
```

```
DTR> FIND FIRST 2 LIT IN EMPLOYEES WITH DEPT CONT "LIT"
[2 records found]
DTR> PRINT ALL

                        FIRST        LAST
 ID      STATUS          NAME        NAME          DEPT          SALARY

02943 EXPERIENCED CASS          TERRY       LITERATURE       $29,908
12643 TRAINEE     JEFF          TASHKENT    LITERATURE       $32,918

DTR> MODIFY ALL USING SALARY = SALARY * 1.13
DTR> PRINT ALL

                        FIRST        LAST
 ID      STATUS          NAME        NAME          DEPT          SALARY

02943 EXPERIENCED CASS          TERRY       LITERATURE       $33,796
12643 TRAINEE     JEFF          TASHKENT    LITERATURE       $37,197

DTR>
```

The first MODIFY ALL statement assigns the value 25000 to the SALARY field for all records in the CURRENT collection. The second MODIFY ALL statement uses the old value in the SALARY field of each record to calculate a 13% raise for each record in the CURRENT collection, so the new SALARY values are not all the same.

As with ERASE ALL, use MODIFY ALL with care. To make sure that MODIFY ALL will change only the records you want to change, use a PRINT ALL statement first.

Operating on All Records in a Record Stream

You can establish context for an entire record stream by including an OF rse clause in a PRINT, MODIFY, or ERASE statement. This creates a record stream context for the statement that overrides any collection and selected record context previously established. When the statement finishes executing, the record stream context disappears.

This type of record stream context is similar to the collection context created with the PRINT ALL, MODIFY ALL, and ERASE ALL statements that operate on collections, so you must include the keyword ALL in MODIFY and ERASE statements that operate on a record stream. This forces you to think carefully about the effect of modifying or erasing all records in a record stream.

The ERASE ALL OF rse statement deletes *all* records in the record stream. The MODIFY ALL OF rse statement changes the specified field or fields in all records in the record stream to the same value, unless the new field values are calculated from differing old field values. For example:

```
DTR> SHOW COLLECTIONS
Collections:
    LIT (also CURRENT)
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "MATH"

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS      $2,000
00891 EXPERIENCED FRED        HOWL        MATHEMATICS     $59,594

DTR> ERASE ALL OF FIRST 2 WORK-EMP WITH DEPT CONT "MATH"
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "MATH"

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

03991 TRAINEE      JAMES       BOOLE       MATHEMATICS     $14,000
11111 TRAINEE      ALLEN       SMITH       MATHEMATICS     $20,000

DTR> MODIFY ALL SALARY OF FIRST 2 WORK-EMP WITH DEPT CONT "MATH"
Enter SALARY: 25000
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "MATH"

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

03991 TRAINEE      JAMES       BOOLE       MATHEMATICS     $25,000
11111 TRAINEE      ALLEN       SMITH       MATHEMATICS     $25,000

DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "LIT"

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

02943 EXPERIENCED CASS        TERRY       LITERATURE      $33,796
12643 TRAINEE     JEFF        TASHKENT    LITERATURE      $37,197
```

```
DTR> MODIFY ALL USING SALARY = SALARY * 1.13 OF
[Looking for "FIRST", domain name, or collection name]
CON>      FIRST 2 WORK-EMP WITH DEPT CONT "LIT"
DTR> PRINT FIRST 2 WORK-EMP WITH DEPT CONT "LIT"

                      FIRST        LAST
 ID      STATUS        NAME        NAME          DEPT         SALARY

02943 EXPERIENCED CASS         TERRY       LITERATURE       $38,189
12643 TRAINEE     JEFF         TASHKENT    LITERATURE       $42,032

DTR>
```

Because a PRINT statement does not change or delete data, you do not have to specify ALL or OF when you include an RSE.

SUMMARY

This chapter showed how to establish context for PRINT, MODIFY, and ERASE statements and how to use context variables:

- You establish single-record context for a selected record with the SELECT statement and for individual records in a record stream by nesting a statement in a FOR statement.
- You establish multiple-record context for collections by including ALL in the PRINT, MODIFY, or ERASE statement and for record streams by including ALL and an OF rse clause.

You can also use context variables in FOR statements to establish context for record fields when using records with the same field names or when using field names in FOR loops. For more information on commands and statements used in this chapter, refer to the following chapters:

- See Chapter 4 for information on establishing context for list fields.
- See Chapter 12 for information on storing, modifying, and deleting information.
- See Chapter 13 for information on FOR, BEGIN-END, IF-THEN-ELSE, and THEN statements.
- See Chapter 14 for information on forming and using collections.

4

Working with Lists

Chapter 4

Working with Lists

List fields, sometimes called repeating fields, contain multiple values. You define fixed-length list fields with the OCCURS field definition clause and variable-length lists with the OCCURS DEPENDING clause. Chapter 8 explains how to use these clauses.

List fields are somewhat more difficult to access than fields in flat records (records without list fields), so this chapter provides many examples of displaying and modifying list values, using the SUPPLIES domain and the SUPPPLIES-REC record:

```
01 WHAT.
   05 ITEM PIC X(15).
   05 ORDER-UNIT PIC X(10).
   05 HOW-MANY PIC 9.
   05 VENDORS OCCURS 0 TO 10 TIMES DEPENDING ON HOW-MANY.
      07 COMPANY PIC X(10).
      07 PRICE PIC 99V99
         EDIT-STRING IS $$$.$$.
      07 DISCOUNT OCCURS 3 TIMES.
         09 MIN-ORDER PIC 9(5)
            EDIT-STRING IS Z(5).
         09 PERCENT PIC 99
            EDIT-STRING IS Z9%.
         09 WE-PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
            EDIT-STRING IS $$$.$$.
;
```

The VENDORS list field, a variable-length list, contains two elementary fields, COMPANY and PRICE, and a fixed-length list, DISCOUNT. The DISCOUNT list contains three elementary fields, MIN-ORDER, PERCENT, and WE-PAY. One SUPPLIES-REC record looks like this:

```
DTR> READY SUPPLIES
DTR> PRINT FIRST 1 SUPPLIES

                ORDER     HOW                              MIN              WE
     ITEM         UNIT     MANY   COMPANY     PRICE     ORDER  PERCENT     PAY

PAPER           REAM        3    ABC SUPPLY  $13.45      150      3%    $13.04
                                                          500      5%    $12.77
                                                         1000      7%    $12.50
                                  MORRIS      $12.98       10      2%    $12.72
                                                           50      4%    $12.46
                                                          100      6%    $12.20
                                  XYZ DATA    $13.00      100      2%    $12.74
                                                          500      4%    $12.48
                                                         1000      7%    $12.09

DTR>
```

The following sections in this chapter describe how to establish context for list fields and how to display and modify list field values.

ESTABLISHING CONTEXT FOR LIST FIELDS

Before you can work with a list field, you must establish context for the list. You cannot, for example, simply specify the list name in a PRINT statement as you do to display a field that is not a list:

```
DTR> READY SUPPLIES
DTR> PRINT ITEM OF SUPPLIES

     ITEM

PAPER
PENS

DTR> PRINT VENDORS OF SUPPLIES
Expected end of statement, encountered "OF"
DTR> MODIFY FIRST 1 VENDORS OF SUPPLIES
Expected end of statement, encountered "OF"
DTR>
```

If you think of a list as a domain within a domain, you can then think of list items as records within a list and can establish context in one of the following ways:

- You can use successive FIND and SELECT statements until the selected "record" contains the list field or fields you want to print or modify.
- You can use nested FOR statements to identify the domain that contains the list and the list that contains the fields you want to print or modify.
- You can use a *list specification* in a PRINT statement to identify the domain that contains the list and the list field or fields you want to print.

The following sections describe how to establish these three types of list context.

Using FIND and SELECT Statements

By using successive FIND and SELECT statements, you can establish a list field or a list item as the selected "record." Then you can display or modify the list item.

First, use a FIND statement to form a collection. Then, use the SELECT statement to select the record that contains the list item or items you want to print or modify:

```
DTR> READY SUPPLIES MODIFY
DTR> FIND FIRST 1 SUPPLIES
[1 Record found]
DTR> PRINT CURRENT

                    ORDER     HOW                              MIN                    WE
     ITEM            UNIT     MANY   COMPANY     PRICE       ORDER      PERCENT     PAY

PAPER              REAM          3   ABC SUPPLY  $13.45        150           3%    $13.04
                                                                500           5%    $12.77
                                                               1000           7%    $12.50
                                      MORRIS      $12.98         10           2%    $12.72
                                                                 50           4%    $12.46
                                                                100           6%    $12.20
                                      XYZ DATA    $13.00         100          2%    $12.74
                                                                500           4%    $12.48
                                                               1000           7%    $12.09

DTR> SELECT
DTR>
```

The FIND statement establishes a collection that contains the first SUPPLIES record. The SELECT statement establishes that record as the selected record.

You use another FIND statement to establish the VENDORS list as a collection and another SELECT statement to establish an item in the list as the selected record:

```
DTR> FIND FIRST 1 VENDORS
[1 Record found]
DTR> PRINT CURRENT

                        MIN                  WE
 COMPANY    PRICE     ORDER     PERCENT    PAY

ABC SUPPLY $13.45      150         3%     $13.04
                       500         5%     $12.77
                      1000         7%     $12.50

DTR> SELECT
DTR>
```

The FIND statement establishes the first item in the VENDORS field in the selected record as the new CURRENT collection and the SELECT statement establishes the ABC SUPPLY item as the selected record. You can use a PRINT statement to display the COMPANY, PRICE, or DISCOUNT field in the list, or you can use a MODIFY statement to change the COMPANY or PRICE field:

```
DTR> PRINT COMPANY, PRICE

 COMPANY    PRICE

ABC SUPPLY $13.45

DTR> PRINT DISCOUNT

 MIN              WE
ORDER PERCENT   PAY

  150     3%    $13.04
  500     5%    $12.77
 1000     7%    $12.50

DTR> PRINT PRICE THEN MODIFY PRICE

PRICE

$13.45
Enter PRICE: 13.89
```

```
DTR> PRINT

                    MIN                 WE
 COMPANY    PRICE  ORDER  PERCENT      PAY

ABC SUPPLY $13.89    150      3%     $13.47
                     500      5%     $13.19
                    1000      7%     $12.91

DTR>
```

If you want to modify a field in the DISCOUNT list, you can include an RSE in the MODIFY statement to identify the item you want to modify. You can also use another sequence of FIND and SELECT statements to select the item. The following example modifies the PERCENT field in the second list item by including an RSE in the MODIFY statement. It then selects and modifies the third PERCENT field:

```
DTR> MODIFY PERCENT OF DISCOUNT WITH PERCENT EQ 5
Enter PERCENT: 6
DTR> PRINT

                     MIN                   WE
 COMPANY    PRICE    ORDER     PERCENT     PAY

ABC SUPPLY $13.89      150         3%     $13.47
                       500         6%     $13.05
                      1000         7%     $12.91

DTR> FIND DISCOUNT
[3 records found]
DTR> SELECT 3
DTR> PRINT
Field "PRICE" is undefined or used out of context
DTR> PRINT PERCENT

PERCENT

   7%

DTR> MODIFY PERCENT
Enter PERCENT: 8
DTR>
```

Note that PRO/DATATRIEVE cannot display the whole DISCOUNT list because the WE-PAY COMPUTED BY field depends on the PRICE field in the VENDORS list. The VENDORS list is no longer part of the selected record, so the PRICE field is out of context. To see the modified ABC SUPPLY list, you need to also print the PRICE field from the VENDORS list. You can form collections, select records, and print fields as you did before, or you can put the PRINT statement in nested FOR loops as explained in the next section.

Using Nested FOR Statements

An easier way to establish context for lists is to use nested FOR statements. The first FOR statement establishes a record stream. Successive FOR statements establish context for list fields. The statement or statements in the inner FOR loop then act on all list items in one target record before acting on any in the next record in the record stream:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> FOR SUPPLIES WITH ITEM EQ "PAPER"
CON>     FOR VENDORS WITH COMPANY CONT "ABC"
CON>         PRINT DISCOUNT

   MIN                 WE
 ORDER     PERCENT   PAY
   150        3%    $13.47
   500        6%    $13.05
  1000        8%    $12.77

DTR>
```

The first FOR statement establishes the record for PAPER as the record stream. The second FOR statement establishes context for the ABC SUPPLY list item in the VENDORS list. Within this context, the PRINT statement displays all items in the DISCOUNT list for ABC SUPPLY.

The following example shows how one FOR statement identifies the first record in the domain as the target record, then how multiple FOR statements identify the VENDORS list and then the DISCOUNT list as the target records:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> FOR FIRST 1 SUPPLIES
CON>     PRINT

                ORDER    HOW                          MIN              WE
     ITEM        UNIT    MANY  COMPANY    PRICE    ORDER  PERCENT    PAY

PAPER          REAM       3   ABC SUPPLY $13.89     150     3%    $13.47
                                                     500     6%    $13.05
                                                    1000     8%    $12.77
                                          MORRIS    $12.98      10     2%    $12.72
                                                       50     4%    $12.46
                                                      100     6%    $12.20
                                          XYZ DATA  $13.00      100     2%    $12.74
                                                      500     4%    $12.48
                                                     1000     7%    $12.09
```

```
DTR> FOR FIRST 1 SUPPLIES
CON>     FOR FIRST 1 VENDORS
CON>         PRINT

                      MIN              WE
 COMPANY    PRICE   ORDER PERCENT    PAY

ABC SUPPLY $13.89    150     3%    $13.47
                     500     6%    $13.05
                    1000     8%    $12.77

DTR> FOR FIRST 1 SUPPLIES
CON>     FOR FIRST 1 VENDORS
CON>         FOR DISCOUNT
CON>             PRINT

 MIN              WE
ORDER PERCENT    PAY

  150     3%    $13.47
  500     6%    $13.05
 1000     8%    $12.77

DTR>
```

Note that the three nested FOR statements that display the DISCOUNT field maintain context for the PRICE field in the VENDORS list, so you can print the WE-PAY field from the DISCOUNT list. You could not print this field when you used FIND and SELECT statements to establish context. Each time you formed a new collection, the context established for the old collection disappeared, so PRO/DATATRIEVE did not know which PRICE value to use to compute WE-PAY values. In contrast, FOR statements maintain context.

To modify list items, you can put the MODIFY statement in nested FOR statements. The following example modifies the PRICE field in the VENDORS list for the ABC SUPPLY record:

```
DTR> READY SUPPLIES MODIFY
DTR> SET NO PROMPT
DTR> FOR FIRST 1 SUPPLIES
CON>     FOR FIRST 1 VENDORS
CON>         PRINT THEN MODIFY PRICE

                      MIN              WE
 COMPANY    PRICE   ORDER PERCENT    PAY

ABC SUPPLY $13.89    150     3%    $13.47
                     500     6%    $13.05
                    1000     8%    $12.77
Enter PRICE: 13.99

DTR>
```

To modify a field in the DISCOUNT list, include another FOR statement to establish context for that list. The following example displays the specified list element, changes the MIN-ORDER of 150 in the DISCOUNT list to 200, and displays the modified record by joining PRINT and MODIFY statements with THEN statements:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES MODIFY
DTR> FOR FIRST 1 SUPPLIES
CON>     FOR VENDORS WITH COMPANY CONT "ABC"
CON>         FOR DISCOUNT WITH MIN-ORDER EQ 150
CON>         PRINT THEN MODIFY MIN-ORDER THEN PRINT

 MIN             WE
ORDER PERCENT   PAY

  150     3%    $13.57
Enter MIN_ORDER: 200

 MIN             WE
ORDER PERCENT   PAY

  200     3%    $13.57

DTR>
```

To modify all three values in the MIN-ORDER field, do not include the RSE in the third FOR statement. PRO/DATATRIEVE then prompts three times for MIN-ORDER values:

```
DTR> FOR FIRST 1 SUPPLIES
CON>     FOR VENDORS WITH COMPANY CONT "ABC"
CON>         FOR DISCOUNT
CON>         PRINT THEN MODIFY MIN-ORDER THEN PRINT

 MIN             WE
ORDER PERCENT   PAY

  200     3%    $13.57
Enter MIN_ORDER: 210

 MIN             WE
ORDER PERCENT   PAY

  210     3%    $13.57
  500     6%    $13.15
Enter MIN_ORDER: 510
  510     6%    $13.15
 1000     8%    $12.87
Enter MIN_ORDER: 1100
 1100     8%    $12.87

DTR>
```

You can modify fields in both the VENDORS list and the DISCOUNT list by putting the PRINT statement that modifies the VENDORS list and the FOR statement that identifies the DISCOUNT list in a BEGIN-END statement. The following example modifies the PRICE field for the MORRIS company, then changes the MIN-ORDER value from 10 to 25:

```
DTR> READY SUPPLIES MODIFY
DTR> SET NO PROMPT
DTR> FOR FIRST 1 SUPPLIES
CON>     FOR VENDORS WITH COMPANY EQ "MORRIS"
CON>     BEGIN
CON>        PRINT THEN MODIFY PRICE
CON>        FOR DISCOUNT WITH MIN-ORDER EQ 10
CON>            PRINT THEN MODIFY MIN-ORDER
CON>     END

                      MIN              WE
 COMPANY    PRICE   ORDER PERCENT    PAY

MORRIS      $12.98     10     2%    $12.72
                       50     4%    $12.46
                      100     6%    $12.20
Enter PRICE: 13.03

 MIN              WE
ORDER PERCENT   PAY

   10     2%    $12.76
Enter MIN_ORDER: 25

DTR>
```

Using the PRINT Statement

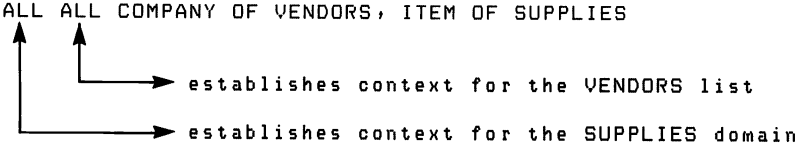
You can also establish context for lists by including a list specification and the keyword ALL in the PRINT statement. A list specification has this format:

```
ALL [print-list-element],... OF list-rse
```

- *Print-list-elements* identify the fields in the list that you want to display and specify how to format the fields. See Chapter 11 for information on formatting displays with the PRINT statement. Use commas to separate print list elements.
- *List-rse* identifies the list that contains the fields you want to display. If you do not specify any fields as print list elements, PRO/DATATRIEVE displays all fields in the list field.

You *must* include the keyword ALL before the list specification to establish context for the list:

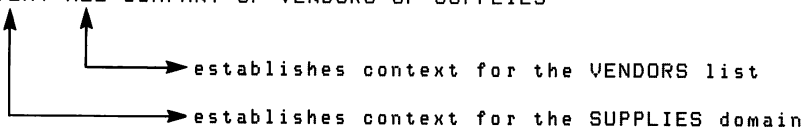
```
DTR> PRINT ALL ALL COMPANY OF VENDORS, ITEM OF SUPPLIES
```



This PRINT statement displays the COMPANY field in the VENDORS list, then the ITEM field from the record. The list RSE, OF VENDORS, identifies the list that contains the COMPANY field. The domain RSE, OF SUPPLIES, identifies the domain that contains the ITEM field.

If you specify the ITEM field first, you need only one ALL to establish context for the list. The ITEM field is not a list field, so it establishes context for the domain:

```
DTR> PRINT ITEM, ALL COMPANY OF VENDORS OF SUPPLIES
```



The list specification in the following example displays all VENDORS for PAPER:

```
DTR> READY SUPPLIES
DTR> SET NO PROMPT
DTR> PRINT ALL ALL VENDORS OF SUPPLIES WITH ITEM EQ "PAPER"

                    MIN              WE
 COMPANY    PRICE  ORDER PERCENT    PAY

ABC SUPPLY $13.99    200     3%   $13.57
                     500     6%   $13.15
                    1000     8%   $12.87
MORRIS     $13.03     25     2%   $12.76
                      50     4%   $12.50
                     100     6%   $12.24
XYZ DATA   $13.00    100     2%   $12.74
                     500     4%   $12.48
                    1000     7%   $12.09

DTR>
```

The first ALL establishes context for the list and is associated with OF SUPPLIES WITH ITEM EQ "PAPER". The second ALL is part of the list specification and is associated with VENDORS to establish the list field as the source of information.

The key to using the PRINT statement to display list fields is to remember to specify ALL and an outer RSE to establish a context for the list, then to specify ALL and a corresponding RSE for each list that contains the information you want to display.

To display only selected fields in the list field, specify the fields you want to see in the inner RSE:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> PRINT ALL ALL COMPANY, DISCOUNT OF VENDORS WITH
CON>       COMPANY CONT "ABC" OF SUPPLIES WITH ITEM EQ "PAPER"

             MIN              WE
 COMPANY    ORDER PERCENT    PAY

ABC SUPPLY    200     3%    $13.57
              500     6%    $13.15
             1000     8%    $12.87

DTR>
```

To display the COMPANY field and only two of the fields in the DISCOUNT list, include another list specification in the PRINT statement to establish context for the inner list:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> PRINT ALL ALL COMPANY, ALL PERCENT, WE-PAY OF DISCOUNT
CON>       OF VENDORS OF SUPPLIES WITH ITEM EQ "PAPER"

                     WE
 COMPANY    PERCENT  PAY

ABC SUPPLY     3%   $13.57
               6%   $13.15
               8%   $12.87
MORRIS         2%   $12.76
               4%   $12.50
               6%   $12.24
XYZ DATA       2%   $12.74
               4%   $12.48
               7%   $12.09

DTR>
```

To display the COMPANY field on every PERCENT line, put the ALL that identifies the DISCOUNT list before the COMPANY field:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> PRINT ALL ALL ALL COMPANY, PERCENT, WE-PAY OF DISCOUNT
CON>       OF VENDORS OF SUPPLIES WITH ITEM EQ "PAPER"

                      WE
 COMPANY    PERCENT   PAY

ABC SUPPLY     3%    $13.57
ABC SUPPLY     6%    $13.15
ABC SUPPLY     8%    $12.87
MORRIS         2%    $12.76
MORRIS         4%    $12.50
MORRIS         6%    $12.24
XYZ DATA       2%    $12.74
XYZ DATA       4%    $12.48
XYZ DATA       7%    $12.09

DTR>
```

To change the order of the display so that the COMPANY field is displayed after the WE-PAY field, change the order of the print list:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> PRINT ALL ALL ALL DISCOUNT, COMPANY, SKIP 2 OF
CON>       VENDORS OF SUPPLIES WITH ITEM EQ "PENS"

 MIN              WE
ORDER PERCENT    PAY     COMPANY

   12     1%    $3.17
   50     2%    $3.14
  500     4%    $3.08 ABC SUPPLY

  100     2%    $2.90
  500     3%    $2.87
 1000     4%    $2.84 MORRIS

   10     1%    $3.08
  100     3%    $3.02
  500     5%    $2.96 HARRIDON


DTR>
```

Notice that this example uses the SKIP keyword to insert a blank line between companies. Chapter 11 explains how to use SKIP and other formatting keywords (SPACE and COL) to format list displays.

To display specific information, restrict the list fields in the list RSEs:

```
DTR> READY SUPPLIES
DTR> SET NO PROMPT
DTR> PRINT ALL ALL COMPANY, ALL PERCENT, WE-PAY OF
CON>       DISCOUNT WITH PERCENT LE 3 OF
CON>       VENDORS OF SUPPLIES WITH ITEM EQ "PENS"

                      WE
 COMPANY    PERCENT   PAY

ABC SUPPLY     1%     $3.17
               2%     $3.14
MORRIS         2%     $2.90
               3%     $2.87
HARRIDON       1%     $3.08
               3%     $3.02

DTR>
```

CREATING LIST REPORTS

You can create reports from lists by including a list specification in the Report Writer PRINT statement:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> REPORT SUPPLIES
RW>  SET COLUMNS-PAGE = 50
RW>  PRINT ITEM, ALL COMPANY, ALL MIN-ORDER, WE-PAY OF
RW>       DISCOUNT OF VENDORS
RW>  END-REPORT

                                          14-Jul-83
                                          Page 1

                                 MIN        WE
        ITEM          COMPANY  ORDER      PAY

   PAPER              ABC SUPPLY   150    $13.04
                                   500    $12.77
                                  1000    $12.50
                      MORRIS        10    $12.72
                                    50    $12.46
                                   100    $12.20
                      XYZ DATA      100     $3.04
                                    500     $2.98
                                   1000     $2.89
```

(continued on next page)

```
   PENS                ABC SUPPLY       12     $3.17
                                        50     $3.14
                                       500     $3.08
                       MORRIS          100     $2.90
                                       500     $2.87
                                      1000     $2.84
                       HARRIDON         10     $3.08
                                       100     $3.02
                                       500     $2.96

DTR>
```

To display the same information with a PRINT statement typed in response to the DTR> prompt, you need to include an RSE to identify the domain, like this:

```
DTR> PRINT ITEM, ALL COMPANY, ALL MIN-ORDER, WE-PAY OF
CON>       DISCOUNT OF VENDORS OF SUPPLIES
                            .
                            .
                            .
DTR>
```

In the Report Writer PRINT statement, you do not have to include an RSE to identify the domain. The REPORT statement establishes the record stream for the report, providing an implied OF SUPPLIES for the PRINT statement.

CHANGING THE LENGTH OF A LIST

You cannot change the length of a fixed-length list such as DISCOUNT, but you can change the length of a variable-length list such as VENDORS. The length of the VENDORS list depends on the value stored in the HOW-MANY field, while the DISCOUNT list can occur only three times as specified in the record definition:

```
05 HOW-MANY PIC 9.
05 VENDORS OCCURS 0 TO 10 TIMES DEPENDING ON HOW-MANY.
   07 COMPANY PIC X(10).
   07 PRICE PIC 99V99
      EDIT-STRING IS $$$.$$.
   07 DISCOUNT OCCURS 3 TIMES.
```

If you want to add an item to the VENDORS list, you must first modify the HOW-MANY field:

```
DTR> READY SUPPLIES MODIFY
DTR> SET NO PROMPT
DTR> FOR SUPPLIES WITH ITEM EQ "PAPER"
CON>     PRINT THEN MODIFY HOW-MANY

                  ORDER     HOW                              MIN              WE
     ITEM          UNIT     MANY   COMPANY     PRICE      ORDER  PERCENT    PAY

PAPER             REAM        3   ABC SUPPLY  $13.99       200      3%   $13.57
                                                           500      6%   $13.15
                                                          1000      8%   $12.87
                                    MORRIS      $13.03       25      2%   $12.76
                                                            50      4%   $12.50
                                                           100      6%   $12.24
                                    XYZ DATA    $13.00       100      2%   $12.74
                                                           500      4%   $12.48
                                                          1000      7%   $12.09

Enter HOW_MANY: 4

DTR> PRINT SUPPLIES WITH ITEM EQ "PAPER"

                  ORDER     HOW                              MIN              WE
     ITEM          UNIT     MANY   COMPANY     PRICE      ORDER  PERCENT    PAY

PAPER             REAM        4   ABC SUPPLY  $13.99       200      3%   $13.57
                                                           500      6%   $13.15
                                                          1000      8%   $12.87
                                    MORRIS      $13.03       25      2%   $12.76
                                                            50      4%   $12.50
                                                           100      6%   $12.24
                                    XYZ DATA    $13.00       100      2%   $12.74
                                                           500      4%   $12.48
                                                          1000      7%   $12.09
                                                                    0%
                                                                    0%
                                                                    0%

DTR>
```

The PRINT statement shows that PRO/DATATRIEVE has added a blank list item to the VENDORS list. By using FIND and SELECT statements to establish context for the blank fields, you can then modify the fields to add the new paper vendor:

```
DTR> FIND SUPPLIES WITH ITEM EQ "PAPER"
[1 Record found]
DTR> SELECT
DTR> FIND VENDORS
[4 records found]
DTR> SELECT 4
DTR> PRINT

                    MIN              WE
 COMPANY    PRICE  ORDER PERCENT    PAY

                           0%
                           0%
                           0%

DTR> MODIFY
Enter COMPANY: NEW DATA
Enter PRICE: 13.55
Enter MIN_ORDER: 100
Enter PERCENT: 3
Enter MIN_ORDER: 500
Enter PERCENT: 5
Enter MIN_ORDER: 1000
Enter PERCENT: 8
DTR> PRINT

                    MIN              WE
 COMPANY    PRICE  ORDER PERCENT    PAY

NEW DATA   $13.55    100     3%    $13.14
                     500     5%    $12.87
                    1000     8%    $12.46

DTR>
```

You can delete the last item in a variable-length list by changing the length of the list. The following example changes the HOW-MANY field in the PAPER record

back to 3 and displays the record to show that PRO/DATATRIEVE deletes the NEW DATA vendor:

```
DTR> PRINT SUPPLIES WITH ITEM EQ "PAPER"

                ORDER     HOW                              MIN                WE
     ITEM        UNIT     MANY  COMPANY     PRICE       ORDER  PERCENT      PAY

PAPER           REAM        4  ABC SUPPLY  $13.99        200       3%    $13.57
                                                          500       6%    $13.15
                                                         1000       8%    $12.87
                                MORRIS      $13.03         25       2%    $12.76
                                                           50       4%    $12.50
                                                          100       6%    $12.24
                                XYZ DATA    $13.00        100       2%    $12.74
                                                          500       4%    $12.48
                                                         1000       7%    $12.09
                                NEW DATA    $13.55        100       3%    $13.14
                                                          500       5%    $12.87
                                                         1000       8%    $12.46

DTR> FOR SUPPLIES WITH ITEM EQ "PAPER"
CON>     MODIFY HOW-MANY
Enter HOW_MANY: 3
DTR> PRINT SUPPLIES WITH ITEM EQ "PAPER"

                ORDER     HOW                              MIN                WE
     ITEM        UNIT     MANY  COMPANY     PRICE       ORDER  PERCENT      PAY

PAPER           REAM        3  ABC SUPPLY  $13.99        200       3%    $13.57
                                                          500       6%    $13.15
                                                         1000       8%    $12.87
                                MORRIS      $13.03         25       2%    $12.76
                                                           50       4%    $12.50
                                                          100       6%    $12.24
                                XYZ DATA    $13.00        100       2%    $12.74
                                                          500       4%    $12.48
                                                         1000       7%    $12.09

DTR>
```

SUMMARY

The key to working with list fields is establishing context for the list. You can establish context for list fields with FIND and SELECT statements, nested FOR statements, and PRINT statements. This chapter showed how to display and modify list fields, how to include list fields in a report, and how to change the length of a variable-length list.

For more information on commands and statements used in this chapter, refer to the following chapters:

- See Chapter 8 for information on defining list fields.
- See Chapter 11 for information on formatting information with the PRINT statement and the Report Writer.
- See Chapter 12 for information on the MODIFY statement.
- See Chapter 13 for information on the FOR statement.
- See Chapter 14 for information on the FIND and SELECT statements.

5

Restructuring a Database

Chapter 5

Restructuring a Database

After you have defined a database, you may decide that you want to add, modify, or reorganize fields in the record definition or that you want to change the organization of the data file.

If you simply want to add or change a COMPUTED BY, QUERY-NAME, QUERY-HEADER, or EDIT-STRING clause, you do not have to go through the restructuring process. You also do not have to restructure if you want to add a group field to contain existing elementary fields.

If, however, you want to make any other kind of changes to the record definition, you must go through the restructuring process, or you may lose information in your data file. If you want to add or change index keys in an indexed file, or if you want to change the organization of the data file, you must also go through the restructuring process.

MAKING SIMPLE CHANGES TO A RECORD DEFINITION

To add or change COMPUTED BY, QUERY-NAME, QUERY-HEADER, or EDIT-STRING clauses, or to add a group field that contains existing elementary fields, take the following steps:

1. If you have readied the domain that uses the record definition you want to modify, use the FINISH command to finish the domain.

2. You can edit the record definition with the PRO/DATATRIEVE Editor or use the EXTRACT command to copy the record definition to a command file. You can then exit from PRO/DATATRIEVE and edit the command file with your PROSE editor. If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message.

Chapter 17 explains how to use the PRO/DATATRIEVE Editor. Chapter 16 explains how to copy existing dictionary definitions to a command file. Chapter 10 explains how to execute command files. The next time you ready the domain, PRO/DATATRIEVE uses the modified record definition. This type of modification of a record definition does not affect the way PRO/DATATRIEVE stores data in the record, the size of the record,or the data file structure.

MAKING STRUCTURAL CHANGES TO A RECORD DEFINITION

Structural changes alter the size of the record, the data type of record fields, the order of fields in the record, validation criteria for fields, and file characteristics. These changes include:

- Adding one or more fields to the record
- Changing the length or data type of a field
- Changing the order of the fields in the record
- Changing or adding validation clauses
- Changing the type of data file from indexed to sequential or sequential to indexed
- Changing an index key or index key attributes (duplicate, change)

Instead of creating a new database, you can make structural changes to an existing database by restructuring. The purpose of restructuring is to change the record definition or the file attributes without changing anything else. Ideally, the new database should have the same domain name, the same record name, most of the same field names and characteristics, and most of the data that the old database had. This way, you do not have to change procedures that operate on the database.

You can restructure a database safely and efficiently, keeping the original domain, record, and file names, by following the steps outlined here and described in the following sections:

1. Define a temporary database identical to the original database. Use a different domain name (TEMP), a different record name (TEMP-REC), and a different data file name (TEMP.DAT).
2. Copy records from the original domain to the temporary domain.
3. Copy the original record definition from the dictionary to a command file and exit from PRO/DATATRIEVE. Use PROSE to edit the command file and make changes to the record definition. Return to PRO/DATATRIEVE and execute the command file to store the modified record definition.
4. Define a new file for the original domain and copy records from the temporary domain to the restructured domain. When you define the new file, you can change the structure of the file or the attributes of index keys. Do not use a new field as an index key.
5. Display records from the restructured domain to confirm the successful transfer of records. Then delete the temporary domain and record definition from your dictionary.

This chapter takes you through the process of restructuring the CUSTOMERS domain you defined with ADT in *PRO/DATATRIEVE for Beginners*. If you did not define this domain, you can restructure another domain, if you like.

The modified CUSTOMERS-REC record definition has:

- A modified COMPANY-NAME field that includes a VALID IF clause to validate a company value from a table
- A group field called ADDRESS that contains the existing elementary fields STREET, CITY, STATE, and ZIP-CODE
- A modified PHONE-NUMBER field with a different length, data type, and edit string clause
- A new field called LAST-PURCHASE to hold the date of the last purchase the customer made

Note that if you wanted to add only the ADDRESS group field, you would not need to restructure the domain. You could simply edit the record definition and add the group field. To make the other changes, however, you must restructure the domain.

Creating the Temporary Database

Use the DEFINE DOMAIN, EXTRACT, and DEFINE FILE commands to create the temporary database.

1. Define the domain:

```
DTR> DEFINE DOMAIN TEMP USING TEMP-REC ON TEMP;
DTR>
```

2. To create the temporary record, use the EXTRACT command to copy the CUSTOMERS-REC definition to a command file:

```
DTR> EXTRACT ON TEMP CUSTOMERS-REC
DTR>
```

   The EXTRACT command copies CUSTOMERS-REC to TEMP.CMD, inserting a DELETE command at the top of the file. Do not execute this command file yet.

3. Exit from PRO/DATATRIEVE and use PROSE to edit the command file, changing CUSTOMERS-REC to TEMP-REC in the DELETE command and in the DEFINE RECORD command. Your command file should then look like this:

```
DELETE TEMP_REC;
DEFINE RECORD TEMP_REC
 USING
01 CUSTOMERS_REC.
   15 COMPANY_NAME       PIC IS X(20)
      QUERY_NAME IS COMPANY.
   15 CONTACT_PERSON     PIC IS X(15)
      QUERY_NAME IS CONTACT.
   15 STREET     PIC IS X(15).
   15 CITY       PIC IS X(10).
   15 STATE      PIC IS X(2).
   15 ZIP_CODE   PIC IS X(5)
      QUERY_NAME IS ZIP.
   15 PHONE_NUMBER       PIC IS X(12)
      QUERY_NAME IS PHONE.
   15 SALES_THIS_YEAR    PIC IS  S9(5)V99
      EDIT_STRING IS $$$,$$$.$$
      QUERY_NAME IS SALES.
   15 LAST_CONTACT       USAGE IS DATE
      EDIT_STRING IS  DD-MMM-YY
      QUERY_NAME IS LAST_C.
;
```

Be sure to change both occurrences of CUSTOMERS-REC. If you do not change it in the DELETE command, PRO/DATATRIEVE deletes that record when you execute the command file. If you do not change CUSTOMERS-REC in the DEFINE RECORD command, PRO/DATATRIEVE stores the record definition with that name and your TEMP domain will not have a record.

4. Select PRO/DATATRIEVE from the Main Menu and execute TEMP.CMD by typing an at sign (@) and the file name. You do not have to specify the .CMD file type because PRO/DATATRIEVE automatically searches for a .CMD file.

5. Use the DEFINE FILE command to define a data file for the domain:

```
DTR> DEFINE FILE FOR TEMP;
DTR>
```

This command defines a sequential file for the TEMP domain. It does not matter whether the file is indexed or sequential. You are using it just as a temporary file and will not be modifying or deleting records.

Copying Records to the Temporary Database

Now copy records from the CUSTOMERS domain to the TEMP domain by following these steps:

1. Ready the CUSTOMERS domain for READ access so that you cannot accidentally write to the domain.

2. Ready the TEMP domain for WRITE access so that you can store records in it.

3. Use a STORE statement in a FOR loop to write all records in the CUSTOMERS domain to the TEMP domain. The record definitions for both domains are identical, so use a context variable in the FOR statement. This differentiates the top-level field of CUSTOMERS-REC (CUSTOMERS-REC) from the top-level field of TEMP-REC (CUSTOMERS-REC).

4. Display the COMPANY and LAST-CONTACT fields for two records from the TEMP domain to confirm the transfer of records. Use the FINISH command to finish both domains.

These steps translate into the following sequence of PRO/DATATRIEVE commands and statements:

```
DTR> SET NO PROMPT
DTR> READY CUSTOMERS READ
DTR> READY TEMP WRITE
DTR> FOR A IN CUSTOMERS
CON>    STORE TEMP USING CUSTOMERS-REC = A.CUSTOMERS-REC
DTR> PRINT COMPANY, LAST-CONTACT OF FIRST 2 TEMP

      COMPANY          LAST
        NAME          CONTACT

BASIC SYSTEMS        18-May-83
INFO ANALYSIS         2-Mar-83

DTR> FINISH
DTR>
```

Extracting and Modifying the Original Record Definition

Now that the original records are safely stored in the TEMP domain, you can modify the original record definition. Use the EXTRACT command again, this time to copy the CUSTOMERS-REC definition to a command file named CUST:

```
DTR> EXTRACT ON CUST CUSTOMERS-REC
DTR>
```

This EXTRACT command copies CUSTOMERS-REC to CUST.CMD, inserting a DELETE command at the top of the file. As you did before, exit from PRO/DATATRIEVE and use PROSE to edit the command file to make the following changes to the record definition:

① Add a VALID IF clause to the COMPANY-NAME field to validate company values from the COMPANIES table.

② Convert the address information into a group field called ADDRESS. To do this, insert an ADDRESS field with a field level number of 15 just before the STREET field. Then change the field level numbers for the STREET, CITY, STATE, and ZIP-CODE fields to a number higher than 15. All four elementary fields should have the same field level number. To make the record more understandable, indent the elementary fields to show that they are subordinate to the ADDRESS group field.

③ Change the data type of the PHONE-NUMBER field from characters to numbers by changing the X to a 9. Change the length of the field from 12 to 10. Add an EDIT-STRING clause so that the phone number prints as XXX-XXX-XXXX. The original data file stored the phone number as a character string with numbers separated by hyphens. By redefining the field as numbers, PRO/DATATRIEVE discards the hyphens stored in the field when transferring data from the field in TEMP-REC to the redefined field in CUSTOMERS-REC. The EDIT-STRING clause then tells PRO/DATATRIEVE how to display phone numbers.

④ Add a new field, LAST-PURCHASE, after the SALES-THIS-YEAR field. This field should have a USAGE IS DATE clause. If you like, include an EDIT-STRING clause and a QUERY-NAME clause.

Be careful not to change the record name in the DELETE or DEFINE RECORD command. When you finish modifying the record definition, your command file should look like this:

```
DELETE CUSTOMERS_REC;
DEFINE RECORD CUSTOMERS_REC
 USING
01 CUSTOMERS_REC.
   15 COMPANY_NAME  PIC IS X(20)
      VALID IF COMPANY_NAME IN COMPANIES                    <-- (1)
      QUERY_NAME IS COMPANY.
   15 CONTACT_PERSON          PIC IS X(15)
      QUERY_NAME IS CONTACT.
   15 ADDRESS.                                              <-- (2)
      25 STREET      PIC IS X(15).
      25 CITY        PIC IS X(10).
      25 STATE       PIC IS X(2).
      25 ZIP_CODE    PIC IS X(5)
         QUERY_NAME IS ZIP.
   15 PHONE_NUMBER  PIC IS 9(10)                            <-- (3)
      EDIT-STRING IS XXX-XXX-XXXX
      QUERY_NAME IS PHONE.
   15 SALES_THIS_YEAR         PIC IS  S9(5)V99
      EDIT_STRING IS $$$,$$$.$$
      QUERY_NAME IS SALES.
   15 LAST-PURCHASE USAGE IS DATE                           <-- (4)
      EDIT-STRING IS DD-MMM-YY
      QUERY-NAME IS LAST_P.
   15 LAST_CONTACT  USAGE IS DATE
      EDIT_STRING IS  DD-MMM-YY
      QUERY_NAME IS LAST_C.
;
```

When your command file looks like this, select PRO/DATATRIEVE from the Main Menu and type @CUST. After PRO/DATATRIEVE executes the command file, you see the message "Record CUSTOMERS_REC is 100 bytes long" and the DTR> prompt, indicating that the modified record definition is now stored in your current dictionary.

Creating the Restructured Database

If you try to ready the CUSTOMERS domain at this point, PRO/DATATRIEVE displays an error message because the size of the new record does not match the size of the original record. Use the DEFINE FILE command to define a new data file to hold the larger records. Create an indexed file and name the COMPANY field as the primary index key by including a key clause:

```
DTR> DEFINE FILE FOR CUSTOMERS KEY = COMPANY
DTR>
```

Now copy records from the TEMP domain to the restructured CUSTOMERS domain. This time, though, ready the CUSTOMERS domain for WRITE access and the TEMP domain for READ access:

```
DTR> SET NO PROMPT
DTR> READY CUSTOMERS WRITE
DTR> READY TEMP READ
DTR> FOR A IN TEMP
CON>     STORE CUSTOMERS USING CUSTOMERS-REC = A.CUSTOMERS-REC
Illegal placement of minus sign in string "617-555-8495"
Illegal placement of minus sign in string "617-555-8495"
Illegal placement of minus sign in string "617-555-7114"
Illegal placement of minus sign in string "617-555-7114"
Illegal placement of minus sign in string "603-555-0570"
Illegal placement of minus sign in string "603-555-0570"
Illegal placement of minus sign in string "617-555-9981"
Illegal placement of minus sign in string "617-555-9981"
Illegal placement of minus sign in string "603-555-7281"
Illegal placement of minus sign in string "603-555-7281"
Illegal placement of minus sign in string "603-555-3886"
Illegal placement of minus sign in string "603-555-3886"
Illegal placement of minus sign in string "603-555-9449"
Illegal placement of minus sign in string "603-555-9449"
Illegal placement of minus sign in string "617-555-4423"
Illegal placement of minus sign in string "617-555-4423"
Illegal placement of minus sign in string "617-555-3723"
Illegal placement of minus sign in string "617-555-3723"
Illegal placement of minus sign in string "603-555-5085"
Illegal placement of minus sign in string "603-555-5085"
DTR>
```

The error messages indicate that PRO/DATATRIEVE interprets each hyphen in the PHONE-NUMBER field as a minus sign and that it discards minus signs from the field values.

Confirming the Restructuring Process

To confirm the successful transfer of data, type PRINT CUSTOMERS. To see just the new or modified fields, specify those fields in a PRINT statement:

```
DTR> PRINT ADDRESS, PHONE, LAST-PURCHASE OF CUSTOMERS

                                        ZIP      PHONE        LAST
    STREET          CITY      STATE CODE      NUMBER     PURCHASE

43 HIGHLAND AVE BOSTON        MA    13532 617-555-8495
9 REDACTOR      DUNKIRK       MA          617-555-7114
                              .
                              .
                              .
DTR>
```

PRO/DATATRIEVE displays all the new and modified fields successfully, so use the DELETE command to delete the TEMP domain and TEMP-REC record definition from your dictionary:

```
DTR> DELETE TEMP;
DTR> DELETE TEMP-REC;
DTR>
```

To store values in the new field, LAST-PURCHASE, use a FOR statement to establish a record stream consisting of all records in the domain. Put a PRINT statement and a MODIFY statement in a BEGIN-END statement in the FOR loop:

```
DTR> SET NO PROMPT
DTR> FOR CUSTOMERS WITH LAST-PURCHASE EQ " "
CON> BEGIN
CON>    PRINT COMPANY, SALES
CON>    MODIFY LAST-PURCHASE
CON> END

                         SALES
      COMPANY             THIS
        NAME              YEAR

BASIC SYSTEMS
Enter LAST_PURCHASE:
```

PRO/DATATRIEVE displays the fields you specify and then prompts for a value to store in the LAST-PURCHASE field for each record in the domain. Confirm the modifications with a PRINT statement and finish the domain with the FINISH command.

After you exit from PRO/DATATRIEVE, you may want to clean up your directory by using File Services to delete the temporary data file, the old version of the restructured data file, and the two command files, TEMP.CMD and CUST.CMD.

SUMMARY

This chapter described how to make simple changes to a record definition and how to restructure a data base to make structural and organizational changes. The section on readying a domain in Chapter 9 describes an alternative way to restructure a database by readying the domain with an alias.

For more information on using context variables, see Chapter 3. For more information on commands and statements used in this chapter, refer to the following chapters:

- See Chapter 10 for information on executing command files.
- See Chapter 13 for information on BEGIN-END and FOR statements.
- See Chapter 16 for information on using the DELETE and EXTRACT commands.
- See Chapter 17 for information on using the PRO/DATATRIEVE Editor.

Part 2

Functional Reference Section

This section describes how to use PRO/DATATRIEVE statements and commands to manage data. Decide what you want to do and go to the appropriate chapter, just as you would make a selection from the Main Menu:

6

Getting Help and Monitoring PRO/DATATRIEVE

Chapter 6

Getting Help and Monitoring PRO/DATATRIEVE

This chapter describes commands you can use to get help and to monitor your dialogue with PRO/DATATRIEVE:

Using Guide Mode
Getting Help
Turning Prompts Off and On
Monitoring a PRO/DATATRIEVE Session

By using the commands described in this chapter, you can use Guide Mode to help form PRO/DATATRIEVE statements and commands, get help on using statements and commands, activate or suppress helpful prompts like "Looking for Boolean expression", and create a log file of a PRO/DATATRIEVE session.

USING GUIDE MODE

Use the SET GUIDE command to start Guide Mode. Guide Mode helps you work step-by-step through a PRO/DATATRIEVE session and saves you typing time by providing additional prompts and information as you type in commands and statements.

Format

```
SET GUIDE
```

Explanation

After you type SET GUIDE, you see the following line:

```
Enter command, type ? for help
```

You can type a question mark or press the HELP key to see that your options at that point are READY, SHOW, and LEAVE.

Guide Mode helps you make entries by letting you know what your options are. It can also complete your entries so you do not have to type the entire command or statement. All you have to do is type enough letters to uniquely identify the word you are typing, like R, S, or L. Guide Mode then completes typing the word for you. Your possible options change as you go along, depending on where you are in creating a command or statement. Note that not all PRO/DATATRIEVE commands and statements are available in Guide Mode.

You can exit from Guide Mode at any time by pressing the EXIT or MAIN SCREEN key. You can also type L (for LEAVE) when you see the "Ready for next command" prompt. When you exit from Guide Mode, PRO/DATATRIEVE displays the message "You're on your own now. Good luck!" and returns you to the DTR> prompt.

Comments

Throughout, Guide Mode prompts you for the next part of your entry, including spaces, quotation marks, and returns. You can press DO or RETURN wherever Guide Mode tells you you can press RETURN.

By pressing the HELP key or typing a question mark, you can find out which commands, statements, keywords, or symbols you can use at that time. PRO/DATATRIEVE displays an error message if you try to use a word or symbol that Guide Mode does not recognize.

If you want to return directly from Guide Mode to the Main Menu, press the INTERRUPT key and then the DO key.

The SET GUIDE command sets the scrolling attribute of your Professional screen to smooth scroll. When you leave Guide Mode, PRO/DATATRIEVE continues the smooth scrolling. When you exit from PRO/DATATRIEVE, the scrolling attribute is set to your original default.

Guide Mode is primarily a training tool. You may also find it useful to save keystrokes while typing. Only you can judge when to use Guide Mode to help you use PRO/DATATRIEVE.

Example

You can learn about Guide Mode best by using it. Type SET GUIDE and press the HELP key or type a question mark to see what your options are at that time. When you finish using Guide Mode, press the MAIN SCREEN or EXIT key to return to the DTR> prompt.

GETTING HELP

To get information about using PRO/DATATRIEVE on-line Help, press the HELP key. PRO/DATATRIEVE displays information on the type of help available. To get information on a particular topic, use the HELP command.

Format

```
HELP [ADVANCED] TOPIC,...
```

Explanation

If you type HELP without specifying a topic or if you press the HELP key, PRO/DATATRIEVE displays information about using the HELP command to get help.

Topic is a PRO/DATATRIEVE command, statement, or clause. To see a list of available topics, type HELP HELP. Then choose one or more topics and type HELP and the topic names. Use commas to separate each topic from the next.

ADVANCED indicates that you want help on advanced PRO/DATATRIEVE language elements and concepts. To see a list of topics for which Advanced Help is available, type HELP ADVANCED. Then choose one or more topics and type HELP ADVANCED and the topic names.

Comments

HELP messages for each topic contain on-line information about using and formatting a particular command or statement. To stop the screen from scrolling while you read all the Help display, press the HOLD SCREEN key.

When PRO/DATATRIEVE finishes displaying the Help message or messages you requested, you return to the DTR> prompt.

ADVANCED, when specified, applies to all topics you list in the HELP command. For example, you get Advanced Help on both WOMBAT and HELP if you specify HELP ADVANCED WOMBAT, HELP.

Example

Press the HELP key to display information on using PRO/DATATRIEVE Help, read the information, then display a list of available topics by typing HELP HELP. Then use the HELP command to get both basic and advanced information on using the FIND command. Use the HOLD SCREEN key to stop and start the display:

```
DTR> HELP FIND, ADVANCED FIND
The FIND statement is used to establish a collection of
records from a domain, a list or a previously established
collection. The forms of the statement are:

              FIND domain-name WITH condition

                FIND CURRENT WITH condition

where "domain-name", if used, must be the name of a readied
domain (see HELP READY) and "condition" is a conditional
expression.

The FIND statement establishes a collection of the records
from the domain, the list or the current collection that
satisfy the given condition. The number of records found
will be printed on your terminal.

Examples:

     FIND YACHTS WITH LENGTH-OVER-ALL BETWEEN 26 AND 30

     FIND CURRENT WITH PRICE < 15000
The full form of the FIND statement is

                          FIND rse

where "rse" is a record selection expression. If "rse"
contains a name for the new collection, the resulting
collection is established under that name.
DTR>
```

TURNING PROMPTS OFF AND ON

Use the SET NO PROMPT and SET PROMPT commands to turn automatic prompting for continued statements and commands off and on. When you first enter PRO/DATATRIEVE, SET PROMPT is in effect. Thus, when you press DO before finishing a command or statement, PRO/DATATRIEVE prompts you for the next element of that statement or command with a line like [Looking for Dictionary Element].

Format

```
SET [NO] PROMPT
```

Explanation

SET NO PROMPT turns automatic prompting off. SET PROMPT turns automatic prompting back on again.

Comments

Automatic prompting for command and statement elements can be helpful when you are first learning to use PRO/DATATRIEVE. However, if you do not need help, turning automatic prompting off makes your screen less cluttered and complex statements and commands easier to read.

Example

Type a partial READY command and then a partial FIND statement to see the PRO/DATATRIEVE prompts, then type SET NO PROMPT and type a partial PRINT statement:

```
DTR> READY
[Looking for Dictionary Element]
CON> SALES-VIEW
DTR> FIND SALES-VIEW WITH
[Looking for Boolean expression]
CON>       LAST-CONTACT LE "2-15-83"
[1 Record found]
DTR> SET NO PROMPT
DTR> PRINT FIRST 1 SALES-VIEW WITH
CON>       LAST-CONTACT LE "2-15-83"

                                    SALES
      COMPANY                        THIS
        NAME          LAST          YEAR
                    CONTACT

KEY SPECIALISTS     11-Dec-82   $1,025.00

DTR>
```

MONITORING A PRO/DATATRIEVE SESSION

You can create a record of a PRO/DATATRIEVE session by using the OPEN command to create a log file of your input and all PRO/DATATRIEVE activity and displays. A log file records your interactive dialogue with PRO/DATATRIEVE. When you are finished monitoring the session, use the CLOSE command to close the file.

Format

```
OPEN file-spec
      .
      .
      .
CLOSE
```

Explanation

File-spec is the file specification of the log file you want PRO/DATATRIEVE to keep.

If you want the file to reside in your current directory, you need specify only a file name. PRO/DATATRIEVE creates the file in the current directory with the name you specify and a file type of .LST. Like all Professional file names, a log file name can consist of from one to nine letters or numbers.

To create a log file in a different directory or on a different device, or to specify a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

You can have only one log file open at a time. If you enter a second OPEN command without closing the first log file, PRO/DATATRIEVE ignores the second OPEN command.

To close a log file without exiting from PRO/DATATRIEVE, use the CLOSE command. If you do not specifically close the file with the CLOSE command before you return to the Main Menu, PRO/DATATRIEVE closes the file when you exit from PRO/DATATRIEVE.

Comments

Do not try to create a log file of a Guide Mode session. PRO/DATATRIEVE does not write your input to the file.

When you include an OPEN command in a procedure, or when you open a log file and then execute a procedure, PRO/DATATRIEVE writes only the output of the procedure to the log file. It does not write any of the statements or commands in a procedure to the log file.

PRO/DATATRIEVE puts a RETURN between prompts and your input. To make the file an exact copy of the dialogue displayed on your video screen, you must edit the file.

Keeping log files of your dialogue with PRO/DATATRIEVE can provide you with a transaction log and can help you in developing and correcting PRO/DATATRIEVE definitions.

Example

Open a log file, display a procedure, execute the procedure, and close the log file. When you exit from PRO/DATATRIEVE, you can use File Services to display the log file or PROSE to edit it:

```
DTR> OPEN LOG
DTR> SHOW MULTIPLY
PROCEDURE MULTIPLY
DECLARE A PIC 99.
DECLARE B PIC 99.
DECLARE ANSWER EDIT_STRING ZZZZ COMPUTED BY A * B.
PRINT "This procedure asks for two two-digit numbers to multiply."
A = *."First number"
B = *."Second number"
PRINT SKIP, "The answer is "|ANSWER
END_PROCEDURE
DTR> :MULTIPLY
This procedure asks for two two-digit numbers to multiply.

Enter First number: 34
Enter Second number: 45

The answer is 1530

DTR> CLOSE
```

The log file created looks like this:

```
DTR>
SHOW MULTIPLY
PROCEDURE MULTIPLY
DECLARE A PIC 99.
DECLARE B PIC 99.
DECLARE ANSWER EDIT_STRING ZZZZ COMPUTED BY A * B.
PRINT "This procedure asks for two two-digit numbers to multiply."
A = *."First number"
B = *."Second number"
PRINT SKIP, "The answer is "!ANSWER
END_PROCEDURE
DTR>
:MULTIPLY
This procedure asks for two two-digit numbers to multiply.

Enter First number:
34
Enter Second number:
45

The answer is 1530

DTR>
CLOSE
```

7

Defining a Database

Chapter 7

Defining a Database

This chapter describes the data definition commands you can use to define a PRO/DATATRIEVE database:

Defining Data Domains
Defining Records
Defining Data Files
Defining a Dictionary Table
Defining View Domains

PRO/DATATRIEVE for Beginners showed you how to use ADT to define a domain, record, and file. ADT provides you with detailed questions and help in your definitions, but does not let you use some field definition clauses or create group fields. You can edit a record definition created with ADT to include field definition clauses that ADT cannot create for you or to create group fields.

In addition to domains, records, and data files, you can define view domains to look at fields and records in one or more domains in a different way. You can also define and use dictionary tables as part of a database. Dictionary tables store code and translation pairs that you can use to validate data and to save space in data files.

Among other things, examples in this chapter show how to:

- Define records with elementary, group, and list fields
- Define domains and data files on different volumes
- Store short customer codes in dictionary tables
- Combine information from one domain with information from another domain

DEFINING DATA DOMAINS

Use the DEFINE DOMAIN command to create and name a data domain. A data domain associates a record definition with a data file.

Format

```
DEFINE DOMAIN domain-name USING record-name ON file-spec;
```

Explanation

Domain-name names the domain. *Record-name* tells PRO/DATATRIEVE which record definition to associate with the domain and *file-spec* specifies which data file to use.

If you want the data file to reside in your current directory, you need only specify a file name for *file-spec*. PRO/DATATRIEVE creates the file in your current directory and gives it a "Data" file type. Like all Professional file names, a data file name can consist of from one to nine letters or numbers.

To create a data file in a different directory or on a different device, or to specify a different file type, use an extended file name, such as DZ1:[CUSTOMERS]CUST.IND. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

After you type a semicolon (;) to end the domain definition, PRO/DATATRIEVE stores the domain definition in your current dictionary. If you press DO before typing a semicolon to end the definition, PRO/DATATRIEVE prompts you to continue with the DFN> prompt.

If you make a mistake while typing the DEFINE DOMAIN command, PRO/DATATRIEVE displays an error message and returns you to the DTR> prompt without storing the domain definition in your dictionary.

Comments

A domain name cannot duplicate the name of any other object in your current dictionary. Like other PRO/DATATRIEVE names, it can consist of from 1 to 31 letters, numbers, hyphens, and underscores and should begin with a letter and end with a letter or number.

It is a good idea to store data files in the same directory as the dictionary that contains the domain definitions. To do this, specify the same device and directory for the file named in the DEFINE DOMAIN command as specified in the DEFINE DICTIONARY command. By doing this, you can access domains in different directories simply by setting your current dictionary to the one containing the domain definition you need.

You can define a domain before you define the record and data file named in the DEFINE DOMAIN command. However, you cannot ready the domain until you have defined the record and data file. The record definition tells PRO/DATATRIEVE how to store and interpret information and the file definition tells PRO/DATATRIEVE to create the data file.

To modify an existing domain definition, use the PRO/DATATRIEVE Editor or copy the definition to a command file with the EXTRACT command, exit from PRO/DATATRIEVE, and use PROSE to edit the command file. If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message. To execute a command file, type an at sign (@) and the file name.

See Chapter 16 for information on the EXTRACT command, Chapter 17 for information on the PRO/DATATRIEVE Editor, and Chapter 10 for information on command files.

Examples

Define a domain named STOCKS. Tell PRO/DATATRIEVE to use the STOCKS-REC record and the STOCK data file and that the data file should reside on volume PERSONAL in the STOCKS directory:

```
DTR> SHOW DICTIONARY
The current dictionary is DW1:[USERFILES]CUSTOMERS.DIC
DTR> SET DICTIONARY PERSONAL:[STOCKS]STOCKS.DIC
DTR> DEFINE DOMAIN STOCKS USING
DFN>       STOCKS-REC ON PERSONAL:[STOCKS]STOCK;
DTR>
```

Define a domain named PERSONNEL in your current dictionary stored in your current directory. Have the domain use the PERSONNEL-REC record definition and the PERSON.IND data file. Do not specify a volume or directory:

```
DTR> DEFINE DOMAIN PERSONNEL USING PERSONNEL-REC ON PERSON.IND
DFN> ;
DTR>
```

DEFINING RECORDS

Use the DEFINE RECORD command to store a record definition in your current dictionary. In the record definition, you name group and elementary fields and define the characteristics of each field.

Format

```
DEFINE RECORD record-name USING

level-number field-name [field-definition-clause(s)] .
                        .
                        .
                        .
;
```

Explanation

Record-name names the record you want to define.

The second and succeeding lines of the record definition define the record's fields:

- A field *level-number* defines the relationship between fields.
- A *field-name* names the field.
- *Field-definition-clauses* define the characteristics of the field and specify how PRO/DATATRIEVE should store and interpret information in the record.

You must type a period after the last field definition clause or after the field name if the field does not contain any definition clauses to end the field definition. If you omit the period, PRO/DATATRIEVE displays an error message when you try to define the next field or end the record definition. PRO/DATATRIEVE returns you to the DTR> prompt without storing the record definition in your dictionary.

PRO/DATATRIEVE prompts with the DFN> prompt until you type a semicolon on a separate line to end the record definition. This ends the record definition. PRO/DATATRIEVE then displays a message that tells you the size of the record. This message indicates that PRO/DATATRIEVE has stored the record definition in your current dictionary.

Comments

You can omit the USING keyword from the DEFINE RECORD command.

A record name cannot duplicate the name of any other object in your current dictionary and a field name should not duplicate the name of another field in the same record. Like other PRO/DATATRIEVE names, record and field names can consist of from 1 to 31 letters, numbers, hyphens, and underscores. These names should begin with a letter and end with a letter or number.

Every field in a record definition must have a level number that specifies its relationship to other fields in the record. PRO/DATATRIEVE recognizes field levels in a record according to the level number you assign to each field.

The highest possible field level number is 1 or 01, and the lowest possible level number is 65. Leading zeros, as in 01 or 05, do not affect the value of the level number. The highest level field in a record, the top-level field, contains all other fields in the record and is usually assigned a field number of 1 or 01. No other field in a record can have the same level number as the top-level field.

Field level numbers need not be consecutive. ADT, for example, assigns the top-level field of a record the number 01 and all other fields the number 15.

PRO/DATATRIEVE records can consist of group fields and elementary fields. A group field encompasses fields with higher level numbers, while an elementary field does not encompass any other fields. A group field allows you to refer to a group of fields as a unit. If there is only one field in a record, that field is the top-level field and an elementary field.

Fields with the same level number have the same relationship to other fields in the record and can be either elementary or group fields. Group fields can contain other group fields that have a higher level number. For example:

```
DTR> DEFINE RECORD CUST-REC USING
DFN> 01 CUST-REC.
DFN>     15 COMPANY-NAME PIC IS X(20)
DFN>        QUERY-NAME IS COMPANY.
DFN>     15 ADDRESS.
DFN>        20 STREET PIC IS X(15).
DFN>        20 CITY PIC IS X(10).
DFN>        20 S-Z.
DFN>           25 STATE PIC IS X(2).
DFN>           25 ZIP-CODE PIC IS X(5)
DFN>              QUERY-NAME IS ZIP.
DFN> ;
[Record CUST_REC is 52 bytes long]
DTR>
```

This record contains one top-level group field, CUST-REC, that contains all other fields in the record. The next lower level fields, COMPANY-NAME, an elementary field, and ADDRESS, a group field, have the same relationship to the top-level field because they have the same level number (15). Fields contained in the ADDRESS field have the same relationship to the ADDRESS field, even though the S-Z field is a group field.

Group fields like ADDRESS and S-Z can save you typing time. To refer to the full address, you can specify the ADDRESS field instead of all the elementary fields in the group (STREET, CITY, STATE, and ZIP-CODE). To refer to the state and ZIP code fields, you can specify the S-Z field. You can, of course, also refer to elementary fields within group fields by specifying the field name.

Field definition clauses describe a field's characteristics. QUERY-NAME IS COMPANY, for example, tells PRO/DATATRIEVE to recognize COMPANY as a short name for the field COMPANY-NAME. PIC IS X(20) tells PRO/DATATRIEVE that the COMPANY-NAME field consists of 20 characters.

With field definition clauses, you can tell PRO/DATATRIEVE:

- That values in the field have a specific data type, length, and format (PICTURE, USAGE, and COMPUTED BY)
- That you want field values displayed or printed in a specific way (EDIT-STRING)
- That only specific values can be stored in the field (VALID IF)
- That you want to use a substitute column header (QUERY-HEADER) or an abbreviated name for the field (QUERY-NAME)
- That a field has an alternate definition (REDEFINES)
- That the field is a list containing multiple occurrences of a field or group of fields (OCCURS)

Chapter 8 describes field definition clauses and shows how to use them in the DEFINE RECORD command.

An elementary field must contain at least one field definition clause. If a record contains only one field, that field is an elementary field and must contain a definition clause. A group field does not have to contain any field definition clauses.

Elementary fields contained in group fields determine a group field's characteristics. The STATE and ZIP-CODE fields in the previous example, for instance, determine the characteristics of the S-Z field, while the STREET, CITY, and S-Z fields determine the characteristics of the ADDRESS field.

You can put field definition clauses on the same line, or you can put each clause on a separate line by pressing the DO key. To separate field definition clauses on the same line, use spaces or press the TAB key.

If you take the time to think about how fields in a record relate to each other and format the record definition by indenting and placing field definition clauses on their own lines to reflect these relationships, your record definition will be easier to understand and read. The way you format a record definition, though, does not affect field characteristics or the relationship between fields. Field level numbers determine the relationship between fields.

To modify an existing record definition, use the PRO/DATATRIEVE Editor or copy the definition to a command file with the EXTRACT command, exit from PRO/DATATRIEVE, and use PROSE to edit the command file. If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message. To execute a command file, type an at sign (@) and the file name.

See Chapter 16 for information on the EXTRACT command, Chapter 17 for information on the PRO/DATATRIEVE Editor, and Chapter 10 for information on command files.

When you edit a record definition, you may not be able to use old data files. If the new record definition is not the same length, or if fields have different data types, you have to restructure the database as described in Chapter 5.

Examples

Define a record for the CUSTOMERS domain that contains a group ADDRESS field:

```
DTR> DEFINE RECORD CUSTOMERS-REC USING
DFN> 01 CUSTOMERS-REC.
DFN>     15 COMPANY-NAME PIC IS X(20)
DFN>        QUERY-NAME IS COMPANY.
DFN>     15 CONTACT-PERSON PIC IS X(15)
DFN>        QUERY-NAME IS CONTACT.
DFN>     15 ADDRESS.
DFN>        25 STREET PIC IS X(15).
DFN>        25 CITY PIC IS X(10).
DFN>        25 STATE PIC IS X(2).
DFN>        25 ZIP-CODE PIC IS X(5)
DFN>           QUERY-NAME IS ZIP.
DFN>     15 PHONE-NUMBER PIC IS 9(10)
DFN>        EDIT-STRING IS XXX-XXX-XXXX
DFN>        QUERY-NAME IS PHONE.
DFN>     15 SALES-THIS-YEAR PIC IS  S9(5)V99
DFN>        EDIT-STRING IS $$$,$$$.$$
DFN>        QUERY-NAME IS SALES.
DFN>     15 LAST-PURCHASE USAGE IS DATE
DFN>        EDIT-STRING IS DD-MMM-YY.
DFN>     15 LAST-CONTACT USAGE IS DATE
DFN>        EDIT-STRING IS  DD-MMM-YY
DFN>        QUERY-NAME IS LAST-C.
DFN> ;
[Record CUSTOMERS_REC is 100 bytes long]
DTR>
```

Define a record for family information that contains a variable-length list field for children's names and ages. To define a list field, use the OCCURS DEPENDING field definition clause to specify how many children the list can contain and how many times to prompt for children's names and ages:

```
DTR> DEFINE RECORD FAMILY USING
DFN> 01 FAMILY.
DFN>     03 PARENTS.
DFN>        06 FATHER PIC IS X(10).
DFN>        06 MOTHER PIC IS X(10).
DFN>     03 NUMBER-KIDS PIC 99
DFN>        EDIT-STRING IS Z9.
DFN>     03 KIDS OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER-KIDS.
DFN>        06 EACH-KID.
DFN>           09 KID-NAME PIC IS X(10)
DFN>              QUERY-NAME IS KID.
DFN>           09 AGE PIC IS 99
DFN>              EDIT-STRING IS Z9.
DFN> ;
[Record FAMILY is 142 bytes long]
DTR>
```

DEFINING DATA FILES

Use the DEFINE FILE command to define a data file for a domain.

Format

```
DEFINE FILE FOR domain-name [key-clause],... [MAX] ;

key-clause:   KEY = field-name [(NO CHANGE, NO DUP)]
```

Explanation

Domain-name names a previously defined domain. PRO/DATATRIEVE creates a file for the domain you specify, using the file name specified in the DEFINE DOMAIN command that created the domain.

Key-clause tells PRO/DATATRIEVE to create an indexed data file and to use the specified *field-name* as an index key. The first key you specify is the primary key. Any other keys you specify are alternate index keys.

NO CHANGE tells PRO/DATATRIEVE not to allow changes to values stored in the key field. *CHANGE* tells PRO/DATATRIEVE to allow changes to values stored in the key field.

- PRO/DATATRIEVE always creates primary key fields with the NO CHANGE attribute. Because PRO/DATATRIEVE never allows changes to primary key fields, you cannot specify CHANGE and you do not need to specify NO CHANGE for a primary key field.
- PRO/DATATRIEVE always creates alternate key fields with the CHANGE attribute. If you do not want to allow changes for an alternate key field, you must specify NO CHANGE.

NO DUP tells PRO/DATATRIEVE not to allow duplicate values in a key field. *DUP* tells PRO/DATATRIEVE to allow duplicate values in the key field:

- PRO/DATATRIEVE always creates primary key fields with the NO DUP attribute. If you want to allow duplicates in a primary field, you must specify DUP.

(continued on next page)

- PRO/DATATRIEVE always creates alternate keys with the DUP attribute. If you do not want to allow duplicates in an alternate key field, you must specify NO DUP.

If you try to store a record with a duplicate key value in a key field that does not allow duplicates, PRO/DATATRIEVE displays an error message and does not store the record. If you try to change a key field that does not allow changes, PRO/DATATRIEVE displays a message and does not modify the field.

If you omit the *key-clause*, PRO/DATATRIEVE creates a sequential file. If the record for a sequential file contains a variable-length list field as defined with an OCCURS DEPENDING clause, you must include the word MAX. MAX tells PRO/DATATRIEVE that all records in the sequential file should be large enough to hold the maximum number of items in the list field. If you do not specify MAX when creating a sequential file to hold records with a variable-length list field, you cannot add more values to the list field than you store in the original record.

Comments

You can omit the FOR keyword and the ending semicolon from the DEFINE FILE command to save typing time.

Indexed and sequential files differ in two ways:

- You can delete records in an indexed file with the ERASE statement, and you can modify all fields except the primary key field with the MODIFY statement. To change a primary key field, erase the erroneous record and store a new record. Deleted records reduce the amount of storage your data file uses.
- You cannot delete records in a sequential file. You can, however, use the MODIFY statement to delete or change all fields in the record. Empty records in a sequential file, though, continue to occupy storage space in your data file.

You can define a file for a domain only after you define the domain and the record definition used by the domain. The DEFINE DOMAIN command tells PRO/DATATRIEVE what to name the file and where it should reside. The DEFINE RECORD command tells PRO/DATATRIEVE how big records in the file have to be.

If you change the size of a record, you need to define a new file for the domain that uses the record. If you do not, PRO/DATATRIEVE displays a message that tells you the file has a bad record size.

PRO/DATATRIEVE does not allow DATE fields to serve as index keys. If you identify a DATE field as an index key, PRO/DATATRIEVE creates a file, but notifies you that it will not support indexed retrieval of records for that field.

Examples

Define an indexed file for a domain named PHONES. Use the LAST-NAME field as the primary index field. Because last names may be identical, specify (DUP) to allow duplicate values:

```
DTR> DEFINE FILE FOR PHONES KEY = LAST-NAME (DUP);
```

Define an indexed file for a domain named PERSONNEL. Use the ID field as the primary index field. PRO/DATATRIEVE automatically creates primary keys with the (NO DUP) attribute:

```
DTR> DEFINE FILE FOR PERSONNEL KEY = ID;
```

Define a sequential file for the TEMP domain:

```
DTR> DEFINE FILE FOR TEMP;
```

DEFINING A DICTIONARY TABLE

Use the DEFINE TABLE command to name and define a dictionary table. A table associates code and translation pairs and can save space in records. You can also use tables to validate values stored in records and to display translations for values stored in records.

Format

```
DEFINE TABLE table-name

    "code" : "translation",
        .          .
        .          .
        .          .
    [ELSE "translation"]

END-TABLE
```

Explanation

Table-name names the table you want to define.

Code and *translation* specify the two values you want to associate. A state table, for example, might associate two-letter codes like "NH" and "MA" with full state names like "NEW HAMPSHIRE" and "MASSACHUSETTS". You must separate code and translation pairs with a colon (:).

If you do not include the optional ELSE line, end each code and translation pair except the last with a comma. If you type a comma after a translation, press DO, and type END-TABLE, PRO/DATATRIEVE displays an error message and does not store the table definition in your dictionary.

The optional ELSE clause allows you to tell PRO/DATATRIEVE what translation to use if you refer to a code that does not exist in the table. If you use this clause, you must type a comma after the last code and translation pair and cannot type a comma after the ELSE clause translation.

PRO/DATATRIEVE prompts with the DFN> prompt until you type the END-TABLE command to end the table definition. If you make a mistake while defining a table, PRO/DATATRIEVE displays an error message and returns you to the DTR> prompt without storing the table definition in your dictionary.

Comments

A table name cannot duplicate the name of any other object in your current dictionary. Like other PRO/DATATRIEVE names, it can consist of from 1 to 31 letters, numbers, hyphens, and underscores and should begin with a letter and end with a letter or number.

You do not have to put quotation marks around all codes and translations, but it is a good idea to get in the habit of doing so. If you do not use quotation marks, PRO/DATATRIEVE converts lowercase letters to uppercase and displays an error message if a code or translation contains a space.

You can include as many code and translation pairs in a table definition as you like.

You can use tables to validate a value before storing it in a field by referring to the table in a VALID IF record definition clause:

```
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>     03 NAME PIC X(20).
DFN>     03 NUMBER PIC 9(7)
DFN>        EDIT-STRING IS XXX-XXXX.
DFN>     03 AREA-CODE PIC XXX VALID IF AREA-CODE IN AREA-TABLE.
DFN> ;
[Record PHONE is 30 bytes long]
DTR>
```

When you store values in the PHONE record, PRO/DATATRIEVE refers to AREA-TABLE to make sure the area code you type is in the table. If it is not, PRO/DATATRIEVE tells you the area code you typed is invalid and prompts again for an area code.

You can also display code translations by using the keyword VIA in a PRO/DATATRIEVE statement to refer to a table. The AREA-CODE table associates area codes with states:

```
DTR> DEFINE TABLE AREA-TABLE
DFN> TABLE AREA-TABLE
DFN>     "603" : "NEW HAMPSHIRE",
DFN>     "617" : "MASSACHUSETTS",
DFN      "802" : "VERMONT"
DFN> END-TABLE
DTR>
```

To display the state associated with an area code stored in a record, use a PRINT statement like this:

```
DTR> PRINT NAME, AREA-CODE, AREA-CODE VIA AREA-TABLE OF PHONES

                      AREA     AREA
        NAME          CODE     CODE

DONNA                 802   VERMONT
JOHN                  617   MASSACHUSE
MARILYN               603   NEW HAMPSH

DTR>
```

This example does not include a USING clause to tell PRO/DATATRIEVE how to display the table value, so PRO/DATATRIEVE displays only the first ten characters of the translation string. To display all the characters in a translation string, specify an edit string in a USING clause. To give the translation field a unique column header, specify a column header for the field:

```
DTR> PRINT NAME, AREA-CODE, AREA-CODE VIA
[Looking for table name]
CON>     AREA-TABLE ("STATE") USING X(15) OF PHONES

                      AREA
        NAME          CODE        STATE

DONNA                 802   VERMONT
JOHN                  617   MASSACHUSETTS
MARILYN               603   NEW HAMPSHIRE
DTR>
```

To modify an existing table definition, use the PRO/DATATRIEVE Editor or copy the definition to a command file with the EXTRACT command, exit from PRO/DATATRIEVE, and use PROSE to edit the command file. If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message. To execute a command file, type an at sign (@) and the file name.

See Chapter 16 for information on the EXTRACT command, Chapter 17 for information on the PRO/DATATRIEVE Editor, and Chapter 10 for information on command files.

Example

Define a table that associates short codes with company names:

```
DTR> DEFINE TABLE COMPANIES
DFN>     "BS"     : "BASIC SYSTEMS",
DFN>     "IA"     : "INFO ANALYSIS",
DFN>     "KS"     : "KEY SPECIALISTS",
DFN>     "LSC"    : "LOGIC SYSTEMS CO.",
DFN>     "PPI"    : "PAPER & PEN, INC.",
DFN>     "PU"     : "PRODUCTS UNLIMITED",
DFN>     "PP"     : "PROSE PROS",
DFN>     "SA"     : "STONE ASSOCIATES",
DFN>     "TA"     : "TACTICAL AID, INC.",
DFN>     "WRA"    : "WRITE RIGHT ASSOC.",
DFN>     ELSE "NOT A VALID CUSTOMER CODE"
DFN> END-TABLE
DTR>
```

DEFINING VIEW DOMAINS

Use the DEFINE DOMAIN command to name and define a view of an existing domain or domains. A view domain allows you to see selected records and fields from one or more domains. A view domain definition includes the record definition for the view in the view domain definition.

Format

```
DEFINE DOMAIN view-name OF domain-name,... USING

    view-field-number view-field-name OCCURS FOR rse .        (1)

    domain-field-number domain-field-name FROM domain-name .  (2)

                        .
                        .
                        .
;
```

Explanation

View-name names the view domain. The *OF* clause tells PRO/DATATRIEVE which domain contains the records you want to include in the view. To view records from more than one domain, list the domain names, separated by commas.

(1) *View-field-number* and *view-field-name* specify a level number and name for a field in the view. The *OCCURS FOR rse* clause specifies the domain that contains the records you want in the view and establishes conditions for selecting records from that domain. The RSE must contain a domain name that matches a domain named in the *OF* clause and must end with a period.

(2) *Domain-field-number* and *domain-field-name* tell PRO/DATATRIEVE what field from a record to include in the view field and the *FROM* clause tells PRO/DATATRIEVE which domain to use. *Domain-name* must match the domain name specified in the RSE in the preceding OCCURS clause.

To include an entire record in a view field, specify the top-level field as the *domain-field-name*. To include only selected fields from the record, repeat line (2) for each field you want in the view. Be sure to end each line with a period.

To include fields from more than one domain in the view, repeat line (1) to define a new view field and a new source of records. Then use line (2) to tell PRO/DATATRIEVE which fields from the new record source to include in the view field. Remember that the domain specified in the RSE in line (1) must match a domain named in the OF clause and that the domain named in line (2) must match the domain name specified in the RSE.

After you type a semicolon (;) to end the definition, PRO/DATATRIEVE stores the view domain definition in your current dictionary. If you press DO before typing a semicolon to end the definition, PRO/DATATRIEVE prompts you to continue with the DFN> prompt. If you make a mistake while typing the DEFINE DOMAIN command, PRO/DATATRIEVE displays an error message and returns you to the DTR> prompt without storing the domain definition in your dictionary.

Comments

You can omit the USING keyword from the DEFINE DOMAIN command to save typing time.

The field numbers you assign when you define a view domain are like the field level numbers you assign to fields in the DEFINE RECORD command. *View-field-number* identifies the top-level field in the view, so it must be the lowest number, like 01. *Domain-field-numbers* identify the elementary and group fields in the view. See the section on defining records for more information on assigning field numbers.

A view domain name cannot duplicate any other object in your current dictionary. Like other PRO/DATATRIEVE names, it can consist of from 1 to 31 letters, numbers, hyphens, and underscores and should begin with a letter and end with a letter or number.

To modify an existing domain definition, use the PRO/DATATRIEVE Editor or copy the definition to a command file with the EXTRACT command, exit from PRO/DATATRIEVE, and use PROSE to edit the command file. If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message. To execute a command file, type an at sign (@) and the file name.

See Chapter 16 for information on the EXTRACT command, Chapter 17 for information on the PRO/DATATRIEVE Editor, and Chapter 10 for information on command files.

You must have the same access privilege (READ, WRITE, MODIFY) to a view domain and to the domains it uses or PRO/DATATRIEVE displays an error message. If you have a domain used by the view domain readied for WRITE access, for example, you cannot ready the view domain for READ access. It is a good idea to finish all ready domains before readying a view domain.

You cannot store or erase records in a view domain, but you can modify fields in the view domain. To modify fields in a view domain, specify WRITE or MODIFY access in the READY command. PRO/DATATRIEVE changes fields in the data domain records to reflect changes made to view domain fields.

Remember that you cannot modify an index field. If you try to modify a field that is an index field for a data domain record, PRO/DATATRIEVE displays an error message and does not change the field.

Examples

Define a view of the CUSTOMERS domain that allows you to see only the COMPANY, LAST-CONTACT, and SALES-THIS-YEAR fields for companies with sales not equal to zero. Then ready the view domain for WRITE or MODIFY access and modify the sales field for the first record in the domain:

```
DTR> DEFINE DOMAIN SALES-VIEW OF CUSTOMERS USING
DFN> 01 CUSTOMER OCCURS FOR CUSTOMERS WITH SALES NE 0.
DFN>     03 COMPANY FROM CUSTOMERS.
DFN>     03 LAST-CONTACT FROM CUSTOMERS.
DFN>     03 SALES FROM CUSTOMERS.
DFN> ;
DTR> READY SALES-VIEW WRITE
DTR> PRINT SALES-VIEW
```

```
                                        SALES
      COMPANY              LAST         THIS
        NAME              CONTACT       YEAR

INFO ANALYSIS            2-Mar-83      $309.00
KEY SPECIALISTS         11-Dec-82    $1,025.00
PAPER & PEN, INC.        9-Jan-83    $2,031.00
PROSE PROS              15-Feb-83    $1,053.00
STONE ASSOCIATES        22-Feb-83    $2,091.00

DTR> FIND FIRST 1 SALES-VIEW
[1 Record found]
DTR> SELECT
DTR> PRINT

                                        SALES
      COMPANY              LAST         THIS
        NAME              CONTACT       YEAR

INFO ANALYSIS            2-Mar-83      $309.00

DTR> MODIFY SALES
Enter SALES_THIS_YEAR: 310
DTR> PRINT

                                        SALES
      COMPANY              LAST         THIS
        NAME              CONTACT       YEAR

INFO ANALYSIS            2-Mar-83      $310.00

DTR> FINISH
DTR> READY CUSTOMERS
DTR> PRINT COMPANY, SALES OF FIRST 1 CUSTOMERS WITH SALES NE 0

                         SALES
      COMPANY             THIS
        NAME              YEAR

INFO ANALYSIS             $310.00

DTR>
```

Define a view domain that combines fields from the INVENTORY and TRANS domains to form an itemized and up-to-date inventory history. Include only posted sales transactions in the view domain by specifying FLAG EQ 1 AND TYPE EQ "S" in the RSE that selects fields from the TRANS domain. To match item numbers in the TRANS and INVENTORY domains, use the top-level field of the INVENTORY domain record (INVENTORY-REC) as a context variable

in the RSE. Use a date variable in the RSE that selects records from the TRANS domain to specify the beginning date for records to include in the view. Then define a procedure that declares the date variable, prompts for a value for it, readies the view, and displays the view:

```
DTR> SET NO PROMPT
DTR> DEFINE DOMAIN INV-VIEW OF INVENTORY, TRANS USING
DFN> 01 INV-DATA OCCURS FOR INVENTORY.
DFN>    03 ITEM-NAME FROM INVENTORY.
DFN>    03 ON-HAND FROM INVENTORY.
DFN>    03 SALES-DATA OCCURS FOR TRANS WITH
DFN>       (FLAG EQ 1) AND (TYPE EQ "S") AND
DFN>       (ITEM-NUMBER EQ INVENTORY-REC.ITEM-NUMBER) AND
DFN>       (TRANS-DATE GE VIEW-DATE) SORTED BY TRANS-DATE.
DFN>       05 TRANS-DATE FROM TRANS.
DFN>       05 QUANTITY FROM TRANS.
DFN> ;
DTR> DEFINE PROCEDURE VIEW-OF-INV
DFN> DECLARE VIEW-DATE USAGE IS DATE.
DFN> VIEW-DATE = *."What date for records in the view"
DFN> FINISH
DFN> READY INV-VIEW
DFN> PRINT INV-VIEW
DFN> FINISH
DFN> END-PROCEDURE
DTR> :VIEW-OF-INV
Enter What date for records in the view: 7-1-83

   ITEM       ON     TRANS
   NAME      HAND     DATE    QUANTITY

BAFFLES       2353  7/07/83      121
BALANCES       900
DO DADS        200  7/07/83     3000
                    7/11/83     1000
FIXITS        3912  7/18/83       34
GADGETS       1078
GLITCHES       284
HITCHES       1539
TANDEMS        800
THINGS        1000
WIDGETS       2688  7/01/83       50

DTR>
```

8

Using Field Definition Clauses

Chapter 8

Using Field Definition Clauses

This chapter describes the field definition clauses you can use to define the characteristics of record fields and variables:

Defining Data Type and Size
Defining Storage for Numeric and Date Fields
Defining Field Display Characteristics
Defining Alternate Field Names
Defining Alternate Column Headers
Defining Alternate Field Definitions
Defining Validation Criteria
Defining COMPUTED BY Fields
Defining a Fixed-length List Field
Defining a Variable-length List Field

Elementary fields must contain a PICTURE, USAGE, or COMPUTED BY clause and can contain other field definition clauses. Group fields can contain only the OCCURS, REDEFINES, and QUERY-NAME clauses and may contain none of these clauses.

A variable defined with the DECLARE statement must include a PICTURE, USAGE, or COMPUTED BY clause to specify the data type, length, and format of the variable. You can use all other field definition clauses except OCCURS or REDEFINES to describe variable characteristics. For information on declaring and using variables, see Chapter 15.

Among other things, examples in this chapter show how to:

- Create edit strings for money, name, and date fields
- Suppress leading zeros in numeric fields
- Create long text fields
- Validate values using tables
- Create a table translation edit string
- Compute field values based on other field values
- Define a list field that contains a list field

Table 8-1 summarizes field definition clauses. Appendix A lists complete record definitions for all records used in examples in this chapter.

Table 8-1: Summary of Field Definition Clauses

Clause	Purpose
COMPUTED BY	Tells PRO/DATATRIEVE how to calculate values for the field
EDIT-STRING	Tells PRO/DATATRIEVE how to display or print values stored in the field
OCCURS	Defines a repeating field and identifies it as a fixed-length list
OCCURS DEPENDING	Defines a repeating field and identifies it as a variable-length list
PICTURE	Specifies the data type and size values stored in the field
QUERY-HEADER	Specifies an alternate column header for the field
QUERY-NAME	Specifies an alternate name for the field
REDEFINES	Specifies an alternate definition for another field in the record
USAGE	Specifies the length and data type of a numeric field or specifies a date field
VALID IF	Specifies validation criteria for PRO/DATATRIEVE to use when storing a value in the field

DEFINING DATA TYPE AND SIZE

Use the PICTURE (or PIC) clause to specify the data type and length of an elementary field or variable.

Format

```
PICTURE IS picture-string
```

Explanation

A *picture-string* consists of one or more special characters that tell PRO/DATATRIEVE what type of data the field can contain and how to store data in the field. An X in the picture string indicates character data (numbers, letters, and other characters), while a 9 indicates numeric data.

The number of Xs or 9s in the picture string tells PRO/DATATRIEVE how long the field should be. The string 99999, for example, tells PRO/DATATRIEVE the field can contain five digits. The string XX specifies a two-character field.

Table 8-2 lists and describes picture string characters.

Table 8-2: Picture String Characters

Picture Character	Data Type	Meaning
X	Character	Each X reserves one place for a letter, digit, other printing character, or space.
9	Numeric	Each 9 reserves one place for a digit. You can specify from 1 to 18 digits for a numeric field.
S	Numeric	An S indicates that the field contains a sign for negative or positive values. If you do not include an S, you cannot store negative values in a numeric field. A picture string can contain only one S and it must be the leftmost character.
V	Numeric	A V indicates a decimal point. The decimal point does not occupy a character position in the field, although PRO/DATATRIEVE uses its location when storing data in the field. A picture string can contain only one V.

You can save typing time by placing the number of characters or digits you want in the picture string in parentheses, as in 9(7) or X(15). The edit string X(15) means the same thing to PRO/DATATRIEVE as XXXXXXXXXXXXXXX.

When you define a numeric field, you can specify a sign with the S character and a decimal point with the V character. A picture string can contain only one S and one V character.

Comments

You can use the complete form of the PICTURE clause, or, to save typing time, you can just type PIC. You can also omit the word IS from the PIC clause.

You can store a negative number in a field by using the S character in the PICTURE clause. However, PRO/DATATRIEVE does not display signs unless the field definition has an EDIT-STRING clause that includes a sign character:

```
03 CURRENT-BALANCE PIC IS S9(5)V99
   EDIT-STRING IS $$$,$$$.$$-.
```

To indicate that a numeric field can contain a decimal value, include a V in the picture string. A V specifies the position of a decimal point, but the decimal point does not occupy a position in the record. PRO/DATATRIEVE uses the "implied" decimal point in computations and Boolean expressions. To display the decimal point, however, you must include an edit string that contains a decimal point.

Example

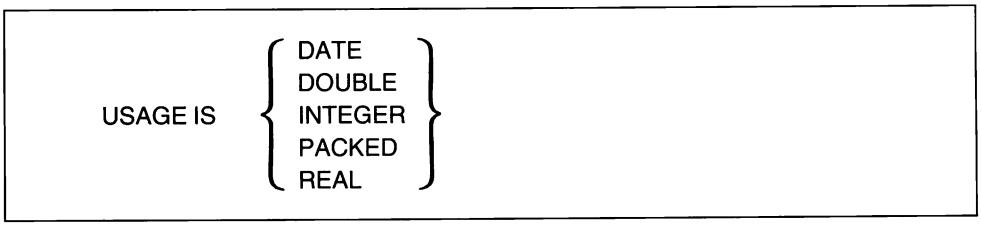
Define a record with a 10-character alphanumeric field, a 5-digit decimal value, a 6-digit signed, nondecimal value, and a 4-digit signed, decimal value.

```
DTR> DEFINE RECORD PIC-TEST USING
DFN> 01 TEST.
DFN>    03 CHAR-FIELD PIC X(10).
DFN>    03 DEC-FIELD PIC 999V99.
DFN>    03 SIGNED-FIELD PIC S9(6).
DFN>    03 SIGNED-DEC-FIELD PIC S99V99.
DFN> ;
[Record PIC_TEST is 25 bytes long]
DTR>
```

DEFINING STORAGE FOR NUMERIC AND DATE FIELDS

Use the USAGE clause to specify the internal format of a numeric or date field or variable. You must include a USAGE clause for date fields and variables.

Format



Explanation

The keyword you use in the USAGE clause tells PRO/DATATRIEVE how to store field values internally:

- *DATE* specifies a date field. PRO/DATATRIEVE stores dates as eight-byte binary values, using a base date of 00:00:00 AM on November 17, 1858. You cannot specify a date value prior to the base date. PRO/DATATRIEVE translates dates to binary values when you store them, and translates binary values to the format specified in the EDIT-STRING clause. If you do not specify an EDIT-STRING clause, PRO/DATATRIEVE displays a date value in the format DD-MMM-YYY.
- *DOUBLE* specifies a double-precision floating-point (real) number. PRO/DATATRIEVE stores double-precision numbers as eight-byte units.
- *INTEGER* specifies an integer stored in binary format. The PIC clause for the field determines the size of the field.
- *PACKED* specifies a number stored in packed-decimal format, two digits per byte. Packed decimal values must contain a sign.
- *REAL* specifies a single-precision floating-point (real) number, stored by PRO/DATATRIEVE as a four-byte unit.

Comments

You can omit the keyword IS from the USAGE clause to save typing time.

A group field cannot contain a USAGE clause. You can use the USAGE clause only in elementary field definitions and variable declarations.

If you do not include a USAGE clause in a numeric field definition, each digit in the field occupies one byte of storage. The PIC clause determines the length of the field.

Use the PRO/DATATRIEVE date value expression, "TODAY", to assign the current date to a DATE field or variable. You must include quotation marks around the word TODAY when you assign it to a variable. If you use TODAY to store the current date in a record field, you do not need the quotation marks:

```
DTR> DECLARE X USAGE IS DATE.
DTR> X = "TODAY"
DTR> PRINT X

     X

18-May-83

DTR> X = *."what date"
Enter what date: TODAY
DTR> PRINT X

     X

18-May-83

DTR>
```

Examples

Define a SALES record that contains a DATE field for the sale date, an INTEGER field for units sold, and a DOUBLE field for the price per unit:

```
DTR> DEFINE RECORD USAGE-REC USING
DFN> 01 TEST-REC.
DFN>     03 BUYER PIC X(16).
DFN>     03 SALES-DATE USAGE IS DATE
DFN>        EDIT-STRING IS DDBM(9)BY(4).
DFN>     03 UNITS-SOLD PIC 9(4)
DFN>        EDIT-STRING IS Z(4)
DFN>        USAGE IS INTEGER.
DFN>     03 PRICE-PER-UNIT PIC S9(4)V99
DFN>        USAGE IS REAL
DFN>        EDIT-STRING IS $$,$$$.$$.
DFN> ;
[Record USAGE_REC is 30 bytes long]
DTR>
```

Define the same record without the USAGE clauses for the numeric fields. Because PRO/DATATRIEVE stores these numeric fields differently, this record is not the same length as the previous record:

```
DTR> DEFINE RECORD USAGE1-REC USING
DFN> 01 TEST-REC.
DFN>     03 BUYER PIC X(16).
DFN>     03 SALES-DATE USAGE IS DATE
DFN>        EDIT-STRING IS DDBM(9)BY(4).
DFN>     03 UNITS-SOLD PIC 9(4)
DFN>        EDIT-STRING IS Z(4).
DFN>     03 PRICE-PER-UNIT PIC S9(4)V99
DFN>        EDIT-STRING IS $$$,$$$.$$.
DFN> ;
DFN> [Record USAGE1_REC is 34 bytes long]
DTR>
```

In both records, PRO/DATATRIEVE allocates 16 bytes for the BUYER field and 8 bytes for the SALES-DATE field. In the first record, the USAGE clauses cause PRO/DATATRIEVE to use 2 bytes for the UNITS-SOLD field and 4 bytes for the PRICE-PER-UNIT field, for a total of 30 bytes. In the record without the USAGE clauses, PRO/DATATRIEVE allocates one byte of storage for each of the digits in the UNITS-SOLD field and PRICE-PER-UNIT field, for a total of 34 bytes.

DEFINING FIELD DISPLAY CHARACTERISTICS

Use the EDIT-STRING clause to indicate the way you want PRO/DATATRIEVE to display or print elementary field or variable values.

Format

```
EDIT-STRING IS edit-string
```

Explanation

An *edit-string* consists of one or more special characters that tell PRO/DATATRIEVE how to display values. If you do not use an EDIT-STRING clause to describe a field or variable, PRO/DATATRIEVE displays and prints values according to the PICTURE or USAGE format specified.

PRO/DATATRIEVE provides you with two basic types of edit string characters:

- *Replacement characters*, such as 9, Z, X, and M, reserve places and indicate how you want the value displayed.
- *Insertion characters*, such as / and -, indicate what special characters you want inserted in the displayed or printed value and where to insert them. Some insertion characters, like the dollar sign, also reserve places for characters in the value.

You can save typing time by placing the number of characters or digits you want in the edit string in parentheses, as in 9(7) or X(15). The edit string X(15) means the same thing to PRO/DATATRIEVE as XXXXXXXXXXXXXXX.

The characters you can use in an edit string depend on the data type of the field or variable. PRO/DATATRIEVE recognizes three types of data:

- *Alphanumeric values* generally consist of printing characters and spaces, as in "LOGIC SYSTEMS, INC." and "3V2".
- *Numeric values* consist of numbers, as in "45678", and may contain commas, a decimal point, and a dollar sign, as in "$45,678.99". Numeric values cannot include spaces.
- *Date values* consist of numbers or a combination of numbers and letters, separated by slashes, hyphens, spaces, commas, or periods, as in "1-5-83", "JUNE 8, 1979", and "5/JUNE/84".

Table 8-3 lists and describes replacement characters. Table 8-4 lists and describes insertion characters.

Table 8-3: Edit String Replacement Characters

Character	Data Type	Effect
X	Alphanumeric	Each X reserves one place for a character or space.
T	Alphanumeric	Each T reserves one place for a character or space. The number of Ts indicates the maximum segment for PRO/DATATRIEVE to print on a line. PRO/DATATRIEVE prints the field using lines of the specified length, without breaking words. If a word will not fit on the line, PRO/DATATRIEVE skips to the next line. If a word exceeds the number of reserved spaces, PRO/DATATRIEVE prints the word on one line.
9	Numeric	Each 9 reserves a place for one digit in the value and tells PRO/DATATRIEVE to display leading zeros as zeros. An edit string of 99999 tells PRO/DATATRIEVE to display the number 111 as 00111.
Z	Numeric	Each Z reserves one place for one digit in the value and tells PRO/DATATRIEVE to display spaces for leading zeros. An edit string of ZZZZZ tells PRO/DATATRIEVE to display the number 111 as ##111, where the pound signs indicate spaces.
DD	Date	Two Ds reserve two places for a number that corresponds to a day of the month. PRO/DATATRIEVE displays days between one and nine with a leading zero, as in "01-Jun-79".
MMM M(9)	Date	Each M reserves one place for a letter of the month name. Use three Ms to display the first three characters of a month name and nine to display the entire month name.
NN	Date	Two Ns reserve two places for a number that corresponds to a month. PRO/DATATRIEVE does not display leading zeros for the months between one and nine.
YY YYYY	Date	Each Y reserves one place for a year digit. Use two Ys to display the last two digits of a year (79) and four Ys to display the entire year (1979).
WWW W(9)	Date	Each W reserves one place for a letter of the day of the week for the date. Use three Ws to display the first three characters of a day and nine Ws to display the entire day.

Table 8-4: Edit String Insertion Characters

Character	Data Type	Effect
+ (plus) - (hyphen) . (period) , (comma)	Alphanumeric	PRO/DATATRIEVE inserts each plus, hyphen, period, or comma in the specified position.
B (space) 0 (zero) % (percent) / (slash)	Alphanumeric and Numeric	PRO/DATATRIEVE inserts each space, zero, percent sign, or slash in the specified position.
+ (plus) - (minus)	Numeric	If you specify only one plus sign at the beginning or end of an edit string, PRO/DATATRIEVE inserts a plus or minus sign, depending on the value, in the specified position. If you specify only one minus sign at the beginning or end of an edit string, PRO/DATATRIEVE inserts a minus sign for negative values in the specified position and no sign for positive values. If you specify more than one plus or minus sign at the beginning of the edit string, PRO/DATATRIEVE suppresses leading zeros before printing the appropriate sign and value.
$	Numeric	PRO/DATATRIEVE inserts the dollar sign in the specified position, if you specify only one dollar sign in an edit string. If you specify more than one dollar sign at the beginning of an edit string, PRO/DATATRIEVE suppresses leading zeros and prints a dollar sign before printing the value. Check dollar edit strings carefully. The first dollar sign reserves a place for a dollar sign, not for a digit. You usually need one more dollar sign in an edit string than there are digits in the value.
. (decimal)	Numeric	PRO/DATATRIEVE inserts a decimal point in the specified position. Do not specify more than one decimal point in a numeric edit string.
, (comma)	Numeric	PRO/DATATRIEVE inserts a comma in the specified position. If all digits that come before the comma are zeros, PRO/DATATRIEVE does not print the comma.
/ (slash) - (hyphen) . (period)	Date	PRO/DATATRIEVE inserts each slash, hyphen, or period in the specified position.

Comments

You can omit the word IS from the EDIT-STRING clause to save typing time.

If you do not specify an EDIT-STRING clause, PRO/DATATRIEVE displays alphanumeric values exactly as you stored them, numeric values with leading zeros to fill the field, and date values in the form DD-MMM-YY, as in 08-Jun-79. By using replacement and insertion characters, you can make values more meaningful and easier to read.

When you use an X edit string, PRO/DATATRIEVE prints characters from a field in left-to-right order. PRO/DATATRIEVE prints as many characters as there are Xs in the edit string, then stops. Specify as many Xs as you expect the longest value a field can contain. If the value has fewer characters than the number of Xs specified in the edit string, PRO/DATATRIEVE prints the entire value and then prints spaces in the remaining X positions.

The number of Ts in an edit string specify how many characters to print on one line. The edit string T(20), for example, indicates that a line should contain no more than 20 characters. If a value contains more characters than specified in a T edit string, PRO/DATATRIEVE prints as many full words on the line as possible. A word, in this instance, is a string of characters that ends with a space. PRO/DATATRIEVE does not divide words. If a single word is longer than the specified line length, PRO/DATATRIEVE prints the entire word on a line.

You cannot use the blank (B), hyphen (-), or slash (/) insertion characters in a T edit string.

Use edit strings whenever possible to make numeric fields more meaningful. You can, for example, store a negative number in a field by using the S character in the PICTURE clause. However, you may not remember that a number is negative and PRO/DATATRIEVE does not display a sign unless you specify an S in the edit string for the field.

You can use edit strings in PRINT statements and in Report Writer statements as well as in EDIT-STRING clauses in record definitions. This can be helpful when you are experimenting with edit strings, when you want to change the way PRO/DATATRIEVE displays or prints values, or when you want to override an

edit string specified in a record definition. The following example uses the plus (+) and minus (–) insertion characters to display leading and trailing signs for fields with no EDIT-STRING clause:

```
DTR> SHOW NEG-REC
RECORD NEG_REC
 USING
01 TEST.
   03 POS_NUM PIC IS S9999.
   03 NEG_NUM PIC IS S9999.
;
DTR> PRINT NEG

POS  NEG
NUM  NUM

1234 1234

DTR> PRINT POS-NUM USING -9999, NEG-NUM USING -9999 OF NEG

 POS   NEG
 NUM   NUM

 1234 -1234

DTR> PRINT POS-NUM USING 9999-, NEG-NUM USING 9999- OF NEG

 POS   NEG
 NUM   NUM

1234  1234-

DTR> PRINT POS-NUM USING +9999, NEG-NUM USING +9999 OF NEG

 POS   NEG
 NUM   NUM

+1234 -1234

DTR> PRINT POS-NUM USING 9999+, NEG-NUM USING 9999+ OF NEG

 POS   NEG
 NUM   NUM

1234+ 1234-

DTR>
```

After you determine the way you want signs displayed for a field, it is a good idea to include the edit string in an EDIT-STRING clause in the field definition.

When creating edit strings for numeric fields, be careful to specify enough characters for the field. Remember that a dollar sign edit string like $$$,$$$ reserves one place for a dollar sign, and only five places for digits.

If you do not specify enough edit string characters, PRO/DATATRIEVE prints asterisks (*) instead of a field value. The following field definitions contain edit strings that do not correspond in length to the PICTURE strings:

```
03 MODEL PIC IS 9(4)
   EDIT-STRING IS 99.
03 PRICE PIC IS 9(6)
   EDIT-STRING IS $$$,$$$.
```

If the MODEL field contains a four-digit number, like 1234, PRO/DATATRIEVE prints two asterisks (**) instead of the number. If the PRICE field contains a six-digit number, PRO/DATATRIEVE can print only five digits. The first dollar sign in the edit string reserves a place for itself, the comma reserves a place for itself, and the remaining dollar signs reserve places for only five digits. Therefore, PRO/DATATRIEVE displays $**,*** instead of the value stored in the field.

Be careful where you put an EDIT-STRING clause that contains a decimal point or period as the last character. PRO/DATATRIEVE treats the trailing decimal point or period as the period that ends the field definition. You can avoid this problem by putting another field definition clause on the same line as the EDIT-STRING clause.

Because multiple plus and minus signs in edit strings suppress leading zeros, they must be the first characters in an edit string:

```
DTR> DECLARE NEG PIC S9(5) EDIT-STRING ----
DTR> DECLARE POS PIC S9(5) EDIT-STRING +++++.
DTR> NEG = -99
DTR> POS = -88
DTR> PRINT NEG, POS

 NEG    POS

  -99    -88

DTR> NEG = 99
DTR> POS = 88
DTR> PRINT NEG, POS

 NEG    POS

   99    +88

DTR>
```

You can precede multiple dollar signs with a single plus or minus sign, as in this field definition:

```
03 BALANCE-ITEM PIC S9(6)V99
   QUERY-NAME IS BALANCE
   QUERY-HEADER IS "BALANCE"
   EDIT-STRING -$$$,$$$.99.
```

Because multiple formatting characters tell PRO/DATATRIEVE how to format an entire value, an edit string cannot contain both multiple dollar and multiple plus or minus signs.

Examples

Define a 100-character text field and use a T edit string to display text stored in the field on 25-character lines:

```
DTR> DEFINE RECORD EVAL-REC USING
DFN> 01 PERSON.
DFN>    03 LAST-NAME PIC IS X(10).
DFN>    03 FIRST-NAME PIC IS X(10).
DFN>    03 EVAL-DATE USAGE IS DATE.
DFN>    03 SUMMARY PIC X(100)
DFN>       EDIT-STRING IS T(25).
DFN> ;
[Record EVAL-REC is 128 bytes long]
DTR>
```

PRO/DATATRIEVE displays data stored in EVAL-REC like this:

```
   LAST       FIRST       EVAL
   NAME        NAME       DATE                SUMMARY

ALLEN       JAMES       5-May-83     Jim is a very
                                     conscientious worker who
                                     follows through on his
                                     ideas and plans.
BROGAN      LOUISE      19-May-83    Louise has mastered
                                     PRO/DATATRIEVE and can
                                     now help newcomers to our
                                     company with it.
HASWELL     WAYNE       9-Jun-83     Wayne has demonstrated
                                     exceptional abilities to
                                     generate reports with
                                     PRO/DATATRIEVE.
```

Define a date variable with an edit string the displays the day of the week, a space, the day number, a space, the month name, a space, and the four-digit year:

```
DTR> SET NO PROMPT
DTR> DECLARE TEST USAGE IS DATE
CON>    EDIT-STRING IS W(9)BDDBM(9)BY(4).
DTR> TEST = "TODAY"
DTR> PRINT TEST

            TEST

Friday      1 July       1983

DTR>
```

Use edit strings to suppress leading zeros for all numeric fields except ITEM-NUMBER, display the UNIT-PRICE and TOTAL-VALUE fields as money values, and display the date field as a three-letter month abbreviation followed by the day number and a four-digit year:

```
DTR> DEFINE RECORD EDIT-TEST-REC USING
DFN> 01 EDIT-TEST-REC.
DFN>    03 ITEM-NUMBER PIC IS 9(6).
DFN>    03 QUANTITY PIC IS 9(5)
DFN>       EDIT-STRING IS Z(5).
DFN>    03 UNIT-PRICE PIC IS 99V99
DFN>       EDIT-STRING IS $$$.$$
DFN>       QUERY-NAME IS PRICE.
DFN>    03 TOTAL-VALUE EDIT-STRING IS $$$$,$$$.$$
DFN>       COMPUTED BY QUANTITY * PRICE.
DFN>    03 LAST-SALE USAGE IS DATE
DFN>       EDIT-STRING IS MMMBDDBYYYY.
DFN> ;
[Record EDIT_TEST_REC is 24 bytes long]
DTR>
```

PRO/DATATRIEVE displays the data stored in EDIT-TEST-REC like this:

```
DTR> PRINT EDIT-TEST-REC

 ITEM              UNIT      TOTAL         LAST
NUMBER QUANTITY  PRICE      VALUE         SALE

000099     100     $1.99       $199.00 Apr  5 1982
000101     567     $2.13     $1,207.71 Jan  7 1979

DTR>
```

DEFINING ALTERNATE FIELD NAMES

Use the QUERY-NAME clause to specify an alternate name for a field or variable. You can use the query name anywhere you can use the field or variable name.

Format

```
QUERY-NAME IS query-name
```

Explanation

A *query-name*, like a record name, can consist of from 1 to 31 letters, numbers, hyphens, and underscores. It must begin with a letter and end with a letter or number and it should not duplicate the name of another field in the record.

Comments

You may omit the word IS from the QUERY-NAME clause to save typing time.

To save typing time, use the QUERY-NAME clause to define a shorter name for a field.

You can specify query names for both group and elementary fields.

Example

Define a record for the EMP domain that uses query names for the elementary fields EMPLOYEE-STATUS, FIRST-NAME, and LAST-NAME, and for the group field EMPLOYEE-NAME:

```
DTR> DEFINE RECORD EMP-REC USING
DFN> 01 PERSON.
DFN>    05 EMPLOYEE-STATUS PIC IS X(11)
DFN>       QUERY-NAME IS STATUS.
DFN>    05 EMPLOYEE-NAME QUERY-NAME IS NAME.
DFN>       10 FIRST-NAME PIC IS X(10)
DFN>          QUERY-NAME IS F-NAME.
DFN>       10 LAST-NAME PIC IS X(10)
DFN>          QUERY-NAME IS L-NAME.
DFN> ;
[Record EMP_REC is 31 bytes long]
DTR>
```

DEFINING ALTERNATE COLUMN HEADERS

Use the QUERY-HEADER clause to specify an alternate column header for PRO/DATATRIEVE to use when displaying or printing a field or variable.

Format

```
QUERY-HEADER IS "header-text" ["header-text"]...
```

Explanation

"Header-text" specifies the text you want displayed as the column header. If you want to split a long header over more than one line, type a slash (/) and then another quoted string of header text.

PRO/DATATRIEVE centers a single-line or a multiple-line column header over the field it describes when you display the field. If you do not use this clause, PRO/DATATRIEVE uses the field name as the column header.

Comments

You may omit the word IS from the QUERY-HEADER clause.

You cannot use this clause to describe group fields. Only elementary fields can have column headers.

Examples

Include a QUERY-HEADER clause in the EMP-REC for the STATUS field.

```
DTR> DEFINE RECORD EMP-REC USING
DFN> 01 PERSON.
DFN>     05 EMPLOYEE-STATUS PIC IS X(11)
DFN>        QUERY-HEADER IS "STATUS"
DFN>        QUERY-NAME IS STATUS.
DFN>     05 EMPLOYEE-NAME QUERY-NAME IS NAME.
DFN>        10 FIRST-NAME PIC IS X(10)
DFN>           QUERY-NAME IS F-NAME.
DFN>        10 LAST-NAME PIC IS X(10)
DFN>           QUERY-NAME IS L-NAME.
DFN> ;
[Record EMP_REC is 31 bytes long]
DTR>
```

DEFINING ALTERNATE FIELD DEFINITIONS

Use the REDEFINES clause to specify an alternate field definition for another field in the record. The alternate field refers to the same portion of the record as the original field definition, but tells PRO/DATATRIEVE to interpret data in the field differently.

Format

```
alternate-field-name REDEFINES original-field-name
```

Explanation

A REDEFINES field has a level number and field name, just as the original field does. Both the original field and the REDEFINES field must have the same level number.

The *alternate-field-name* follows the level number and specifies a new name for the field you are redefining. Like any other field name, the alternate name can consist of from 1 to 31 letters, numbers, hyphens, and underscores. It must begin with a letter and end with a letter or number and it should not duplicate the name of another field in the record.

Original-field-name is the group or elementary field you are redefining. This field definition must appear in the record definition before its redefinition. It cannot itself contain a REDEFINES clause.

Neither the original field nor the alternate field definition can contain an OCCURS DEPENDING or a COMPUTED BY clause.

Comments

The REDEFINES clause is particularly useful if you need to refer to parts of a numeric field as well as to the whole numeric field. You can, for example, redefine a 10-digit part number field as a group field that consists of 4 parts that add up to 10 digits.

You can define an alternate field as a smaller field than the original, but not as a larger field.

Example

Define an inventory record that contains a 10-digit part number. Include two REDEFINES group fields that redefine the digits in the part number field as component parts and groups. The first REDEFINES field breaks the part number into four part codes:

- A two-digit product group code
- A two-digit product year code
- A one-digit assembly code
- A five-digit supervisor code

The second REDEFINES field breaks the part number into two group components:

- A four-digit group ID
- A six-digit detail ID

You can then work with the 10-digit field itself or with its component parts, while storing only the original 10 digits:

```
DTR> DEFINE RECORD PART-REC USING
DFN> 01 PART-RECORD.
DFN>    05 PART-NAME PIC X(10).
DFN>    05 PART-NUMBER PIC 9(10).
DFN>    05 PART-NUMBER-PARTS REDEFINES PART-NUMBER.
DFN>       07 PRODUCT-GROUP PIC 99.
DFN>       07 PRODUCT-YEAR PIC 99.
DFN>       07 ASSEMBLY-CODE PIC 9.
DFN>       07 SUP-ASSEMBLY PIC 9(5).
DFN>    05 PART-NUMBER-GROUPS REDEFINES PART-NUMBER.
DFN>       07 PRODUCT-GROUP-ID PIC 9(4).
DFN>       07 PART-DETAIL-ID PIC 9(6).
DFN> ;
[Record PART_REC is 20 bytes long]
DTR>
```

DEFINING VALIDATION CRITERIA

Use the VALID IF clause to establish validation criteria for field and variable values. When a field definition includes a VALID IF clause, PRO/DATATRIEVE checks each value you attempt to store in the field against the validation criteria.

Format

```
VALID IF boolean-expression
```

Explanation

When storing or modifying a value in a field defined with a VALID IF clause, PRO/DATATRIEVE evaluates the *boolean-expression* in the VALID IF clause and stores the value if the expression is true. If the expression evaluates to false, PRO/DATATRIEVE displays a message and does not store the value. If you typed the value in response to a STORE or MODIFY prompt, PRO/DATATRIEVE prompts again for a value.

You cannot validate a COMPUTED BY field, so a field definition cannot include both a VALID IF and COMPUTED BY clause.

Comments

To validate values based on dictionary tables, use the operators IN and NOT IN in the Boolean expression.

Examples

Define a record for phone numbers that uses the AREA-TABLE table to validate area code values. This protects you from storing an incorrect area code value:

```
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>     03 NAME PIC X(20).
DFN>     03 NUMBER PIC 9(7)
DFN>        EDIT-STRING IS XXX-XXXX.
DFN>     03 AREA-CODE PIC XXX VALID IF AREA-CODE IN AREA-TABLE.
DFN> ;
[Record PHONE is 30 bytes long]
DTR>
```

Define a transaction record with VALID IF clauses that check for the following:

- For the TRANS-TYPE field, allow only an O (order) or an S (sales).
- For the COMPANY-NAME field, allow only company name codes that match codes stored in the COMPANIES table.
- For the ITEM-NUMBER and QUANTITY fields, allow only values not equal to zero.

```
DTR> DEFINE RECORD TRANS-REC USING
DFN> 01 TRANSACTION.
DFN>    15 TRANS-TYPE PIC IS X(1)
DFN>       VALID IF TYPE EQ "S" OR TYPE EQ "O"
DFN>       QUERY-NAME IS TYPE.
DFN>    15 TRANS-DATE USAGE IS DATE
DFN>       QUERY-NAME IS T-DATE.
DFN>    15 COMPANY-NAME PIC IS X(3)
DFN>       VALID IF COMPANY-NAME IN COMPANIES
DFN>       QUERY-NAME IS COMPANY.
DFN>    15 ITEM-NUMBER PIC IS 9(5)
DFN>       VALID IF ITEM-NUMBER NE 0.
DFN>    15 QUANTITY PIC IS 9(5)
DFN>       EDIT-STRING IS Z(5)
DFN>       VALID IF QUANTITY NE 0.
DFN>    15 UNIT-PRICE PIC IS S9(5)V99
DFN>       EDIT-STRING IS $$$,$$$.$$
DFN>       QUERY-NAME IS PRICE.
DFN>    15 TOTAL-TRANS EDIT-STRING IS $$$,$$$.$$
DFN>       COMPUTED BY QUANTITY * UNIT-PRICE.
DFN> ;
[Record TRANS_REC is 30 bytes long]
DTR>
```

DEFINING COMPUTED BY FIELDS

Use the COMPUTED BY clause to define a field or variable whose value is determined by other values. The other values can be values stored in other fields in the record, values stored in other variables, values stored in dictionary tables, or literal numeric values.

Format

```
COMPUTED BY expression
```

Explanation

Expression specifies the computation that determines the field's value. It can consist of field names, query names, variable names, numbers, arithmetic operators, and table names.

Because a COMPUTED BY field does not actually exist in a record, you cannot assign a value to it and you cannot use the field as a sort key or an index key.

You do not have to include any other field definition clauses for a COMPUTED BY field. However, it is a good idea to include an EDIT-STRING clause to make sure PRO/DATATRIEVE displays the COMPUTED BY value the way you want.

Comments

COMPUTED BY fields can save you typing time and calculation time. They also save space in records, because PRO/DATATRIEVE does not store COMPUTED BY values in the physical record. Each time you refer to a COMPUTED BY field or variable in a statement, PRO/DATATRIEVE calculates a value for the field.

You can, for example, have PRO/DATATRIEVE calculate a value for a field named SALARY by multiplying the values stored in WAGE and HOURS. Define the SALARY field like this:

```
DFN>    03 WAGE PIC IS 99V99
DFN>       EDIT-STRING IS $$$.$$.
DFN>    03 HOURS PIC IS 99V99
DFN>       EDIT-STRING IS ZZ.ZZ.
DFN>    03 SALARY EDIT-STRING IS $$$$.$$
DFN>       COMPUTED BY WAGE * HOURS.
```

When you store values in these fields, PRO/DATATRIEVE prompts for WAGE and HOURS, but not for SALARY. When you display records, though, you see the SALARY value.

By using COMPUTED BY to refer to a table, you can save typing time while still including the information you want in a record. You might, for example, define a table that contains short department numbers and longer department names:

```
DTR> DEFINE TABLE DEPT-TABLE
DFN>     "PE" : "Plant Engineering",
DFN>     "CS" : "Customer Support",
DFN>     "RD" : "Research and Development",
DFN>     "SD" : "Sales Department"
DFN> END-TABLE
DTR>
```

You could then define a DEPT field that contains short department codes and an AREA field that PRO/DATATRIEVE computes by looking in DEPT-TABLE:

```
DFN>     03 DEPT PIC XX.
DFN>     03 AREA EDIT-STRING IS T(25)
DFN>        COMPUTED BY DEPT VIA DEPT-TABLE.
```

When you store values in these fields, PRO/DATATRIEVE prompts for a two-character DEPT value, then looks in DEPT-TABLE to determine the longer AREA value. When you print records containing these fields, you see the department translation, even though you never stored it in the record:

```
DTR> PRINT NAME, DEPT, AREA OF PAY

   LAST       FIRST
   NAME        NAME     DEPT            AREA

JONES        JOHN        PE    Plant Engineering
SMITH        MARY        RD    Research and Development
WALLACE      FRANK       SD    Sales Department

DTR>
```

If a department name changes, you need to change the name only once, in the table, to keep your payroll information up-to-date.

Examples

Define a procedure to declare a COMPUTED BY variable that concatenates values stored in the FIRST-NAME and LAST-NAME fields of a record to make a neater name display. Then invoke the procedure and use the NEAT-NAME variable to display the PAY domain names:

```
DTR> DEFINE PROCEDURE NEAT-NAME
DFN>     DECLARE NEAT-NAME
DFN>         COMPUTED BY FIRST-NAME|" "|LAST-NAME.
DFN> END-PROCEDURE
DTR> :NEAT-NAME
DTR> READY PAY
DTR> PRINT NEAT-NAME, WAGE, AREA OF PAY

        NEAT
        NAME            WAGE            AREA

JOHN JONES              $5.43 Plant Engineering
MARY SMITH              $9.54 Commercial Engineering
FRANK WALLACE           $7.47 Sales Department

DTR>
```

Define a procedure to multiply two values and display the answer as a COMPUTED BY value:

```
DTR> DEFINE PROCEDURE MULTIPLY
DFN>    DECLARE A PIC 99.
DFN>    DECLARE B PIC 99.
DFN>    DECLARE ANSWER
DFN>        EDIT-STRING ZZZZ COMPUTED BY A * B.
DFN>    PRINT "This procedure multiplies two two-digit numbers."
DFN>    A = *."First number"
DFN>    B = *."Second number"
DFN>    PRINT SKIP, "The answer is "|ANSWER
DFN> END-PROCEDURE
DTR> :MULTIPLY
This procedure multiplies two two-digit numbers.

Enter First number: 25
Enter Second number: 25

The answer is 625

DTR>
```

DEFINING A FIXED-LENGTH LIST FIELD

Use the OCCURS clause to define a fixed-length list field. By defining a field as a list, you can store more than one value for a field or group of fields in one record. Lists are also called repeating fields, and records that contain lists are called hierarchical records. A fixed-length list can contain only a specified number of items.

Format

```
OCCURS number TIMES
```

Explanation

Number tells PRO/DATATRIEVE the maximum number of times the field can occur in the record. PRO/DATATRIEVE reserves space in each record for the maximum number of occurrences, even though some records may contain fewer occurrences.

Because PRO/DATATRIEVE does not store COMPUTED BY values in the physical record, a list field definition cannot contain a COMPUTED BY clause.

Comments

You can define any number of group and elementary fields in a record as fixed-length lists. Furthermore, you can define a group field as a list and a field within the group field as a sublist.

When you store values in a list field, PRO/DATATRIEVE prompts for each occurrence of the field or fields. If you do not want to store the maximum number of items in the list, press the space bar in response to the prompts.

Examples

Define a record with a group list, KIDS-NAMES, that can occur a maximum of three times. Within that list, specify another group list for school subjects and averages, AVG-GRADES, that can occur a maximum of five times:

```
DTR> DEFINE RECORD KIDS-REC USING
DFN> 01 KIDS.
DFN>     03 LAST-NAME PIC X(10).
DFN>     03 KIDS-NAMES OCCURS 3 TIMES.
DFN>        05 FIRST-NAME PIC X(10).
DFN>        05 MIDDLE-INITIAL PIC X(1).
DFN>        05 AGE PIC 99
DFN>           EDIT-STRING IS Z9.
DFN>        05 GRADE PIC 99
DFN>           EDIT-STRING IS Z9.
DFN>        05 AVG-GRADES OCCURS 5 TIMES.
DFN>           07 SUBJECT PIC X(10).
DFN>           07 AVERAGE PIC 999
DFN>              EDIT-STRING IS Z99.
DFN> ;
[Record KIDS_REC is 250 bytes long]
DTR>
```

PRO/DATATRIEVE displays data stored in KIDS-REC like this:

```
   LAST       FIRST     MIDDLE
   NAME        NAME     INITIAL AGE GRADE  SUBJECT    AVERAGE

ALLEN       JENNIFER       M     12    6   MATH           92
                                           READING        94
                                           SCIENCE        89
                                           SOC STUD       84
                                           LANG           93
            LESLIE         M      9    4   MATH           91
                                           READING        92
                                           SCIENCE        93
                                           SOC STUD       89
                                           LANG           90
            JONATHAN       T      7    2   MATH           83
                                           READING        79
                                           SCIENCE        83
                                           SOC STUD       81
                                           LANG           84
```

This domain contains a list within a list. To display just the FIRST-NAME field and the AVG-GRADES list field from the first child in the KIDS-NAMES list in the first record, you must establish context for the list fields. In the following example, the first FOR statement establishes context for the KIDS domain. The second FOR statement then establishes context for the KIDS-NAMES list:

```
DTR> SET NO PROMPT
DTR> FOR FIRST 1 KIDS
CON>     FOR FIRST 1 KIDS-NAMES
CON>         PRINT FIRST-NAME, AVG-GRADES

  FIRST
   NAME      SUBJECT    AVERAGE

JENNIFER    MATH           92
            READING        94
            SCIENCE        89
            SOC STUD       84
            LANG           93

DTR>
```

See Chapter 4 for more information on displaying list fields.

Define a record for supplies with two fixed-length list fields, SUPPLIERS and DISCOUNT:

```
DTR> DEFINE RECORD SUPPLIERS-REC USING
DFN> 01 WHAT.
DFN>    05 ITEM PIC X(15).
DFN>    05 ORDER-UNIT PIC X(10).
DFN>    05 SUPPLIERS OCCURS 3 TIMES.
DFN>       07 COMPANY PIC X(10).
DFN>       07 PRICE PIC 99V99
DFN>          EDIT-STRING IS $$$.$$.
DFN>       07 DISCOUNT OCCURS 3 TIMES.
DFN>          09 MIN-ORDER PIC X(10).
DFN>          09 PERCENT PIC 99
DFN>             EDIT-STRING IS Z9%.
DFN>          09 WE-PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
DFN>             EDIT-STRING IS $$$.$$.
DFN> ;
[Record SUPPLIERS_REC is 175 bytes long]
DTR>
```

This record definition specifies that the group field, SUPPLIERS, can occur three times. Within that list, the DISCOUNT group field can occur three times. PRO/DATATRIEVE displays data stored in SUPPLIERS like this:

```
DTR> PRINT FIRST 1 SUPPLIERS

                ORDER                                MIN
     ITEM        UNIT      COMPANY     PRICE       ORDER      PERCENT

PAPER           REAM      ABC SUPPLY  $11.35  50              3%
                                                  250             4%
                                                  500             6%
                          HARRIDON    $10.86  100             4%
                                                  500             6%
                                                  1000            9%
                          MORRISON    $11.03  10              2%
                                                  50              4%
                                                  100             6%

DTR>
```

DEFINING A VARIABLE-LENGTH LIST FIELD

Use the OCCURS DEPENDING clause to define a variable-length list. The length of a variable-length list is determined by the value of another field in the record.

Format

```
OCCURS min-number TO max-number DEPENDING ON field-name
```

Explanation

Min-number and *max-number* specify the minimum and maximum number of list items the field can contain. *Min-number* can be zero or any positive integer. *Max-number* must be greater than zero and greater than *min-number*.

The actual number of list items in each record varies depending on the value stored in another field in the record, as specified by *field-name*. If, for example, you specify a minimum of 0 and a maximum of 10, a record can contain from zero to ten list items. A record with the value 3 stored in the specified field would contain three list items, while a record with the value 0 in the specified field would contain no list items.

Field-name must be a numeric field and cannot be defined to contain decimal or negative values.

A field definition cannot contain an OCCURS and a COMPUTED BY or REDEFINES clause. PRO/DATATRIEVE does not store COMPUTED BY or REDEFINES values in the physical record.

Comments

Because the size of a record that contains an OCCURS DEPENDING clause varies, define the file containing the records as an indexed file. This way you can add items to a list by increasing the value of the field that determines the length of the list. If you define a sequential file for records with a variable-length list, you must include the MAX keyword when you define the file so that PRO/DATATRIEVE allocates the maximum amount of space for the list field. Otherwise, you cannot add items to the list after you have stored a record. See Chapter 7 for information on defining data files.

You can specify OCCURS DEPENDING for only one group or elementary field in a record. Furthermore, the OCCURS DEPENDING field must be the last elementary or group field in the record definition. No other field definition can follow a variable-length list field.

A group field defined as a variable-length list with the OCCURS DEPENDING clause can contain one or more fields defined as a fixed-length list with the OCCURS clause. A group field defined as a fixed-length list cannot contain a field defined as a variable-length list. For example:

```
DTR> SHOW REPS-REC
RECORD REPS-REC
 USING
01 JOB.
   03 TERRITORY PIC X(15).
   03 NUMBER_IN_JOB PIC 9.
   03 REP OCCURS 0 TO 8 TIMES DEPENDING ON NUMBER_IN_JOB.
      05 LAST_NAME PIC X(10).
      05 FIRST_NAME PIC X(10).
      05 CUSTOMERS PIC X(15) OCCURS 5 TIMES.
;
DTR>
```

This record defines the variable-length list, REP, first, then defines the fixed-length list, CUSTOMERS, within the REP list. If you reverse this order in a record definition, PRO/DATATRIEVE displays a message telling you that OCCURS DEPENDING is not allowed in an OCCURS list.

Example

Define a record to track supplies and compare suppliers' prices and discounts. In the record, include a variable-length group field to list suppliers of an item, each supplier's price, and three different order and discount values:

```
DTR> DEFINE RECORD SUPPLIES-REC USING
DFN> 01 WHAT.
DFN>    05 ITEM PIC X(15).
DFN>    05 ORDER-UNIT PIC X(10).
DFN>    05 HOW-MANY PIC 9.
DFN>    05 VENDORS OCCURS 0 TO 10 TIMES DEPENDING ON HOW-MANY.
DFN>       07 COMPANY PIC X(10).
DFN>       07 PRICE PIC 99V99
DFN>          EDIT-STRING IS $$$.$$.
```

```
DFN>        07 DISCOUNT OCCURS 3 TIMES.
DFN>           09 MIN-ORDER PIC X(10).
DFN>           09 PERCENT PIC 99
DFN>              EDIT-STRING IS Z9%.
DFN>           09 WE-PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
DFN>              EDIT-STRING IS $$$.$$.
DFN> ;
[Record SUPPLIES_REC is 526 bytes long]
DTR>
```

PRO/DATATRIEVE displays data stored in SUPPLIES-REC like this:

```
DTR> PRINT FIRST 2 SUPPLIES

                  ORDER     HOW                              MIN                 WE
     ITEM          UNIT     MANY  COMPANY    PRICE          ORDER     PERCENT  PAY

PAPER             REAM        2   ABC SUPPLY $13.45 100                   4%   $12.91
                                                    500                   5%   $12.77
                                                    1000                  7%   $12.50
                                    MORRIS     $12.98 10                    2%   $12.72
                                                    50                    4%   $12.46
                                                    100                   6%   $12.20
PENS              GROSS       3   ABC SUPPLY  $3.21 12                    1%    $3.17
                                                    50                    2%    $3.14
                                                    500                   4%    $3.08
                                    MORRIS     $2.96 100                   2%    $2.90
                                                    500                   3%    $2.87
                                                    1000                  4%    $2.84
                                    HARRIDON   $3.12 10                    1%    $3.08
                                                    100                   3%    $3.02
                                                    500                   5%    $2.96

DTR>
```

9

Readying and Finishing Domains and Releasing Tables

Chapter 9

Readying and Finishing Domains and Releasing Tables

This chapter describes the commands you use to ready and finish domains and to release tables:

Readying Domains
Finishing Domains
Releasing Tables

When you ready a domain or access a table, you make the domain or table available in memory for your access. Then, because a ready domain takes up space in memory, you end your control over it by finishing it. An accessed table also takes up space in memory, so you must release it to remove it.

This chapter includes an example that shows an alternate method of restructuring by readying a domain with an alias. By using this method, you can change a record definition or the type of data file used by a domain without redefining the domain.

READYING DOMAINS

Use the READY command to gain access to a domain and to specify the type of access privilege you need.

Format

```
READY domain-name [AS alias] [access]

access: { READ
        { WRITE
        { MODIFY
```

Explanation

Domain-name is the name of the domain you want to access.

Alias is an alternative name for the domain. You can refer to the domain by the alias in PRO/DATATRIEVE commands and statements and save yourself typing time. You can, for example, write a procedure that creates a mailing list for a domain named MAILING, then ready the CUSTOMERS, EMPLOYEES, or any other domain with the alias MAILING to create a mailing list.

Access specifies the type of access you want to the domain and determines what you can do with records in the domain. PRO/DATATRIEVE allows the following types of access to domains:

- READ — You can only *read* records.
- MODIFY — You can read and *modify* records.
- WRITE — You can read, modify, *store*, and *erase* records.

PRO/DATATRIEVE readies the domain for READ access if you do not specify an access keyword when you ready a domain. READY CUSTOMERS and READY CUSTOMERS READ are equivalent commands. With READ access, you can retrieve records, but you cannot store, modify, or erase them.

Comments

You cannot ready a domain unless both the domain definition and its associated record definition are stored in your current dictionary. You also cannot ready a domain unless the associated data file for the domain exists.

If you do not specify an alias for the domain, you must use the domain name to refer to the domain in commands and statements. If you do specify an alias for the domain name, you must use the alias to refer to the domain in commands and statements. You cannot access the domain by its domain name if you have readied it with an alias. To access the domain by its domain name, you must finish the readied alias with the FINISH command and ready the domain without an alias.

If you want to ready a domain readied with an alias, you must use the same alias specified in the previous READY command. If you do not use the same alias, PRO/DATATRIEVE displays an error message and does not ready the domain.

You can ready a domain that is currently readied, with the following effects:

- The access specified in the new READY command replaces the previous access.
- Collections formed from the domain when it was first readied remain available to you.
- If you redefine the format of the record associated with a readied domain, the modified record definition does not take effect when you ready the domain again. To use the modified record definition, you must finish the domain with the FINISH command and then ready it again.

Use the SHOW READY command to show information about readied domains. PRO/DATATRIEVE displays the domain name, file type, and access privilege of all readied domains, with the most recently readied domain at the top of the display.

Statements that operate on domains require different access privileges:

- ERASE and STORE require *WRITE* access
- MODIFY requires *MODIFY* or *WRITE* access
- FIND, PRINT, REPORT, SORT, and SUM require *MODIFY*, *READ*, or *WRITE* access

Use the SHOW READY command to make sure you have readied the target domain with the appropriate access.

A domain stays ready until you finish it with the FINISH command or exit from PRO/DATATRIEVE. The FINISH command ends your control over a domain, releases all collections formed from the domain, and frees the computer memory used by the domain.

If you want to change a record definition or the type of data file used by a domain without redefining the domain, follow these steps to define a new data file and transfer the data with the STORE statement:

1. Ready the domain as an alias:

```
DTR> READY CUSTOMERS AS OLD-CUSTOMERS
DTR> SHOW READY
Ready domains:
        OLD_CUSTOMERS: RMS SEQUENTIAL, PROTECTED READ
DTR>
```

2. If you want to change the record definition, use the PRO/DATATRIEVE Editor to change the record definition or use the DEFINE RECORD command to redefine the record.

3. Define a new data file for the domain:

```
DTR> DEFINE FILE FOR CUSTOMERS KEY = COMPANY;
DTR>
```

   This DEFINE FILE statement creates a new indexed version of the file associated with the readied domain, but does not affect the link between the readied domain and the original sequential version of the file.

4. Ready the domain with a different alias for WRITE access:

```
DTR> READY CUSTOMERS AS NEW-CUSTOMERS WRITE
DTR> SHOW READY
Ready domains:
        NEW_CUSTOMERS: RMS INDEXED, PROTECTED WRITE
        OLD_CUSTOMERS: RMS SEQUENTIAL, PROTECTED READ
DTR>
```

   This READY command uses the new record definition, if you changed it, and opens the indexed data file created by the DEFINE FILE command.

5. Use a FOR loop to move data from the original sequential file associated with OLD-CUSTOMERS to the new indexed file associated with NEW-CUSTOMERS. PRO/DATATRIEVE transfers data from fields in the original data file into fields with the same names in the new data file:

```
DTR> SET NO PROMPT
DTR> FOR A IN OLD-CUSTOMERS
CON> STORE NEW-CUSTOMERS USING CUSTOMERS-REC = A.CUSTOMERS-REC
DTR> SHOW READY
Ready domains:
        NEW_CUSTOMERS: RMS INDEXED, PROTECTED WRITE
        OLD_CUSTOMERS: RMS SEQUENTIAL, PROTECTED READ
DTR>
```

6. Type FINISH to finish both domains, then ready the CUSTOMERS domain and use the SHOW READY command to see that the data file associated with the domain is now an indexed file:

```
DTR> READY CUSTOMERS
DTR> SHOW READY
Ready domains:
        CUSTOMERS: RMS INDEXED, PROTECTED READ
DTR>
```

This method of redefining a domain is particularly useful if you want to convert a file to a different type of file. You can convert indexed files that are not often used or modified to sequential files, because sequential files take up less space on a diskette. You can also use this technique to change the primary key of an indexed file so that it allows duplicates.

Examples

Ready the domain CUSTOMERS as C for WRITE access. You can then refer to the domain by the shorter alias to save yourself typing time:

```
DTR> READY CUSTOMERS AS C WRITE
DTR>
```

Ready the EMPLOYEES domain for READ access. You do not have to specify READ, as PRO/DATATRIEVE automatically grants the READ access:

```
DTR> READY EMPLOYEES
DTR>
```

FINISHING DOMAINS

Use the FINISH command to end your control over one or all domains and to release any collections formed from the domain or domains.

Format

```
FINISH [domain-name],...
```

Explanation

Domain-name is the name of a readied domain you want to finish. Separate multiple domain names with commas.

If you do not specify the names of any domains, PRO/DATATRIEVE ends your control over all readied domains. If you specify the names of one or more domains, PRO/DATATRIEVE ends your control only over the specified domains.

When you finish a domain, PRO/DATATRIEVE also finishes all collections formed from the specified domain, but does not remove any tables from memory. To remove tables from memory, use the RELEASE command.

Comments

Use the FINISH command to clear your computer's memory of domains you no longer need. You do not need to use the FINISH command before you exit from PRO/DATATRIEVE. PRO/DATATRIEVE automatically finishes all readied domains and releases all collections and dictionary tables when you return to the Main Menu.

If you change the definition of a readied domain or its associated record, the changes do not take effect until you finish the domain and ready it again.

If you delete the definition of a readied domain or its associated record, you can continue to work with the domain until you end your control over it with the FINISH command. You can even ready the domain again to change your access to it, as long as you do not finish the domain. Once you finish a domain, the domain and record definition are removed from memory. If you have deleted either definition, you can no longer access the domain.

Examples

Use the FINISH command to end your control of the EMPLOYEES domain:

```
DTR> SHOW READY
Ready domains:
        EMPLOYEES: RMS INDEXED, PROTECTED WRITE
        CUSTOMERS: RMS INDEXED, PROTECTED READ
DTR> FINISH EMPLOYEES
DTR> SHOW READY
Ready domains:
        CUSTOMERS: RMS INDEXED, PROTECTED READ
DTR>
```

Define a domain and a simple record. Then delete and redefine the record to show how new definitions do not take effect until the domain is finished and readied again:

```
DTR> DEFINE DOMAIN D USING D-REC ON D;
DTR> DEFINE RECORD D-REC USING
DFN> 01 TOP PIC X.
DFN> ;
[Record D_REC is 1 bytes long]
DTR> DEFINE FILE FOR D
DTR> READY D WRITE
DTR> STORE D
Enter TOP: A
DTR> DELETE D-REC;
DTR> SHOW READY
Ready domains:
        D: RMS SEQUENTIAL, PROTECTED WRITE
DTR> DEFINE RECORD D-REC USING
DFN> 01 TOP PIC XX.
DFN> ;
[Record D_REC is 2 bytes long]
DTR> STORE D
Enter TOP: BB
Truncation during assignment
Re-enter TOP: B
DTR> PRINT ALL D

TOP

 A
 B

DTR> FINISH D
DTR> READY D WRITE
File and domain lengths don't match (D=2)
DTR> DEFINE FILE FOR D
DTR> READY D WRITE
DTR> STORE D
Enter TOP: AA
DTR>
```

RELEASING TABLES

Use the RELEASE command to end your control over one or more tables and to free the computer memory they occupy.

Format

```
RELEASE table-name,...
```

Explanation

Table-name is the name of the table you want to release. To release more than one table, separate table names with commas.

Comments

You can also use the RELEASE command to release collections and global variables. Simply list the tables and/or variables you want to release as you would multiple table names.

When the RELEASE command executes, PRO/DATATRIEVE releases the specified tables, collections, and variables by removing them from memory. The effect is very much like the FINISH command.

When you specify more than one table, collection, or variable in the RELEASE command, PRO/DATATRIEVE releases the items in left-to-right order. If PRO/DATATRIEVE cannot release an item, it displays a message indicating which item it could not release, releases all items in the command preceding the one that it could not release, and does not release any items that follow.

Use the SHOW TABLES command to show tables currently in memory and the names of tables defined in your current dictionary:

```
DTR> SHOW TABLES
Tables loaded:
        COMPANIES
Tables:
        AREA_TABLE        COMPANIES
DTR>
```

You do not have to release a table before exiting from PRO/DATATRIEVE. PRO/DATATRIEVE automatically releases all collections, tables, and global variables and finishes all domains when you return to the Main Menu.

Example

Ready the TRANS domain and display the first five records using the table translations for the COMPANY field. Use the SHOW TABLES command to see that the table has been accessed, then release the table with the RELEASE command:

```
DTR> SET NO PROMPT
DTR> READY TRANS
DTR> PRINT COMPANY VIA COMPANIES USING X(20),
CON>     TRANS-DATE OF FIRST 5 TRANS

      COMPANY          TRANS
        NAME            DATE

TACTICAL AID, INC.     3/04/83
PAPER & PEN, INC.      3/03/83
PROSE PROS             3/03/83
WRITE RIGHT ASSOC.     3/05/83
STONE ASSOCIATES       3/03/83

DTR> SHOW TABLES
Tables loaded:
        COMPANIES
Tables:
        AREA_TABLE       COMPANIES
DTR> RELEASE COMPANIES
DTR> SHOW TABLES
Tables:
        AREA_TABLE       COMPANIES
DTR>
```

10

Creating and Using Procedures and Command Files

Chapter 10

Creating and Using Procedures and Command Files

This chapter shows how to create and execute procedures, how to write existing dictionary definitions to command files, and how to execute command files:

Creating Procedures
Executing Procedures
Creating Command Files
Executing Command Files

You can save typing time by using command files and procedures. A procedure is a sequence of PRO/DATATRIEVE commands and statements that you store as a named item in your current dictionary. You execute the commands and statements in a procedure by typing a colon (:) and the procedure name in response to the DTR> prompt or in a PRO/DATATRIEVE statement.

You can also edit statements in procedures with the PRO/DATATRIEVE Editor, or you can write the entire procedure to a command file, exit from PRO/DATATRIEVE, and edit the command file with PROSE. If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message.

A command file is a group of PRO/DATATRIEVE statements and commands stored in a file. When you execute a command file, PRO/DATATRIEVE executes the commands and statements in the command file as though you typed them in response to the DTR> prompt.

CREATING PROCEDURES

Use the DEFINE PROCEDURE command to create a procedure and store the procedure definition in your current dictionary.

Format

```
DEFINE PROCEDURE procedure-name

    statement or command
                .
                .
                .

END-PROCEDURE
```

Explanation

Procedure-name names the procedure. PRO/DATATRIEVE stores the sequence of commands and statements you put in the procedure in your current dictionary, using the specified procedure name.

After you type the DEFINE PROCEDURE line, PRO/DATATRIEVE prompts for commands and statements to store in the procedure with the DFN> prompt. To end the procedure, use the END-PROCEDURE command.

Comments

A procedure name cannot duplicate the name of any other definition in your current dictionary. Like other PRO/DATATRIEVE names, it can consist of from 1 to 31 letters, numbers, hyphens, and underscores and should begin with a letter and end with a letter or number.

You cannot define a procedure from within another PRO/DATATRIEVE statement or command.

Do not use any other DEFINE command in a procedure definition. If you do, PRO/DATATRIEVE displays an error message when you try to execute the procedure. Put DEFINE DICTIONARY, DEFINE DOMAIN, DEFINE RECORD, and DEFINE FILE commands in a command file, not in a procedure.

To modify a procedure already stored in your current dictionary, use the PRO/DATATRIEVE Editor. PRO/DATATRIEVE deletes the old procedure definition and stores the updated procedure definition in your current dictionary when you exit from the Editor. See Chapter 17 for more information on using the PRO/DATATRIEVE Editor.

To stop while defining a procedure, press the EXIT key. PRO/DATATRIEVE displays the DTR> prompt and does not store the procedure definition.

Examples

Define a procedure to set your current dictionary to CUSTOMERS.DIC on volume CUST, Then use the SHOW command to show the name of domains stored in the dictionary:

```
DTR> DEFINE PROCEDURE DICT
DFN> SET DICTIONARY CUST:[CUSTOMERS]CUSTOMERS.DIC
DFN> SHOW DICTIONARY
DFN> SHOW DOMAINS
DFN> END-PROCEDURE
DTR> :DICT
The current dictionary is CUST:[CUSTOMERS]CUSTOMERS.DIC;1
Domains:
        CUSTOMERS        SALES_VIEW
DTR>
```

Define a procedure that declares a variable to display the FIRST-NAME and LAST-NAME fields with only one space between them. Then define a procedure that executes the NEAT-NAME procedure and creates a report for the EMPLOYEES domain using the NEAT-NAME variable to display names:

```
DTR> DEFINE PROCEDURE NEAT-NAME
DFN> DECLARE NEAT-NAME COMPUTED BY FIRST-NAME|" "|LAST-NAME.
DFN> END-PROCEDURE
DFN> DEFINE PROCEDURE DEPT-REP
DFN> :NEAT-NAME
DFN> READY EMPLOYEES
DFN> REPORT EMPLOYEES WITH DEPT CONT
DFN>         *,"first three characters of a department"
DFN>     SET REPORT-NAME = *,"a title, in quotes"
DFN>     SET COLUMNS-PAGE = 60
DFN>     PRINT NEAT-NAME, STATUS, ID
DFN> END-REPORT
DFN> END-PROCEDURE
DTR> :DEPT-REP
Enter a title, in quotes: "ASTRONOMY DEPT REPORT"
Enter first three characters of a department: AST
```

(continued on next page)

```
                    ASTRONOMY DEPT REPORT          28-Jun-83
                                                   Page 1

            NEAT
            NAME                    STATUS         ID

      CHARLOTTE SPIVA               EXPERIENCED    00012
      LYDIA HARRISON                EXPERIENCED    78923
      GERALD BOOLE                  EXPERIENCED    83771
      NATHANIEL CHONTZ              TRAINEE        87701
      MARTHA BOOLE                  TRAINEE        93811

DTR>
```

EXECUTING PROCEDURES

To execute a procedure, type a colon (:) and the procedure name.

Format

```
: procedure-name
```

Explanation

The colon tells PRO/DATATRIEVE to execute the procedure specified by *procedure-name*. You can omit the space between the colon and the procedure name to save typing time.

Comments

You can execute a procedure in another statement or in response to the DTR>, RW>, or CON> prompts. You cannot execute a procedure in the DEFINE DOMAIN, DEFINE RECORD, or DEFINE TABLE command, or while you are in ADT, GUIDE Mode, or the PRO/DATATRIEVE Editor.

When you execute a procedure from a REPEAT or FOR loop, enclose the procedure in a BEGIN-END block or enclose the statements in the procedure in a BEGIN-END block. This ensures that PRO/DATATRIEVE executes all the statements in the procedure each time through the loop. Do not include any commands or FIND, SELECT, DROP, or RELEASE statements in a procedure you plan to execute from a BEGIN-END block.

If the procedure consists of full commands or statements, PRO/DATATRIEVE executes each command or statement in the procedure in order. If the procedure contains only a portion of a command or statement, PRO/DATATRIEVE includes the procedure within the command or statement that executed the procedure.

You can execute a procedure from another procedure, but be careful not to allow the procedure to execute itself. You may create an infinite loop. If this ever happens, press the INTERRUPT/DO key sequence twice in succession to end the procedure and exit from PRO/DATATRIEVE.

You can execute a procedure more than once from a REPEAT statement. You can also make the statements in a procedure operate on a group of records by executing the procedure in a FOR statement. Use care, though, when executing a procedure in these statements. The results of the following, for example, may be unexpected:

```
DTR> REPEAT 5 :TEST
```

This REPEAT statement does not execute the TEST procedure 5 times unless the TEST procedure contains only one statement. If the TEST procedure contains more than one statement, PRO/DATATRIEVE executes the first statement in the procedure, then assumes that the REPEAT statement is complete. Therefore, PRO/DATATRIEVE executes the **first** statement in the TEST procedure five times, then executes the remaining statements in the procedure once.

To repeat the entire TEST procedure, execute the procedure in a BEGIN-END block:

```
REPEAT 5
   BEGIN
      :TEST
   END
```

Use a similar technique for controlling procedures executed in a FOR loop. For example:

```
FOR CUSTOMERS
    BEGIN
       :TEST
    END
```

Examples

Define a procedure that displays records for employees with salaries greater than $40,000 for a department you specify, then execute this procedure from a REPEAT statement. Use prompting value expressions to prompt for the number of times to repeat the procedure and for the department:

```
DTR> SET NO PROMPT
DTR> DEFINE PROCEDURE BIG-GUYS
DFN> FOR EMPLOYEES WITH DEPT CONT *,"first 3 characters of department"
DFN>    BEGIN
DFN>      IF SALARY GT 40000
DFN>         THEN PRINT EMPLOYEES-REC
DFN>      END
DFN> END-PROCEDURE
```

```
DTR> READY EMPLOYEES
DTR> REPEAT *,"how many departments"
CON>    BEGIN
CON>       :BIG-GUYS
CON>       PRINT " "
CON>    END
Enter how many departments: 2
Enter first 3 characters of department: MAT

                      FIRST       LAST
 ID     STATUS         NAME       NAME          DEPT        SALARY

00891 EXPERIENCED FRED        HOWL        MATHEMATICS     $59,594
78375 EXPERIENCED ALBERT      EINSTEIN    MATHEMATICS     $40,095
83764 EXPERIENCED JIM         MEADER      MATHEMATICS     $41,029

Enter first 3 characters of department: LIT
38462 EXPERIENCED BILL        SWAY        LITERATURE      $54,000
48475 EXPERIENCED GAIL        CASSIDY     LITERATURE      $55,407


DTR>
```

Define a procedure that creates an edit string clause for a money value. Execute this procedure in a report procedure to display the minimum, maximum, average, and total salaries for departments in the EMPLOYEES domain:

```
DTR> DEFINE PROCEDURE MONEY-EDIT
DFN>    USING $$$$,$$$,$$$
DFN> END-PROCEDURE
DTR> DEFINE PROCEDURE TOTAL-SAL
DFN> READY EMPLOYEES
DFN> FIND EMPLOYEES SORTED BY DEPT
DFN> REPORT CURRENT
DFN> SET COLUMNS-PAGE = 70
DFN> AT BOTTOM OF DEPT PRINT COL 1, DEPT, COL 15,
DFN>    MAX SALARY ("MAXIMUM"/"SALARY") :MONEY-EDIT, COL 28,
DFN>    MIN SALARY ("MINIMUM"/"SALARY") :MONEY-EDIT, COL 41,
DFN>    TOTAL SALARY ("TOTAL"/"SALARY")  :MONEY-EDIT, COL 54,
DFN>    AVERAGE SALARY ("AVERAGE"/"SALARY") :MONEY-EDIT
DFN> END-REPORT
DFN> END-PROCEDURE
DTR> :TOTAL-SAL
```

(continued on next page)

```
                                                    11-Jul-83
                                                    Page 1

                MAXIMUM        MINIMUM        TOTAL          AVERAGE
     DEPT       SALARY         SALARY         SALARY         SALARY

AGRICULTURE
                $26,723        $26,723        $26,723        $26,723
ASTRONOMY
                $75,892        $20,000       $182,141        $36,428
BIOLOGY
                $58,462        $33,738       $169,547        $42,386
GEOGRAPHY
                $21,000        $21,000        $21,000        $21,000
LITERATURE
                $55,407        $23,908       $262,266        $37,466
MATHEMATICS
                $59,594         $2,000       $219,718        $27,464
PHILOSOPHY
                $56,847        $26,392       $200,614        $40,122
PSYCHOLOGY
                $57,598        $25,023       $117,554        $39,184

DTR>
```

CREATING COMMAND FILES

Command files contain PRO/DATATRIEVE commands and statements. When you execute a command file with the @ command, PRO/DATATRIEVE executes the lines of a command file as though you had just typed them.

You can create a new command file with the PROSE editor, or you can use the EXTRACT command to copy one or more existing dictionary definitions to a command file:

- To create a command file with PROSE, choose PROSE from the Main Menu and type in the commands and statements you want included in the command file. When you exit from PROSE, give the file a .CMD file type to identify it as a command file.
- To create a command file that contains existing dictionary definitions, use the EXTRACT command. The EXTRACT command copies the dictionary definitions you specify from your current dictionary to a command file. You can then exit from PRO/DATATRIEVE to edit the definitions in the command file with PROSE, or you can store the copied definitions in a different dictionary. See the section on copying dictionary definitions in Chapter 16 for information on using the EXTRACT command.

If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message.

Include only complete commands and statements in a command file. When you execute a command file that ends with an incomplete command or statement, PRO/DATATRIEVE displays the CON> prompt when the command file finishes executing, indicating that you need to supply the missing elements. When you execute a command file that contains an incomplete command or statement in the middle of the file, PRO/DATATRIEVE looks at the next line in the command file and displays an error message if the command or statement is incomplete.

EXECUTING COMMAND FILES

Use the @ command to execute a command file.

Format

```
@ file-spec
```

Explanation

The *at sign* (@) tells PRO/DATATRIEVE that you want to execute a command file. *File-spec* identifies the command file you want to execute. You can omit the space between the at sign and the file specification to save typing time.

If the command file you want to execute resides in your current directory and has a file type of .CMD, you need to specify only a file name for *file-spec*.

To execute a command file in a different directory, on a different device, or with a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

You must put the @ command on a line by itself.

Comments

When you execute a command file that consists of complete commands and statements, PRO/DATATRIEVE displays and executes each command and statement in order. If the command file ends with an incomplete PRO/DATATRIEVE command or statement, PRO/DATATRIEVE displays the CON> prompt when the command file executes, indicating that you need to supply the missing elements.

When you execute a command file with an incomplete command or statement in the middle of the file, PRO/DATATRIEVE looks at the next line in the file and displays an error message if the command or statement is incomplete.

You cannot execute a command file from a procedure, but you can include a command file in a procedure definition. When you include an @ command in a procedure definition, PRO/DATATRIEVE includes the commands and statements in the command file in the procedure as though you had just typed them yourself.

You cannot execute a command file while you are in ADT or Guide Mode, or from a BEGIN-END statement or a loop created by the FOR, REPEAT, or WHILE statements.

You can execute a command file from another command file, but be careful not to allow a command file to execute itself. You may create an infinite loop. If this ever happens, press the INTERRUPT/DO key sequence twice in succession to end the procedure and exit from PRO/DATATRIEVE.

To include comments in a command file, place an exclamation mark (!) before each comment line.

Example

Execute a command file, SALARIES.CMD, to display employee records with salaries greater than $40,000 for a particular department. PRO/DATATRIEVE displays all lines in the command file, including comments, then prompts for the department.

```
DTR> @SAL
! This file reports on employees who make more than $40,000
!
READY EMPLOYEES
FOR EMPLOYEES WITH DEPT CONT *,"first 3 characters of department"
   BEGIN
     IF SALARY GT 40000
        THEN PRINT EMPLOYEES-REC
     END
Enter first 3 characters of department: MAT

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT         SALARY

00891 EXPERIENCED FRED       HOWL       MATHEMATICS      $59,594
78375 EXPERIENCED ALBERT     EINSTEIN   MATHEMATICS      $40,095
83764 EXPERIENCED JIM        MEADER     MATHEMATICS      $41,029

DTR>
```

11

Displaying Information and Creating Reports

Chapter 11

Displaying Information and Creating Reports

This chapter describes how to use the PRINT statement and the Report Writer to format information.

Use the PRINT statement to display information on your Professional or to write information to a file. Use the Report Writer to organize and summarize information in a formatted report that you can display on your Professional or write to a file.

Among other things, examples in this chapter show how to:

- Format names and create column headers
- Format a mailing list
- Include totals, averages, and maximum and minimum values in a report

(continued on next page)

- Create report title pages
- Include salary increases in a report
- Create summary reports

Appendix A contains a comprehensive report specification for the EMPLOYEES domain and the resulting report.

DISPLAYING INFORMATION

Use the PRINT statement to format and display fields, records, and other values. You can display information on your Professional or you can write the print display to a file. You can form a record stream with the PRINT statement and display information from records in that record stream. You can also use the PRINT statement to display information from a selected record, records in a record stream formed by a FOR statement, and records in a collection.

Format

```
PRINT [ALL] [print-list-element],... [OF rse] [ON file-spec]

print-list-element: { data-item [format-modifier] }
                    { format-item                 }
                    { list-specification          }
```

Explanation

If you do not put the PRINT statement in a FOR loop and do not include an RSE or the word ALL, PRO/DATATRIEVE displays information from the selected record of the most recently formed collection containing a selected record.

ALL, when used alone in the PRINT statement, displays all records in the CURRENT collection. When ALL precedes *print-list-elements* (a print list), PRO/DATATRIEVE displays the specified information for all records in the CURRENT collection.

Print-list-elements identify the data you want to display and specify how you want PRO/DATATRIEVE to format the data. Use commas to separate print list elements.

Data-item identifies the data you want to display:

- *Field name* displays an elementary field, a group field, or a list field. If you specify a group field, PRO/DATATRIEVE displays all of the elementary fields contained in the group. If you specify a list field, PRO/DATATRIEVE displays all the fields contained in the list.
- *Variable name* displays the current contents of a variable.

(continued on next page)

- *Literal* displays a character string or numeric literal. You must enclose a character string literal (such as "Type Y for YES") in quotation marks.
- *Prompting value expression* displays a prompt you specify for a field name or other data item to display. See Chapter 2 for information on forming prompting value expressions.
- *Arithmetic expression* displays the results of an arithmetic operation. See Chapter 2 for information on forming arithmetic expressions.
- *Statistical expression* displays the results of a statistical expression. See Chapter 2 for information on forming statistical expressions.

You can use a *format-modifier* to tell PRO/DATATRIEVE how to display an item, to display the table translation for the item, or to suppress or display a column heading for an item. Table 11-1 lists and describes format modifiers for data items.

Table 11-1: Data Item Modifiers

Modifier	Effect
("column-header")	Specifies a column header for the preceding data item. You must enclose column header text in double quotation marks and the entire modifier in parentheses. To specify a multiline column header, separate one segment of the column header from another with a slash (/), as in ("WHOLE"/"NAME"). When you use this modifier, the specified column header overrides the column header or query header specified in the definition of a field or variable.
(-)	Suppresses the printing of a column header for the preceding data item. You must enclose the hyphen in parentheses.
USING edit-string	Tells PRO/DATATRIEVE how to format the preceding data item. The specified edit string overrides any edit string specified in the definition of a field or variable. See the section on the EDIT-STRING clause in Chapter 7 for information on forming edit strings. If you follow an edit string with another print list element, include a space between the last character of the edit string and the comma that separates one print list element from another. Otherwise, PRO/DATATRIEVE will consider the comma part of the edit string.
VIA table-name	Displays the table translation for the preceding field or variable. If you do not specify an edit string for a table translation, PRO/DATATRIEVE displays a maximum of 10 characters.

Format-item is a keyword that tells PRO/DATATRIEVE where to print the following data item:

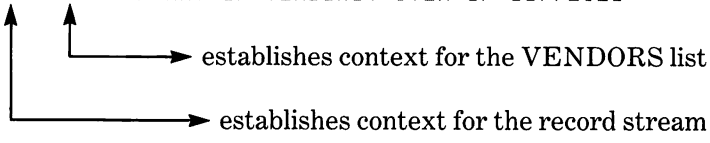
- *COL number* displays the next data item in the column number you specify. The first column in each line is column 1. If you specify a number lower than the current column number, PRO/DATATRIEVE skips to the next line and displays the next data item in the specified column.
- *SKIP [number]* inserts as many blank lines between the current line and the next line as you specify. If you do not specify a number, PRO/DATATRIEVE moves to the beginning of the next line.
- *SPACE [number]* inserts as many horizontal spaces as you specify before printing the next data item. If you do not specify a number, PRO/DATATRIEVE inserts one space before printing the next data item.

List-specification identifies a list field as the source for the information you want to display and has the following format:

```
ALL [print-list-element],... OF list-rse
```

- *List-rse* identifies a list field as the source for print list elements. If you do not specify any print list elements in a list specification, PRO/DATATRIEVE displays all fields within a list field. To see selected fields within the list field, specify the field names in the list specification.
- By nesting a list specification in another list specification, you can display information from a list field within a list field. PRO/DATATRIEVE displays the specified fields of the list field identified in the list RSE for each record being displayed.
- You must include the keyword ALL before the list specification to establish the proper context for PRO/DATATRIEVE:

```
DTR> PRINT ALL ALL COMPANY OF VENDORS, ITEM OF SUPPLIES
```



This PRINT statement displays the COMPANY field in the VENDORS list and then the ITEM field. The list RSE, OF VENDORS, identifies the list that contains the COMPANY field. The domain RSE, OF SUPPLIES, identifies the domain that contains the fields.

See Chapter 4 for more information on displaying list fields.

Rse is a record selection expression that creates a record stream for the source of information to display. When you specify an RSE, PRO/DATATRIEVE displays the specified data items from records in the record stream.

The *ON* clause tells PRO/DATATRIEVE to write the specified display to a file or to your Professional. File-spec creates and names a file to contain the display:

- If you want PRO/DATATRIEVE to prompt you for a file name, use a prompting value expression for the file specification. For example:

```
PRINT CUSTOMERS ON *,"TI: for screen, LP: for printer, or a file name"
```

  TI: is the device name of your Professional video screen. If you respond to the prompt with TI:, PRO/DATATRIEVE prints the display to your Professional video screen. LP: is the device name of your line printer. If you respond to the prompt with LP:, PRO/DATATRIEVE prints the information on the printer.

- If you want the display file to reside in your current directory, you need to specify only a file name. PRO/DATATRIEVE creates the file in your current directory and gives it a .LST file type. Like all Professional file names, a report file name can consist of from one to nine letters or numbers.

- To create a display file in a different directory or on a different device, or to specify a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

Comments

Before you can display information from records in a domain, you must ready the domain.

To print character string literals, enclose the string in double quotation marks. To include a quotation mark in a character string literal, type two quotation marks for every one you want in the display:

```
DTR> PRINT "They said, ""We're going."""
They said, "We're going."
DTR>
```

When you do not include any print list elements in a PRINT statement, PRO/DATATRIEVE displays and formats information as follows:

- Information comes from the selected record (PRINT), the records in the CURRENT collection (PRINT ALL), the records in the record stream formed by a FOR statement that contains the PRINT statement (FOR rse PRINT), or the records in the record stream formed by the RSE in the PRINT statement (PRINT rse).
- The record definition determines the format of information. The display of each record begins in column one. PRO/DATATRIEVE single spaces each line of data, inserting a single blank line between the header and the first line of information.
- The edit string (if specified), the column header, or the length of the longest value stored in a field determines the horizontal spacing of a line. If all the fields of a record will not fit on a single line, PRO/DATATRIEVE displays the remaining fields on the next line.
- The column header for each field is the query header, if specified in the field definition. If no query header exists for a field, PRO/DATATRIEVE uses the field name for the header. If the field name contains a hyphen (or hyphens), PRO/DATATRIEVE places each part of the name on a separate line, discarding the hyphens and centering each part of the name above the column of data. If you specify only a hyphen for the query header, PRO/DATATRIEVE does not print any header.

You can combine the various components of the PRINT statement in many ways to display information:

- PRINT

  Displays the selected record in the CURRENT collection. If the CURRENT collection has no selected record, PRO/DATATRIEVE displays the selected record in the most recently formed named collection. If no selected record exists for any collection, PRO/DATATRIEVE displays the entire CURRENT collection. If no CURRENT collection exists, PRO/DATATRIEVE displays an error message.

- PRINT ALL

  Displays the whole CURRENT collection. If no CURRENT collection exists, PRO/DATATRIEVE displays an error message.

(continued on next page)

- PRINT COMPANY, CONTACT

  Displays the COMPANY and CONTACT fields of the selected record of the CURRENT collection. If the CURRENT collection has no selected record, or if the selected record does not contain the specified fields, PRO/DATATRIEVE displays values from the selected record in the most recently formed collection with COMPANY and CONTACT fields.

- PRINT ALL COMPANY, CONTACT

  Displays the COMPANY and CONTACT fields for all records in the CURRENT collection.

- PRINT ALL COMPANY OF CUSTOMERS WITH STATE EQ "NH"

  Displays the COMPANY fields for all records in the record stream formed by the RSE.

- FOR CUSTOMERS WITH STATE EQ "NH"
  PRINT COMPANY, CONTACT

  Displays the COMPANY and CONTACT fields for all records in the record stream formed by the RSE in the FOR statement.

- PRINT ITEM, ALL COMPANY OF VENDORS OF SUPPLIES

  Displays the ITEM and COMPANY fields from the SUPPLIES domain. The RSE, OF SUPPLIES, identifies the domain and OF VENDORS identifies the list field that contains the COMPANY field.

- PRINT ALL ALL COMPANY OF VENDORS, ITEM OF SUPPLIES

  Displays the COMPANY field from the VENDORS list, then the ITEM field which is not in a list. You must specify ALL ALL when you display a list item first.

When you specify a prompting value expression for a data item, or when you use a VIA modifier to display the table translation for a data item, PRO/DATATRIEVE uses a 10-character edit string. You can override this edit string by including a USING modifier for the data item.

Examples

Form a collection of records from the SUPPLIES domain and select the first record. Use PRINT statements to display the ITEM and ORDER-UNIT fields for the selected record and for the entire collection:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> FIND SUPPLIES
[2 records found]
DTR> SELECT
DTR> PRINT ITEM, ORDER-UNIT

                    ORDER
     ITEM            UNIT

PAPER              REAM

DTR> PRINT ALL ITEM, ORDER-UNIT

                    ORDER
     ITEM            UNIT

PAPER              REAM
PENS               GROSS

DTR>
```

Ready the EMPLOYEES domain and use a PRINT statement in a FOR loop to display the ID, FIRST-NAME, and LAST-NAME fields of the first two employee records:

```
DTR> READY EMPLOYEES
DTR> FOR FIRST 2 EMPLOYEES PRINT ID, FIRST-NAME, LAST-NAME

         FIRST         LAST
 ID       NAME         NAME

00001 GEORGE      BOOLE
00012 CHARLOTTE   SPIVA

DTR>
```

Declare a variable called WHOLE-NAME that formats the FIRST-NAME and LAST-NAME with one space between them. Use the variable in a PRINT statement to display the FIRST-NAME and LAST-NAME fields of the first 2 records

in the EMPLOYEES domain. Then display the fields again, specifying the column header NAME for the WHOLE-NAME variable:

```
DTR> DECLARE WHOLE-NAME COMPUTED BY FIRST-NAME|!" "|LAST-NAME.
DTR> READY EMPLOYEES
DTR> PRINT WHOLE-NAME OF FIRST 2 EMPLOYEES

        WHOLE
        NAME

GEORGE BOOLE
CHARLOTTE SPIVA

DTR> PRINT WHOLE-NAME ("NAME") OF FIRST 2 EMPLOYEES

        NAME

GEORGE BOOLE
CHARLOTTE SPIVA

DTR>
```

Ready the CUSTOMERS domain and use format modifiers and literals to display information from the first two records in mailing list format:

```
DTR> SET NO PROMPT
DTR> READY CUSTOMERS
DTR> FOR FIRST 2 CUSTOMERS
CON>     PRINT SKIP 2,
CON>           COL 10, COMPANY(-),
CON>           COL 10, CONTACT(-),
CON>           COL 10, STREET(-),
CON>           COL 10, CITY|!", "|STATE|", "|ZIP


         BASIC SYSTEMS
         ANNE DUGGAN
         43 HIGHLAND AVE
         BOSTON, MA, 13532


         INFO ANALYSIS
         WILMA SWAYER
         9 REDACTOR
         DUNKIRK, MA,

DTR>
```

Ready the CUSTOMERS domain and print the COMPANY, CONTACT, and PHONE fields on a printer. Use the device name LP: to specify the printer:

```
DTR> READY CUSTOMERS
DTR> PRINT COMPANY, CONTACT, PHONE OF EMPLOYEES ON LP:
DTR>
```

BEGINNING AND ENDING A REPORT

Use the REPORT statement to call the Report Writer and begin a report specification. The REPORT statement identifies the records that contain the information you want in the report and creates a file for the report or displays the report on your Professional. Report Writer statements in the report specification tell the Report Writer how to format the report and what information to include in the report. Use the END-REPORT statement to end the report specification.

Format

```
REPORT [rse] [ON file-spec]

    Report Writer statement
            .
            .
            .

END-REPORT
```

Explanation

Rse is a record selection expression that forms a record stream containing records you want in the report. You can create reports from readied domains and collections.

If you do not include an RSE, the Report Writer creates a report from the CURRENT collection. If no CURRENT collection exists, the Report Writer displays a message and does not create a report.

The *ON* clause tells PRO/DATATRIEVE to write the report to a file or to your Professional. *File-spec* creates and names a file to contain the report:

- If you want PRO/DATATRIEVE to prompt you for a file name, use a prompting value expression for the file specification. For example:

```
REPORT CUSTOMERS ON *,"TI: for screen, LP: for printer, or a file name"
```

TI: is the device name of your Professional video screen. If you respond to the prompt with TI:, PRO/DATATRIEVE prints the report to your Professional video screen. LP: is the device name of your line printer. If you respond to the prompt with LP:, PRO/DATATRIEVE prints the report on the printer.

- If you want the report file to reside in your current directory, you need to specify only a file name. PRO/DATATRIEVE creates the file in your current directory and gives it a .LST file type. Like all Professional file names, a report file name can consist of from one to nine letters or numbers.
- To create a report file in a different directory or on a different device, or to specify a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

When you include an ON clause, the Report Writer inserts a character at the end of the report to create a new page. If you do not include an ON clause, the Report Writer displays the report on your Professional video screen. Thus, if you include an ON clause when displaying a report on your video screen, the last page of the report scrolls off the screen.

Report Writer statements tell the Report Writer what information to include in the report and how to format the report. When you type the REPORT statement in response to the DTR> prompt, the Report Writer prompts with the RW> prompt until you end the report with the END-REPORT statement. If you type the REPORT statement in response to the DFN> prompt while defining a procedure, PRO/DATATRIEVE continues to prompt with the DFN> prompt until you end the procedure definition with the END-PROCEDURE statement.

The END-REPORT statement ends a report specification. After executing the END-REPORT statement, the Report Writer prompts you for any values in the report that you requested in the specification and produces the report.

The remaining sections of this chapter describe how to use the Report Writer statements summarized in Table 11-2.

Table 11-2: Summary of Report Writer Statements

Statement	Function
AT BOTTOM	Specifies the value, position, and format of data elements in the summary lines displayed at the bottom of reports, report pages, and control group sections.
AT TOP	Specifies the value, position, and format of data elements in the header lines displayed at the top of reports, report pages, and control group sections.
PRINT	Controls the content, format, and column headers for the detail lines in a report.
SET COLUMNS-PAGE	Controls the width of a report page.
SET DATE	Specifies a date for the report, if you do not want the current date.
SET LINES-PAGE	Controls the length of a report page.
SET NO DATE	Suppresses the printing of the current date in the upper right-hand corner of report pages.
SET NO NUMBER	Suppresses the printing of page numbers beneath the date on report pages.
SET REPORT-NAME	Specifies a title for the report.

Comments

Before you can create a report, you must ready the domain that contains the data for the report. You can then create a collection of records that contains the particular information you want and report on the collection. You can also identify the information you want with an RSE in the REPORT statement.

If you want to report on sorted records, use the FIND statement to form a sorted CURRENT collection. Then use the REPORT statement to report on the sorted CURRENT collection. PRO/DATATRIEVE uses memory to sort records and to execute Report Writer statements. If you use an RSE with a SORTED BY clause in the REPORT statement, there may not be enough memory available for PRO/DATATRIEVE to both sort and report.

If you type the REPORT statement in response to the DTR> prompt and Report Writer statements in response to the RW> prompt, PRO/DATATRIEVE displays a message and returns you to the DTR> prompt if you make a typing error. You must then type the entire report specification again to correct the error.

To avoid having to type a report specification again, create the report within a procedure. If you make errors in the REPORT statement or in any Report Writer statements, you can simply edit the procedure and correct the errors. See Chapter 17 for information on editing procedures with the PRO/DATATRIEVE Editor.

Examples

Ready the EMPLOYEES domain and use an RSE in the REPORT statement to display a salary report for employees in the astronomy department:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES
DTR> REPORT EMPLOYEES WITH DEPT CONT "ASTR"
RW>  SET REPORT-NAME = "ASTRONOMY SALARIES"
RW>  SET COLUMNS-PAGE = 50
RW>  PRINT LAST-NAME!:", "!FIRST-NAME, STATUS, SALARY
RW>  AT BOTTOM OF REPORT PRINT SKIP, "TOTAL SALARY",
RW>     TOTAL SALARY USING $$$,$$$,$$$
RW>  END-REPORT

                ASTRONOMY SALARIES        30-Jun-83
                                          Page 1


                                STATUS          SALARY

 SPIVA, CHARLOTTE           EXPERIENCED        $75,892
 HARRISON, LYDIA            EXPERIENCED        $40,747
 BOOLE, GERALD              EXPERIENCED        $21,000
 CHONTZ, NATHANIEL          TRAINEE            $24,502
 BOOLE, MARTHA              TRAINEE            $20,000

      TOTAL SALARY                            $182,141

DTR>
```

Define a procedure that creates a report on contacts and the last contact date for the CUSTOMERS domain. Use a prompting value expression in the RSE, so that

the Report Writer prompts for a date to compare with the LAST-CONTACT field. Use a second prompting value expression for the file specification:

```
DTR> DEFINE PROCEDURE CUST-REP
DFN> READY CUSTOMERS
DFN> REPORT CUSTOMERS WITH LAST-CONTACT LE
DFN>     *,"Before which date" ON
DFN>     *,"TI: for a screen display, a file name for a file"
DFN> SET REPORT-NAME = "CUSTOMER CONTACT LIST"
DFN> SET COLUMNS-PAGE = 60
DFN> PRINT COMPANY, CONTACT, PHONE, LAST-CONTACT
DFN> END-REPORT
DFN> END-PROCEDURE
DTR> READY CUSTOMERS
DTR> :CUST-REP
Enter Before which date: 1-1-83
Enter TI: for a screen display, a file name for a file: TI:

                    CUSTOMER CONTACT LIST                27-Jun-83
                                                         Page 1


       COMPANY              CONTACT          PHONE         LAST
         NAME               PERSON           NUMBER      CONTACT

LOGIC SYSTEMS CO.     GEORGE BOOLE     617-555-9981 17-Apr-81
PRODUCTS UNLIMITED    DANA MCMANUS     617-555-3886 13-Apr-82
DTR>
```

If you had specified a file named CONTACTS instead of your screen, PRO/DATATRIEVE would have produced a file named CONTACTS.LST that you could print on a printer.

CONTROLLING REPORT HEADERS

Use the SET REPORT-NAME statement to specify a report title. Use the SET DATE, SET NO DATE, and SET NO NUMBER statements to control the printing of the date and page numbers in the report. If you do not include these statements in a report specification, the Report Writer omits a title and displays the page number and current date in the upper right-hand corner of the report.

Format

```
SET REPORT-NAME = "report-name"

SET DATE = "date"

SET NO DATE

SET NO NUMBER
```

Explanation

Report-name is a title for the report:

- You must enclose the report name in double quotation marks.
- To create a multiline title, enclose text for each line in quotation marks and separate text for each line with a slash:

```
RW> SET REPORT-NAME = "INVENTORY REPORT"/"FOR DEPT 99"
```

- If you want the Report Writer to prompt you for a title when creating the report, specify a prompting value expression instead of a report name:

```
RW> SET REPORT-NAME = *,"title for the report"
```

  When the Report Writer prompts you for the report name, enclose the report name in quotation marks or the Report Writer displays an error message. To specify a two-line title, enclose both portions in quotation marks, separated by a slash, as in the previous example.

Date is a date for the report. You must enclose the date value in double quotation marks. The Report Writer then includes the specified date in the upper right-hand corner of every report page. PRO/DATATRIEVE uses the current date if you do not specify a date.

Comments

To specify a date other than the current date for the report, use the SET DATE statement. To suppress the printing of the date, use the SET NO DATE statement.

To suppress the printing of page numbers, use the SET NO NUMBER statement.

Example

Ready the INVENTORY domain and create a report dated May 31, 1983. Use the SET NO NUMBER statement to suppress page numbering. Include prompting value expressions for a report name and for the records to include in the report. When the Report Writer prompts for the report name, specify a two-line title:

```
DTR> SET NO PROMPT
DTR> READY INVENTORY
DTR> REPORT INVENTORY WITH ON-HAND LT *,"maximum number on hand"
RW>  SET REPORT-NAME = *,"report title"
RW>  SET NO NUMBER
RW>  SET DATE = "5-31-83"
RW>  PRINT ITEM-NAME, ON-HAND
RW>  END-REPORT
Enter REPORT TITLE: "INVENTORY REPORT:"/"BELOW 1000"
Enter maximum number on hand: 1000


                INVENTORY REPORT:
                    BELOW 1000            31-May-831


              ITEM                    ON
              NAME                  HAND

           BALANCES                  900
           GLITCHES                  284

DTR>
```

CONTROLLING REPORT PAGE SIZE

Use the SET COLUMNS-PAGE and SET LINES-PAGE statements to control the width and length of report pages.

Format

```
SET COLUMNS-PAGE = number

SET LINES-PAGE = number
```

Explanation

If you do not include a SET COLUMNS-PAGE statement in a report specification, the Report Writer spaces the fields evenly across an 80-character line. To change the line width, specify a *number* between 1 and 255 or a prompting value expression in the SET COLUMNS-PAGE statement. If you specify a prompting value expression, the Report Writer prompts for the number of columns per page.

If you do not include a SET LINES-PAGE statement in a report specification, report pages have 60 lines per page. To change the page length, specify a *number* between 1 and 32,767 or a prompting value expression in the SET LINES-PAGE statement. If you specify a prompting value expression, the Report Writer prompts for the number of lines per page.

Comments

The least number of lines a report page can have is determined by the number of records, not counting the report header and column headers. If you set the lines per page to 1, the Report Writer prints the report header and one record, then begins a new page.

If a report contains a list field, the Report Writer considers each item in the list as part of one detail line. Long list fields may override the lines per page you specify.

Examples

Ready the EMPLOYEES domain and print names, salaries, and total salaries for employees in the biology department in a 40-column report:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES
DTR> REPORT EMPLOYEES WITH DEPT CONT "BIOL"
RW>  SET COLUMNS-PAGE = 40
RW>  PRINT LAST-NAME!:", "!FIRST-NAME, SALARY
RW>  AT BOTTOM OF REPORT PRINT SKIP 2, "TOTAL SALARY",
RW>     TOTAL SALARY USING $$,$$$,$$$
RW>  END-REPORT

                                27-Jun-83
                                Page 1

                               SALARY

   ROBERTS, DAN                $41,395
   IACOBONE, ANTHONY           $58,462
   DONCHIKOV, BRUNO            $35,952
   PODERESIAN, RANDY           $33,738

        TOTAL SALARY          $169,547
```

Ready the SUPPLIES domain and print the ITEM and VENDORS list field in a 60-column report. Set the lines per page to seven to see that the Report Writer considers a list field as part of one detail line:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> REPORT FIRST 1 SUPPLIES
RW>  SET LINES-PAGE = 7
RW>  SET COLUMNS-PAGE = 60
RW>  PRINT ITEM, VENDORS
RW>  END-REPORT

                                                    30-Jun-83
                                                    Page 1

                                          MIN                    WE
      ITEM          COMPANY     PRICE   ORDER     PERCENT      PAY

PAPER               ABC SUPPLY  $13.45  100          4%     $12.91
                                        500          5%     $12.77
                                        1000         8%     $12.37
                    MORRIS      $12.98  10           2%     $12.72
                                        50           4%     $12.46
                                        100          6%     $12.20
                    XYZ DATA    $11.57  50           2%     $11.33
                                        500          4%     $11.10
                                        1000         7%     $10.76
```

CREATING REPORT DETAIL LINES

Detail lines in a report contain information from individual records. Use the Report Writer PRINT statement to tell the Report Writer what to include in a detail line and how to format the detail line.

Format

```
PRINT print-list-element,...

print-list-element: { data-item [format-modifier] }
                    { format-item                 }
                    { list-specification          }
```

Explanation

Print-list-elements identify the data you want to display and specify how you want the Report Writer to format the data. Use commas to separate print list elements.

Data-item identifies the data you want to display:

- *Field name* displays an elementary field, a group field, or a list field. If you specify a group field, the Report Writer displays all elementary fields contained in the group. If you specify a list field, the Report Writer displays all the fields contained in the list.
- *Variable name* displays the current contents of a variable derived from field values. Usually a variable in a Report Writer PRINT statement is a COMPUTED BY variable that depends on a field value in the print list.
- *Literal* displays a character string or numeric literal. You must enclose a character string literal (such as "plus") in double quotation marks.
- *Arithmetic expression* displays the results of an arithmetic operation. Operands in the expression are usually field values and literal values. See Chapter 2 for information on forming arithmetic expressions.

You can use a *format-modifier* to tell the Report Writer how to display an item, to display the table translation for the item, or to suppress or display a column heading for an item. You can use the same format modifiers as you can use in the PRINT statement you type in response to the DTR> prompt:

("Column header")	To specify a column header
(-)	To suppress a column header
USING edit-string	To specify an edit string
VIA table-name	To print a table translation

Table 11-1 in the section on displaying information lists and describes format modifiers for data items.

Format-item is a keyword that tells the Report Writer where to print the following data item:

- *COL number* displays the next data item in the column number you specify. The first column in each line is column 1. If you specify a number lower than the current column number, the Report Writer skips to the next line and displays the next data item in the specified column.
- *SKIP [number]* inserts as many blank lines between the current line and the next line as you specify. If you do not specify a number, the Report Writer moves to the beginning of the next line.
- *SPACE [number]* inserts as many horizontal spaces as you specify before printing the next data item. If you do not specify a number, the Report Writer inserts one space before printing the next data item.

List-specification identifies a list field as the source for information to include in the detail line and has the following format:

```
ALL [print-list-element],... OF list-rse
```

- *List-rse* identifies a list field as the source for print list elements. If you do not specify any print list elements in a list specification, the Report Writer displays all fields within a list field. To see selected fields within a list field, specify the field names in the list specification.
- By nesting a list specification in another list specification, you can include information from a list field within a list field. The Report Writer prints the specified fields of the list field identified in the list RSE for each record being displayed.

You must specify at least one print list element in a Report Writer PRINT statement. If you do not, the Report Writer prompts for one.

Comments

A report specification can include only one PRINT statement. Unless the report specification contains an AT BOTTOM or AT TOP statement, it must contain a PRINT statement.

The Report Writer does not split fields between lines. If you specify a column value with the COL keyword that makes the column too small for a field, the Report Writer carries the field onto the next line. The column headers of the overflow fields may be lost.

When the data you want in the report is in a list, use a list specification to specify the value, position, and format for fields in the list. To include fields from a list within a list, nest the inner list specification in the outer list specification. For example:

```
RW>  PRINT ALL COMPANY, ALL MIN-ORDER OF DISCOUNT OF VENDORS
```

This PRINT statement displays the COMPANY field in the VENDORS list field, and the MIN-ORDER field in the DISCOUNT list field contained in the VENDORS list field. See Chapter 4 for more information on printing list fields.

Each item of the list takes at least one physical line of printing, but the Report Writer considers the entire list as one detail line.

If you do not include USING clauses to tell the Report Writer how to format items in a detail line, the Report Writer uses the edit string for the item, if specified. If no edit string exists, the Report Writer uses the PICTURE clause for the field or variable. If no PICTURE clause exists, the Report Writer invents an edit string for the data item. The Report Writer prints table translations using a 10-character edit string and you may lose characters. Specify edit strings for data items with the USING format modifier to control the format of data items in detail lines.

Examples

Ready the TRANS domain and create a report of transactions since 6-1-83. Include a VIA modifier and an edit string in the PRINT statement to display the table translation for the COMPANY field:

```
DTR> SET NO PROMPT
DTR> READY TRANS
DTR> REPORT TRANS WITH TRANS-DATE GT "6-1-83"
RW>  SET NO NUMBER
RW>  SET COLUMNS-PAGE = 50
RW>  PRINT COMPANY VIA COMPANIES USING X(20),
RW>        QUANTITY, TOTAL-TRANS
RW>  END-REPORT

                                              27-Jun-83

         COMPANY                              TOTAL
           NAME              QUANTITY         TRANS

   WRITE RIGHT ASSOC.             500          $395.00
   PAPER & PEN, INC.              300          $237.00

DTR>
```

Define a procedure that creates a report on current and projected salaries for a department in the EMPLOYEES domain. Use the value of the SALARY field to figure increases of 10% and 15% and display these fields in the detail lines of the report. Include edit strings to format the new fields and assign columns to all three salary fields:

```
DTR> DEFINE PROCEDURE NEW-SAL
DFN> READY EMPLOYEES
DFN> REPORT EMPLOYEES WITH DEPT CONT
DFN>        *,"which department -- first 3 letters"
DFN> SET NO DATE
DFN> SET NO NUMBER
DFN> SET COLUMNS-PAGE = 60
DFN> SET REPORT-NAME = *,"title for report"
DFN> PRINT LAST-NAME!!", "!FIRST-NAME ("NAME"),
DFN>       SALARY ("CURRENT"/"SALARY") USING $$$$,$$$,
DFN>       SALARY * 1.1 ("10%"/"INCREASE") USING $$$$,$$$,
DFN>       SALARY * 1.15 ("15%"/"INCREASE") USING $$$$,$$$
DFN> AT BOTTOM OF REPORT PRINT SKIP, COL 4, "TOTALS",
DFN>       TOTAL SALARY USING $$$,$$$,$$$,
DFN>       TOTAL SALARY * 1.1 USING $$$,$$$,$$$,
DFN>       TOTAL SALARY * 1.15 USING $$$,$$$,$$$
DFN> END-REPORT
DFN> END-PROCEDURE
```

(continued on next page)

```
DTR> :NEW-SAL
Enter title for report: "ASTRONOMY SALARIES"/"CURRENT AND PROJECTED"
Enter which department -- first 3 letters: AST

                    ASTRONOMY SALARIES
                  CURRENT AND PROJECTED

                                CURRENT        10%          15%
          NAME                  SALARY       INCREASE     INCREASE

 SPIVA, CHARLOTTE              $75,892      $83,481      $87,275
 HARRISON, LYDIA               $40,747      $44,821      $46,859
 BOOLE, GERALD                 $21,000      $23,100      $24,150
 CHONTZ, NATHANIEL             $24,502      $26,952      $28,177
 BOOLE, MARTHA                 $20,000      $22,000      $23,000

   TOTALS                      $182,141     $200,355     $209,462

DTR>
```

CREATING TITLE PAGES, HEADER LINES, AND SUMMARY LINES

Use AT TOP and AT BOTTOM statements to create and format title pages, header information, and summary information in a report. AT TOP and AT BOTTOM statements allow you to print information at the top or bottom of each page, at the beginning or end of a report, or at the top or bottom of control groups. Control groups are groups of records that contain the same value in one or more fields.

Format

```
AT TOP OF REPORT PRINT print-list-element,...

AT BOTTOM OF REPORT PRINT print-list-element,...

AT TOP OF PAGE PRINT print-list-element,...

AT BOTTOM OF PAGE PRINT print-list-element,...

AT TOP OF field-name PRINT print-list-element,...

AT BOTTOM OF field-name PRINT print-list-element,...
```

```
print-list-element:  { data-item   }
                     { format-item }
```

Explanation

Print-list-elements identify the data you want to display and specify how you want the Report Writer to format the data. Separate print list elements with commas.

Data-item identifies the data you want to display:

- *Field name* in an AT TOP statement prints the common field value for the following group of records. In an AT BOTTOM statement, *field name* prints the common field value for the preceding group of records.
- *Variable name* displays the current contents of a variable derived from field values. Usually a variable in a Report Writer AT statement is a COMPUTED BY variable that depends on a field value in the print list.

(continued on next page)

- *Literal* displays a character string or numeric literal. You must enclose a character string literal (such as "EMPLOYEE 15") in double quotation marks.
- *Arithmetic expression* displays the results of an arithmetic operation. Operands in the expression are usually field values and literal values. See Chapter 2 for information on forming arithmetic expressions.
- *Statistical expression* displays the results of a statistical expression. The Report Writer calculates results for all records in the report for statistical expressions in AT TOP statements and in the AT BOTTOM OF REPORT statement, for all records on a page in the AT BOTTOM OF PAGE statement, and for all records in the specified control group in the AT BOTTOM OF field-name statement. See Chapter 2 for information on forming and using statistical expressions.

You can use a *format-modifier* to tell the Report Writer how to display an item, to display the table translation for the item, or to suppress or display a column heading for an item. You can use the same format modifiers as you can use in the PRINT statement you type in response to the DTR> prompt:

("Column header")	To specify a column header
(-)	To suppress a column header
USING edit-string	To specify an edit string
VIA table-name	To print a table translation

Table 11-1 in the section on displaying information lists and describes format modifiers for data items.

Format-item is a keyword that tells the Report Writer where to print the following data item:

- *COL number* displays the next data item in the column number you specify. The first column in each line is column 1. If you specify a number lower than the current column number, the Report Writer skips to the next line and displays the next data item in the specified column.
- *COLUMN-HEADER* overrides the suppression of column headers caused by an AT TOP OF REPORT or AT TOP OF PAGE statement. Use COLUMN-HEADER at the end of an AT TOP statement to display column headers. *Do not use COLUMN-HEADER in an AT BOTTOM statement.*

- *NEW-PAGE* tells the Report Writer to begin a new page before printing the next data item.
- *NEW-SECTION* tells the Report Writer to begin a new page and a new sequence of line numbers, beginning with page one. Use NEW-SECTION *only* in AT TOP OF field-name and AT BOTTOM OF field-name statements.
- *REPORT-HEADER* tells the Report Writer to display the report header, including the report name, date, and page number, at the top of each report page. Use REPORT-HEADER in an AT TOP OF REPORT or an AT TOP OF PAGE statement to override the suppression of the report header those statements cause. *Do not use REPORT-HEADER in AT BOTTOM statements.*
- *SKIP [number]* inserts as many blank lines between the current line and the next line as you specify. If you do not specify a number, the Report Writer moves to the beginning of the next line.
- *SPACE [number]* inserts as many horizontal spaces as you specify before printing the next data item. If you do not specify a number, the Report Writer inserts one space before printing the next data item.

The *AT TOP OF REPORT* statement prints the specified information above the detail lines of a report and suppresses the report header (title, date, and page number) for the first page of the report. The report header is displayed on the following pages of the report, and the page numbers begin with Page 2 on the second physical page of the report.

The *AT BOTTOM OF REPORT* statement prints the specified information at the bottom of the last page of a report.

The *AT TOP OF PAGE* statement prints the specified information at the top of each page of the report, replacing the report header (title, date, and page number).

The *AT BOTTOM OF PAGE* statement prints the specified information at the bottom of each page of the report.

Field-name in an AT TOP or AT BOTTOM statement establishes a pattern of control breaks for the entire report, dividing the report into groups of records with a common value for the specified field:

- The *AT TOP OF field-name* statement prints the specified information at the top of the control group indicated by the field name.
- The *AT BOTTOM OF field-name* statement prints the specified information at the bottom of the control group indicated by the field name. When you specify AT BOTTOM OF field-name PRINT field-name, the Report Writer prints the value in the specified field of the last record in the control group.

If you want to display information about control groups in AT TOP or AT BOTTOM statements, you must sort records for the report using the *field-name* as the sort key.

You must specify at least one print list element in a Report Writer AT TOP or AT BOTTOM statement. If you do not, the Report Writer prompts for one.

Comments

A report specification can contain all six AT statements. Appendix A contains a copy of a report produced by a report specification that uses all the AT statements.

Use AT TOP statements to enhance your reports with title pages, page headings, and group headings:

- Use the AT TOP OF REPORT statement to suppress the default report heading (the title, date, and page number) for the first page of the report. The next page of the report is "Page 2" and includes the report and column headers. You can then design a title page for the report with print list elements, ending the print list with a NEW-PAGE or NEW-SECTION format keyword. If you do not specify NEW-PAGE or NEW-SECTION as the last print list element, the Report Writer prints the specified information at the top of the first report page instead of on a separate page.
- Use the AT TOP OF PAGE statement to suppress the default report and column headings on every page of the report and to print a special heading on each page. If you also want the report and/or column headings on the page, include the REPORT-HEADER or COLUMN-HEADER format item as part of the print list in the statement.

- Use the AT TOP OF field-name statement to print header information for control groups.

Use AT BOTTOM statements to enhance your reports with averages, totals, minimum and maximum values, the number of records, and other summary information about all records in the report, all records on a page, and all records in a control group:

- Use an AT BOTTOM OF REPORT statement to display a conclusion page and summary information about all records in a report. To create a conclusion page, specify NEW-PAGE as the first print list element in the statement.
- Use an AT BOTTOM OF PAGE statement to print information such as "CONFIDENTIAL" or summary information at the bottom of each page of the report.
- Use an AT BOTTOM OF field-name statement to display summary information about groups of records in the report.

You can also use AT BOTTOM statements to create a report consisting only of summary lines for control groups and the entire report. For example:

```
DTR> DEFINE PROCEDURE SALARIES
DFN> READY EMPLOYEES
DFN> FIND EMPLOYEES SORTED BY DEPT
DFN> REPORT CURRENT
DFN> SET COLUMNS-PAGE = 60
DFN> SET REPORT-NAME = "SALARIES"/"BY DEPARTMENT"
DFN> AT BOTTOM OF DEPT PRINT COL 5, DEPT,
DFN>     COL 20, COUNT ("NUMBER"/"EMPLOYEES"),
DFN>     COL 35, TOTAL SALARY ("TOTAL"/"SALARY")
DFN>         USING $$$,$$$,$$$,
DFN>     COL 50, AVERAGE SALARY ("AVERAGE"/"SALARY")
DFN>         USING $$$,$$$,$$$
DFN> AT BOTTOM OF REPORT PRINT SKIP, COL 10,
DFN>     "* * * * * * * * * * * * * * * * * * * * * * * * * * * * *",
DFN>     SKIP 2, COL 5, "ALL DEPARTMENTS:", COL 22, COUNT,
DFN>     COL 35, TOTAL SALARY USING $$$,$$$,$$$,
DFN>     COL 50, AVERAGE SALARY USING $$$,$$$,$$$
DFN> END-REPORT
DFN> END-PROCEDURE
DTR> :SALARIES
```

(continued on next page)

```
                         SALARIES                        28-Jun-83
                      BY DEPARTMENT                      Page 1

                     NUMBER              TOTAL           AVERAGE
      DEPT          EMPLOYEES            SALARY          SALARY

   AGRICULTURE          1                $26,723         $26,723
   ASTRONOMY            5               $182,141         $36,428
   BIOLOGY              4               $169,547         $42,386
   GEOGRAPHY            1                $21,000         $21,000
   LITERATURE           7               $262,266         $37,466
   MATHEMATICS          8               $219,718         $27,464
   PHILOSOPHY           5               $200,614         $40,122
   PSYCHOLOGY           3               $117,554         $39,184

       * * * * * * * * * * * * * * * * * * * * * * * * *

   ALL DEPARTMENTS     34             $1,199,563         $35,281

DTR>
```

Note that this report specification does not have a PRINT statement. The Report Writer PRINT statement prints and formats only detail lines in a report and this report does not contain any detail lines. The AT BOTTOM OF DEPT statement prints and formats each summary line that forms the body of the report and the AT BOTTOM OF REPORT statement prints and formats the aggregate summary line of the report.

If you specify a field in an AT TOP or AT BOTTOM statement not used as a sort key for the records you want in the report, the Report Writer displays an error message and does not create the report. To report on control groups, form a sorted CURRENT collection with the FIND statement, then use the REPORT statement to create a report.

If you specify a field in an AT TOP or AT BOTTOM statement without having sorted the records at all, the Report Writer displays a message and divides the detail lines into control groups anyway, forming a new control group every time the value in the specified field changes.

Examples

Define a procedure to create a salary report for the EMPLOYEES domain. Ready the EMPLOYEES domain and form a collection of records sorted by the LAST-NAME and FIRST-NAME fields. Declare a date variable and assign the current date to it so that you can display the date on the title page, then use SKIP, COL, and NEW-PAGE in an AT TOP OF REPORT statement to create a title page for the report and an AT BOTTOM OF REPORT statement to summarize information in the report:

```
DTR> DEFINE PROCEDURE SAL1
DFN> READY EMPLOYEES
DFN> FIND EMPLOYEES SORTED BY LAST-NAME, FIRST-NAME
DFN> DECLARE DATE-TODAY USAGE IS DATE.
DFN> DATE-TODAY = "TODAY"
DFN> REPORT CURRENT ON *,"Report file spec - TI: for screen"
DFN> SET COLUMNS-PAGE = 70
DFN> AT TOP OF REPORT PRINT SKIP 15, COL 20,
DFN>    "* * * * * * * * * * * * * * * * * * * * *",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 32, "SALARY REPORT", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 30, "FOR ALL EMPLOYEES", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 33, DATE-TODAY, COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "* * * * * * * * * * * * * * * * * * * *",
DFN>    SKIP 5, COL 24, "AN EQUAL OPPORTUNITY EMPLOYER",
DFN>    SKIP 10, COL 28, "COMPANY CONFIDENTIAL",
DFN>    SKIP, COL 28, "******* ************",
DFN>    NEW-PAGE
DFN> PRINT COL 15, LAST-NAME!!", "!FIRST-NAME ("EMPLOYEE NAME"),
DFN>    COL 40, STATUS, COL 55, SALARY
DFN> AT BOTTOM OF REPORT PRINT SKIP 2,
DFN>    COL 15, "* * * * * * * * * * * * * * * * * * * * * * * * * * * * *",
DFN>    SKIP 2, COL 40, "TOTAL SALARIES",
DFN>    TOTAL SALARY USING $$$,$$$,$$$
DFN> END-REPORT
DFN> FINISH EMPLOYEES
DFN> END-PROCEDURE
DTR>
```

The title page produced by this report specification looks like this:

```
* * * * * * * * * * * * * * * * * * * *
*                                       *
*             SALARY REPORT             *
*                                       *
*           FOR ALL EMPLOYEES           *
*                                       *
*              28-Jun-83                *
*                                       *
* * * * * * * * * * * * * * * * * * * * *


      AN EQUAL OPPORTUNITY EMPLOYER




          COMPANY CONFIDENTIAL
          ******* ************
```

The end of the report produced by the SAL1 procedure looks like this:

```
                              .
                              .
                              .
TERRY, CASS                   EXPERIENCED          $29,908
WINSLOW, JOANNE               EXPERIENCED          $34,125
WITTGEN, STAN                 TRAINEE              $25,023

* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                              TOTAL SALARIES   $1,199,563
```

Define a procedure that creates a salary report by department for the EMPLOYEES domain. Ready the EMPLOYEES domain and form a collection of records sorted by the DEPT, LAST-NAME, and FIRST-NAME fields. Include an AT TOP OF PAGE statement to print a formatted header on every page of the report, an AT TOP OF field-name statement to print the department at the top of each group of records, and AT BOTTOM statements to print salary information for each department and for the entire report:

```
DTR> DEFINE PROCEDURE SAL2
DFN> READY EMPLOYEES
DFN> FIND EMPLOYEES SORTED BY DEPT, LAST-NAME, FIRST-NAME
DFN> REPORT ON CURRENT *,"Report file spec - TI: for screen"
DFN> SET COLUMNS-PAGE = 70
DFN> AT TOP OF PAGE PRINT REPORT-HEADER,
DFN>    SKIP, COL 20, "* * * * * * * * * * * * * * * * * * * * *",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 32, "SALARY REPORT:", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 32, "CONFIDENTIAL", COL 56, "*",
DFN>    SKIP, COL 20, "*", COL 56, "*",
DFN>    SKIP, COL 20, "* * * * * * * * * * * * * * * * * * * *",
DFN>    SKIP 2
DFN> AT TOP OF DEPT PRINT SKIP, COL 30, DEPT (-),
DFN>    SKIP, COL 28, "---------------", SKIP
DFN> PRINT COL 15, LAST-NAME!!", "!FIRST-NAME ("EMPLOYEE NAME"),
DFN>    COL 40, STATUS, COL 55, SALARY
DFN> AT BOTTOM OF DEPT PRINT SKIP 2, COL 40,
DFN>        "TOTAL SALARY:", TOTAL SALARY USING $$$,$$$,$$$,
DFN>    SKIP, COL 40,
DFN>        "AVERAGE SALARY:", AVERAGE SALARY USING $$$,$$$,$$$
DFN> AT BOTTOM OF REPORT PRINT SKIP, COL 15,
DFN>    "* * * * * * * * * * * * * * * * * * * * *",
DFN>    SKIP 2, COL 20,  "GRAND TOTALS",
DFN>    SKIP, COL 20, "************",
DFN>    SKIP 2, COL 20,
DFN>        "TOTAL SALARY:", TOTAL SALARY USING $$$,$$$,$$$,
DFN>    SKIP, COL 20,
DFN>        "AVERAGE SALARY:", AVERAGE SALARY USING $$$,$$$,$$$
DFN> END-REPORT
DFN> FINISH EMPLOYEES
DFN> END-PROCEDURE
DTR>
```

The beginning and end of the report produced by the SAL2 procedure look like this:

```
                                                        1-Jul-83
                                                        Page 1

      * * * * * * * * * * * * * * * * * * * *
      *                                       *
      *            SALARY REPORT:             *
      *                                       *
      *            CONFIDENTIAL               *
      *                                       *
      * * * * * * * * * * * * * * * * * * * *


                  AGRICULTURE
                ---------------

SCHWEIK, THOMAS             TRAINEE              $26,723


                            TOTAL SALARY:        $26,723
                            AVERAGE SALARY:      $26,723

                  ASTRONOMY
                ---------------

BOOLE, GERALD               EXPERIENCED          $21,000
BOOLE, MARTHA               TRAINEE              $20,000
CHONTZ, NATHANIEL           TRAINEE              $24,502
HARRISON, LYDIA             EXPERIENCED          $40,747
SPIVA, CHARLOTTE            EXPERIENCED          $75,892


                            TOTAL SALARY:       $182,141
                            AVERAGE SALARY:      $36,428

                            .
                            .
                            .


* * * * * * * * * * * * * * * * * * * * * * * * * * * *

      GRAND TOTALS
      ************

      TOTAL SALARY:                          $1,199,563
      AVERAGE SALARY:                           $35,281
```

Define a procedure to create a summary report on salaries. Form a collection of records sorted by DEPT, then report on the collection. Use AT BOTTOM statements to print the names of departments, the number of employees in each department, and salary information for each department and for all departments. Use (-) to suppress column headers for salary totals, averages, minimums, and maximums. Include an AT BOTTOM OF PAGE statement to print and underline the word "CONFIDENTIAL" at the bottom of each report page:

```
DTR> DEFINE PROCEDURE SALARY-INFO
DFN> READY EMPLOYEES
DFN> FIND EMPLOYEES SORTED BY DEPT
DFN> REPORT ON *,"Report file spec - TI: for screen"
DFN> SET COLUMNS-PAGE = 50
DFN> SET REPORT-NAME = "SALARY SUMMARY"
DFN> AT BOTTOM OF DEPT PRINT COL 10,
DFN>    DEPT|:|" ("|COUNT|:|")", SKIP 2,
DFN>    COL 12, "MAXIMUM SALARY:", COL 30,
DFN>        MAX SALARY (-) :MONEY-EDIT, SKIP,
DFN>    COL 12, "MINIMUM SALARY:", COL 30,
DFN>        MIN SALARY (-) :MONEY-EDIT, SKIP,
DFN>    COL 12, "AVERAGE SALARY:", COL 30,
DFN>        AVERAGE SALARY (-) :MONEY-EDIT, SKIP,
DFN>    COL 12, "TOTAL SALARY:", COL 30,
DFN>        TOTAL SALARY (-) :MONEY-EDIT,
DFN>    SKIP
DFN> AT BOTTOM OF PAGE PRINT SKIP 2, COL 28,
DFN>    "CONFIDENTIAL", SKIP, COL 28,
DFN>    "**********"
DFN> AT BOTTOM OF REPORT PRINT SKIP,
DFN>    COL 10, "* * * * * * * * * * * * * * * * * *", SKIP 2,
DFN>    COL 10, "SUMMARY ("|:|COUNT|:|")", SKIP 2,
DFN>    COL 12, "MAXIMUM SALARY:", COL 30,
DFN>        MAX SALARY (-) :MONEY-EDIT, SKIP,
DFN>    COL 12, "MINIMUM SALARY:", COL 30,
DFN>        MIN SALARY (-) :MONEY-EDIT, SKIP,
DFN>    COL 12, "AVERAGE SALARY:", COL 30,
DFN>        AVERAGE SALARY (-) :MONEY-EDIT, SKIP,
DFN>    COL 12, "TOTAL SALARY:", COL 30,
DFN>        TOTAL SALARY (-) :MONEY-EDIT
DFN> END-REPORT
DFN> END-PROCEDURE
DTR>
```

The beginning and end of the report produced by the SALARY-INFO procedure look like this:

```
          SALARY SUMMARY                1-Jul-83
                                        Page 1

AGRICULTURE (1)

  MAXIMUM SALARY:              $26,723
  MINIMUM SALARY:              $26,723
  AVERAGE SALARY:              $26,723
  TOTAL SALARY:                $26,723

ASTRONOMY (5)

  MAXIMUM SALARY:              $75,892
  MINIMUM SALARY:              $20,000
  AVERAGE SALARY:              $36,428
  TOTAL SALARY:               $182,141

                    .
                    .
                    .

* * * * * * * * * * * * * * * * * * *

SUMMARY (34)

  MAXIMUM SALARY:              $75,892
  MINIMUM SALARY:               $2,000
  AVERAGE SALARY:              $35,281
  TOTAL SALARY:             $1,199,563

                      CONFIDENTIAL
                      ************
```

12

Storing, Modifying, and Deleting Information

Chapter 12

Storing, Modifying, and Deleting Information

This chapter describes the statements you can use to store, modify, and delete information from a PRO/DATATRIEVE database.

You can store, modify, and delete an entire record or only particular fields of a record. Examples in this chapter show how to do many things, including:

- Modifying a field value based on the old field value
- Transferring information from one domain to another
- Using information in one domain to modify another domain
- Verifying information before storing it
- Choosing whether to delete records or not

Chapter 13 describes the statements you can use to control the execution of statements that store, modify, and delete information.

STORING INFORMATION

Use the STORE statement to create a record in a PRO/DATATRIEVE domain and store values in one or more fields of the record.

Format

```
STORE domain-name [USING statement] [VERIFY USING statement]
```

Explanation

Domain-name is the name of the domain you want to contain the new record. If you readied the domain with an alias, you must use the alias as the domain name.

The *USING* keyword introduces a PRO/DATATRIEVE statement that contains values for one or more fields in the new record or prompting value expressions that tell PRO/DATATRIEVE what and how to prompt for values to store in the record's fields. See Chapter 2 for information on prompting value expressions.

If you omit the *USING* clause, PRO/DATATRIEVE prompts for a value for the first elementary field in the record with the following type of prompt:

```
ENTER field-name:
```

Field-name is the name of the first elementary field in the record. After you enter a value and press DO, PRO/DATATRIEVE prompts for a value for the next elementary field in the record by name, and so on. When you have entered values for all elementary fields in the record, PRO/DATATRIEVE writes the record to the data file.

The *VERIFY* keyword introduces a statement you want PRO/DATATRIEVE to execute before storing the new record. Use this clause to validate values to be stored in the record. PRO/DATATRIEVE does not store any data in the record until the statement in the VERIFY USING clause executes successfully.

Statement in both the USING and VERIFY USING clauses can be a single statement or a compound statement, such as BEGIN-END, THEN, or IF-THEN-ELSE.

If the VERIFY USING clause contains an ABORT statement in an IF-THEN-ELSE statement, PRO/DATATRIEVE stops the STORE operation without storing the record and returns to the DTR> prompt when the conditions for the ABORT statement are met.

Comments

You must ready a domain for WRITE access before you can store records in it.

You can extend a STORE statement over more than one line by pressing DO after USING or after VERIFY. If you press DO after typing the domain name, PRO/DATATRIEVE considers the statement finished and prompts for the first field value.

You can omit the keyword USING, but if you do, you must type at least the first element of a statement next to the domain name before pressing DO.

When you use the VERIFY clause to check data for validity by putting an ABORT statement in an IF-THEN-ELSE statement, PRO/DATATRIEVE first checks the value you supply for a field against any VALID IF conditions specified for the field in the record definition. If the value passes this test, PRO/DATATRIEVE then checks it against the conditions specified in the VERIFY clause.

If you always use the same validation criteria for storing data, put those conditions in VALID IF clauses in the record definition. That way, PRO/DATATRIEVE prompts again for a value for the field if the first value you supply does not pass the validation test. When you use an ABORT statement in the VERIFY clause to validate a value, PRO/DATATRIEVE returns you to the DTR> prompt if the value does not pass the validation test.

Because COMPUTED BY values are not stored in records, PRO/DATATRIEVE does not prompt for COMPUTED BY fields.

To store all spaces in an alphanumeric field or all zeros in a numeric field, press the space bar before pressing DO. If you do not type a value in response to a prompt before pressing DO, PRO/DATATRIEVE repeats the prompt.

If you try to store more characters or digits than a field allows, or if the value you specify conflicts with the conditions specified in a VALID IF clause or with the data type of the field, PRO/DATATRIEVE displays a message and prompts again for a value.

Take special care when storing data in primary key fields of indexed files and in other key fields with the NO CHANGE attribute. You cannot modify values in these fields and must erase the record and create a new record if you store incorrect values.

If you try to store a record in an indexed file with field values that duplicate values stored in keys that do not allow duplicates (NO DUP), PRO/DATATRIEVE displays an error message and does not store the record.

If you type in wrong information and want to stop the STORE statement from storing the record, press the EXIT key. PRO/DATATRIEVE stops prompting for values, displays the message "Execution terminated by operator", and displays the DTR> prompt without storing the record.

If you want to store information in only a few of the fields in a record, you can bypass the STORE prompts by assigning values to the fields in the USING clause. For example:

```
DTR> STORE CUSTOMERS USING COMPANY = "HODGE PODGE, INC."
```

This statement stores the value HODGE PODGE, INC. in the COMPANY field of the CUSTOMERS domain. To store values in more than one field, put assignment statements in a BEGIN-END statement, like this:

```
DTR> SET NO PROMPT
DTR> STORE CUSTOMERS USING
CON> BEGIN
CON>     COMPANY = "HODGE PODGE, INC."
CON>     CONTACT = "OPHELIA JOHNSON"
CON> END
DTR>
```

This example stores two values in a new record in the CUSTOMERS domain and suppresses prompts for other field values for the record.

To store more than one record at a time, put the STORE statement in a REPEAT statement:

REPEAT number STORE domain-name

Number specifies the number of records you want to store. If you do not know how many records you want to store, specify a number larger than the estimated number of new records. When you finish storing records, stop the REPEAT statement by pressing the EXIT key. PRO/DATATRIEVE stops prompting for field values and displays the DTR> prompt.

PRO/DATATRIEVE stores a new record in the data file whenever it successfully completes executing a STORE statement. A STORE statement in a loop creates a new record each time PRO/DATATRIEVE executes the loop. If you end the loop by pressing the EXIT key, records already stored remain stored.

You can use a context variable to identify a domain name in a STORE statement by using this format:

```
[context-variable IN] domain-name
```

Context variables are especially useful if you want to refer to values in records by name before the STORE statement finishes executing or if you want to use values stored in one domain to verify, identify, or calculate values for the new record.

By nesting the STORE statement in a FOR loop and using context variables, you can transfer information from one domain to another:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES
DTR> READY WORK-EMP WRITE
DTR> FOR A IN EMPLOYEES
CON>     STORE WORK-EMP USING
CON>         EMPLOYEES-REC = A.EMPLOYEES-REC
DTR>
```

This FOR loop transfers information from elementary fields in the EMPLOYEES domain to elementary fields in the WORK-EMP domain. You must ready EMPLOYEES for READ access and WORK-EMP for WRITE access. In this example, the group fields in the assignment statement (EMPLOYEES-REC and A.EMPLOYEES-REC) contain identical elementary fields, but that need not always be the case. If group fields contain different elementary fields, PRO/DATATRIEVE transfers values only between the elementary fields with identical field names.

You can also use context variables and a STORE statement in a FOR statement to transfer data between elementary fields and to calculate values for elementary fields. The following example identifies a value to store in the ITEM-NUMBER field in the TRANS domain by using values stored in fields in the INVENTORY domain:

```
DTR> SET NO PROMPT
DTR> FOR A IN INVENTORY WITH ITEM-NAME EQ "BAFFLES"
CON>     STORE TRANS USING
CON>         BEGIN
CON>         ITEM-NUMBER = A.ITEM-NUMBER
CON>         UNIT-PRICE = COST
CON>         END
DTR>
```

Because both domains have elementary fields named ITEM-NUMBER, the context variable A identifies A.ITEM-NUMBER as the ITEM-NUMBER field in the INVENTORY domain. You do not need to use the context variable to identify the COST field, though, since its name is unique to the INVENTORY domain.

You can also use a STORE statement in a FOR statement to transfer data between sequential and indexed files. Both domains can share the same record definition, but one domain definition should specify a sequential file, and the other an indexed file. See the section in Chapter 7 for information on defining files and specifying file organization and Chapter 5 for more information on copying data from one domain to another.

You cannot use the USING clause to store values in list fields. To store values in list fields, omit the USING clause and PRO/DATATRIEVE will prompt you for a value for each field in the record.

Examples

Store two records in the INVENTORY domain:

```
DTR> SET NO PROMPT
DTR> READY INVENTORY WRITE
DTR> REPEAT 2 STORE INVENTORY
Enter ITEM_NAME: BALANCES
Enter ITEM_NUMBER: 0881433332
Enter ON_HAND: 400
Enter UNIT_PRICE: 0.34
```

```
Enter ITEM_NAME: THINGS
Enter ITEM_NUMBER: 0082933000
Enter ON_HAND: 500
Enter UNIT_PRICE: 3.35
DTR> PRINT INVENTORY WITH (ITEM-NAME EQ "THINGS") OR
CON>       (ITEM-NAME EQ "BALANCES")

   ITEM          ITEM        ON      UNIT       TOTAL
   NAME        NUMBER      HAND     PRICE      VALUE

BALANCES    0881433332       400      $.34     $136.00
THINGS      0082933000       500     $3.35   $1,675.00

DTR>
```

The domain WORK contains employee names and you want it to also contain an ID field. Define a new domain (TEMP) and a new record with the same fields as WORK-REC with the additional field, ID. Define a file for TEMP and use a context variable and a FOR statement to transfer records from WORK to TEMP. Use a MODIFY statement to add ID values to records in the TEMP domain and finish TEMP and WORK. Edit WORK-REC to add the ID field and define a new file for WORK using the ID field as the indexed key. Ready TEMP for READ access and WORK for WRITE access and repeat the store process you used to transfer records from WORK to TEMP to transfer records from TEMP to WORK. Then delete TEMP and TEMP-REC:

```
DTR> SET NO PROMPT
DTR> DEFINE DOMAIN TEMP USING TEMP-REC ON TEMP.DAT;
DTR> DEFINE RECORD TEMP-REC USING
DFN> 01 WORK-REC.
DFN>    05 ID PIC 9(5).
DFN>    05 EMPLOYEE-NAME QUERY-NAME IS NAME.
DFN>       10 FIRST-NAME PIC X(10)
DFN>          QUERY-NAME IS F-NAME.
DFN>       10 LAST-NAME PIC X(10)
DFN>          QUERY-NAME IS L-NAME.
DFN>    05 NORMAL-NAME COMPUTED BY FIRST-NAME!!" "!LAST-NAME
DFN>       QUERY-NAME IS NORMAL.
DFN>    05 ALPHA-NAME COMPUTED BY LAST-NAME!!", "!FIRST-NAME
DFN>       QUERY-NAME IS ALPHA.
DFN> ;
[Record WORK_REC is 25 bytes long]
DTR> DEFINE FILE FOR TEMP;
DTR> READY WORK
DTR> READY TEMP WRITE
DTR> FOR A IN WORK
CON>    STORE TEMP USING WORK-REC = A.WORK-REC
DTR> FOR TEMP MODIFY ID
Enter ID: 123456
Enter ID: 654321
```

(continued on next page)

```
DTR> FINISH
DTR> EDIT WORK-REC ADVANCED
QED> "05"
           05 EMPLOYEE_NAME QUERY_NAME IS NAME.
QED> I
IN>            05 ID PIC 9(5).
IN>      EXIT
QED> EXIT
DTR> DEFINE FILE FOR WORK KEY = ID;
DTR> READY WORK WRITE
DTR> READY TEMP
DTR> FOR A IN TEMP
CON>     STORE WORK USING WORK-REC = A.WORK-REC
DTR> PRINT FIRST 2 WORK

        FIRST        LAST            NORMAL                    ALPHA
 ID      NAME        NAME             NAME                      NAME

00001 GEORGE      BOOLE       GEORGE BOOLE           BOOLE, GEORGE
00012 CHARLOTTE   SPIVA       CHARLOTTE SPIVA        SPIVA, CHARLOTTE

DTR> DELETE TEMP;
DTR> DELETE TEMP-REC;
DTR> FINISH
DTR>
```

Define a domain named EMP that uses EMPLOYEES-REC and EMP.DAT. Then define a procedure that defines an indexed file for the domain and copies all records from the EMPLOYEES domain to the EMP domain. This creates a duplicate data file that you can modify and delete records from while the original EMPLOYEES data file remains unchanged. Whenever you want a clean data file to work with, simply execute the CLEAN-DATA procedure to create a fresh data file for EMP:

```
DTR> DEFINE DOMAIN EMP USING EMPLOYEES-REC ON EMP.DAT;
DTR> DEFINE PROCEDURE CLEAN-DATA
DFN>    DEFINE FILE FOR EMP KEY = ID
DFN>    READY EMP WRITE
DFN>    READY EMPLOYEES
DFN>    FOR A IN EMPLOYEES
DFN>       STORE EMP USING EMPLOYEES-REC = A.EMPLOYEES-REC
DFN> END-PROCEDURE
DTR> :CLEAN-DATA
DTR> PRINT FIRST 3 EMP

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT          SALARY

00001 EXPERIENCED GEORGE     BOOLE       MATHEMATICS       $2,000
00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY        $75,892
00891 EXPERIENCED FRED       HOWL        MATHEMATICS      $59,594

DTR>
```

Define a procedure to store a record in the TRANS domain and include a VERIFY USING clause in the STORE statement to make sure that the ITEM-NUMBER value being stored matches an ITEM-NUMBER value in the INVENTORY domain. Use context variables to identify the ITEM-NUMBER fields in the two domains and a FOR statement to loop through the INVENTORY domain checking item numbers:

```
DTR> DEFINE PROCEDURE TRANS-WRITE
DFN> READY TRANS WRITE
DFN> READY INVENTORY
DFN> STORE A IN TRANS USING
DFN>     BEGIN
DFN>     ITEM-NUMBER = *,"Item number - 10 digits"
DFN>     TRANS-TYPE = *,"Type of transaction - S or O"
DFN>     TRANS-DATE = *,"Date - dd/mm/yy or TODAY"
DFN>     COMPANY = *,"Three-character company code"
DFN>     QUANTITY = *,"Quantity"
DFN>     END VERIFY USING
DFN>         FOR B IN INVENTORY
DFN>         IF NOT ANY B IN INVENTORY WITH
DFN>             B.ITEM-NUMBER  = A.ITEM-NUMBER THEN
DFN>             ABORT "Bad number - check inventory codes"
DFN> END-PROCEDURE
DTR> :TRANS-WRITE
Enter Item number - 10 digits: 9999999999
Enter Type of transaction - S or O: S
Enter Date - dd/mm/yy or TODAY: TODAY
Enter Three-character company code: PPI
Enter Quantity: 50
ABORT: Bad number - check inventory codes
Execution terminated by "ABORT" statement
DTR> :TRANS-WRITE
Enter Item number - 10 digits: 1182413601
Enter Type of transaction - S or O: S
Enter Date - dd/mm/yy or TODAY: TODAY
Enter Three-character company code: PPI
Enter Quantity: 50
DTR>
```

MODIFYING INFORMATION

Use the MODIFY statement to change the value of one or more fields in a selected record, all records in a collection, or records in a record stream.

Format

```
MODIFY [ALL] field-name,... [USING statement] [OF rse]
```

Explanation

ALL indicates that you want to modify all records in the CURRENT collection.

Field-name specifies the name of a field to modify in the target record. To modify more than one field, separate field names with commas. PRO/DATATRIEVE prompts you to supply a value for each specified field with the following type of prompt:

```
ENTER field-name:
```

If *field-name* is a group field, PRO/DATATRIEVE prompts for each elementary field in the group field.

You can make any of the following responses to the ENTER prompt:

- To leave the current field value unchanged, press the TAB key and then DO or RETURN.
- To change the value of the field, type a new value, then press DO or RETURN. The new value must conform to the field's data type as defined in the record definition or PRO/DATATRIEVE displays an error message and prompts again for a new value.
- To change the value of an alphanumeric field to spaces, press the space bar and press DO or RETURN. To change the value of a numeric field to zero, type a zero (0) or press the space bar and press DO or RETURN.
- To stop the prompting cycle, press the EXIT key. PRO/DATATRIEVE stops the modification process and does not make any changes to the record being modified when you stopped the process. Changes made to other records in the collection or record stream remain, since PRO/DATATRIEVE changed each of those records in the data file as soon as you responded to the last prompt for that record.

If you press DO or RETURN in response to an ENTER prompt for a field value, PRO/DATATRIEVE prompts for the value again.

The *USING* clause contains a PRO/DATATRIEVE statement that assigns values to one or more fields in the target records you want to modify. *Statement* is usually an assignment statement or a BEGIN-END statement that contains assignment statements and other PRO/DATATRIEVE statements, such as PRINT. To see the target record before changing it, put a PRINT statement before the assignment statements in the BEGIN-END block.

When you include a USING clause, PRO/DATATRIEVE does not prompt for field values unless the USING clause contains an assignment statement with a prompting value expression, like this:

```
USING value-expression = *.prompt-name.
```

You cannot use a prompting value expression in a USING clause to assign a value to a group field.

If you do not specify any field names and do not include a *USING* clause, PRO/DATATRIEVE prompts for a value for each elementary field in the record just as it does when you store new records. Because you cannot modify the primary key field of an indexed file, press the TAB key to keep the value stored in the key field. If you try to change the primary key, PRO/DATATRIEVE prompts for values for all fields, and then displays the error message "Attempt to update key field without change attribute" and does not modify the record.

The *OF rse* clause forms a record stream containing the records you want to modify. Use the OF rse clause with care. PRO/DATATRIEVE prompts only once for each field you want to change and assigns the specified value or values to the appropriate field in all records in the record stream.

To modify each record in a record stream, put the MODIFY statement in a FOR statement, like this:

```
FOR rse
    MODIFY field-name,... USING statement
```

Comments

You must ready a domain for WRITE or MODIFY access before you can modify records in it.

Use the MODIFY statement with care, as it changes information in the data file. When PRO/DATATRIEVE executes a MODIFY statement, it immediately changes the information in the data file. PRO/DATATRIEVE catches any errors you might make in entering the statement. Even a correct statement, though, may contain a logical error that causes the wrong records to be changed or fields to be modified incorrectly.

If PRO/DATATRIEVE prompts you for a field you did not expect, stop the modification process by pressing the EXIT key. If you have already typed a value in response to the prompt, press EXIT instead of DO to stop the modification from taking place.

It is good practice to print each record before you modify it. Once you have modified values, you can recover the previous information only by explicitly changing the new values back to the old ones. You may have to use several MODIFY statements to make the necessary changes.

You can extend a MODIFY statement over more than one line by pressing DO after the USING keyword. If you press DO after typing the domain name, PRO/DATATRIEVE considers the statement finished and prompts for the first field value for the first record.

If you like, you can omit the keyword USING. If you do, type at least the first element of *statement* before pressing DO.

You cannot modify the primary key of an indexed file. You also cannot modify a COMPUTED BY field, a REDEFINES field, or an alternate key field that has the NO CHANGE attribute

You should always be aware of record context when using the MODIFY statement. Context is the way PRO/DATATRIEVE recognizes field names and identifies target records on which the MODIFY statement acts. You can establish context in four ways:

- To modify a single record, establish a single record context by forming a collection with a FIND statement and selecting a record with the

SELECT statement. The MODIFY statement operates on the selected record of the CURRENT collection. If no selected record exists for the CURRENT collection, the MODIFY statement operates on the most recently established collection that has a selected record. If there is no selected record for any existing collection, PRO/DATATRIEVE displays an error message.

- To modify all records in the CURRENT collection, form a CURRENT collection with a FIND statement and type MODIFY ALL. Do not specify an *OF rse* clause. MODIFY ALL operates on all records in the CURRENT collection. PRO/DATATRIEVE assigns the same value to each field of every record in the CURRENT collection except when the MODIFY statement includes a USING clause with an arithmetic calculation that uses the old value of the field. In this case, PRO/DATATRIEVE uses the result of the arithmetic calculation to modify the field value in the corresponding record.

  If the USING clause contains prompting value expressions, PRO/DATATRIEVE prompts once for the specifice field values. If the USING clause does not contain any prompting value expressions, PRO/DATATRIEVE does not prompt for field values.

- To modify all records in a record stream, specify an OF rse clause. PRO/DATATRIEVE assigns the same value to each field of every record in the record stream except when the MODIFY statement includes a USING clause that assigns a value to a field with an arithmetic calculation that uses the old value of the field. In this case, PRO/DATATRIEVE uses the value of each calculation to modify the field value in the corresponding record.

- To modify each individual record in a record stream, nest the MODIFY statement in a FOR statement. The MODIFY statement then operates on each individual record in the record stream formed by the RSE in the FOR statement. PRO/DATATRIEVE prompts for field values for each record in a record stream.

Table 12-1 shows the results of using the MODIFY statement in each of these contexts.

Table 12-1: MODIFY Statement Format and Results

MODIFY Format	Result
Modifying the Selected Record:	
MODIFY	Prompts once for each field in the record definition. Changes each field in the selected record using the values you supply to the corresponding prompts.
MODIFY field-name,...	Prompts once for each field specified by *field-name*. Changes each field using the values you supply to the corresponding prompts.
MODIFY USING statement	No prompts unless *statement* contains prompting value expressions. Changes each field in the selected record using the values supplied by the assignment statements in *statement*.
Modifying All Records in the CURRENT Collection:	
MODIFY ALL	Prompts once for each field in the record definition. Changes each field of every record in the CURRENT collection using the values you supply to the corresponding prompts.
MODIFY ALL field-name,...	Prompts once for each field specified by *field-name*. Changes each field of every record in the CURRENT collection using the values you supply to the corresponding prompts.
MODIFY ALL USING statement	No prompts unless *statement* contains prompting value expressions. Changes each field of every record in the CURRENT collection using the values supplied by the assignment statements in *statement*.

(continued)

Table 12-1: MODIFY Statement Format and Results (cont.)

MODIFY Format	Result
Modifying All Records in a Record Stream:	
MODIFY ALL OF rse	Prompts once for each field in the record definition. Changes each field of every record in the record stream to the value you supply to the corresponding prompt. *Use with care.*
MODIFY ALL field-name,... OF rse	Prompts once for each field specified by *field-name*. Changes each field of every record in the record stream to the value you supply to the corresponding prompt. *Use with care.*
MODIFY ALL USING statement OF rse	No prompts unless *statement* contains prompting value expressions. Changes each field of every record in the record stream to the value supplied by the assignment statements in *statement*. *Use with care.*
Modifying Each Record in a Record Stream:	
FOR rse MODIFY	Prompts once for every field in the record definition for each record in the record stream formed by the FOR statement. Changes each field of each record using the value you supply to the corresponding prompt.
FOR rse MODIFY field-name,...	Prompts once for every *field-name* for each record in the record stream formed by the FOR statement. Changes each field of each record using the value you supply to the corresponding prompt.
FOR rse MODIFY USING statement	No prompts unless *statement* contains prompting value expressions. Changes each field of each record in the record stream using the values supplied by the assignment statements in *statement*. The *.prompt prompting value expression prompts once for each record in the record stream. The **.prompt prompting value expression prompts only once, during the first execution of the FOR loop.

A field name in a MODIFY statement must be the name of a group or elementary field that PRO/DATATRIEVE can recognize in the context established for the MODIFY statement. If the field name does not exist in the established context, PRO/DATATRIEVE displays an error message.

If, in response to a prompt, you enter a value that is longer than the length of the field to which it is being assigned, PRO/DATATRIEVE displays an error message and prompts again for the value.

If you specify a USING clause that prompts for a value used in an arithmetic calculation, such as USING SALARY = SALARY + *."RAISE", PRO/DATATRIEVE uses your response to the prompt to calculate the value of the arithmetic expression; the value of that arithmetic expression is then the value stored in the updated field.

If you do not want all the records in the CURRENT collection to have the same new field value, specify an assignment statement in the USING clause that makes the new value of a field in the target record depend on the old value of that field. For example:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES MODIFY
DTR> MODIFY ALL USING SALARY = SALARY * 1.15 OF
CON>     EMPLOYEES WITH STATUS EQ "EXPERIENCED"
DTR>
```

You can use the MODIFY statement to transfer information from one domain to another, from one group field to another, and from one elementary field to another by including one of the following assignment statements in the USING clause:

field-name-1 = field-name-2

group-field-name-1 = group-field-name-2

In both these statements, the field on the left of the assignment statement (1) specifies the target field and the field on the right specifies the source field (2).

Use context variables to identify field names and establish the proper context for the assignment operation. The following procedure modifies two domains and uses context variables to identify fields with identical names in order to correctly calculate values to store in the INVENTORY domain:

```
DTR> SHOW UPDATE-MASTER
PROCEDURE UPDATE_MASTER
READY TRANS WRITE
READY INVENTORY WRITE
FOR A IN TRANS WITH FLAG = 0
   BEGIN
   FOR B IN INVENTORY WITH B.NUM = A.NUM
      BEGIN
      IF TYPE = "S" THEN
      MODIFY USING B.ON_HAND = B.ON_HAND - A.QUANTITY ELSE
      MODIFY USING B.ON_HAND = B.ON_HAND + A.QUANTITY
      END
   MODIFY USING A.FLAG = 1
   END
END_PROCEDURE
DTR>
```

This procedure uses information in the TRANS domain to update the master INVENTORY domain. A flag of zero in a TRANS domain record indicates that the transaction has not been posted, so the procedure sets the FLAG field in each TRANS record to 1 after using the transaction to update the INVENTORY domain.

Transferring data is easiest when the field definitions are exactly the same. If the field definitions differ, truncations may result if the lengths of the fields do not match. You must also anticipate any conflicts between data types. You can, for example, modify an alphanumeric field with a numeric value. Modifying a numeric field with an alphanumeric value that contains nondigit characters, however, causes PRO/DATATRIEVE to display a warning message.

You can create an audit trail of modified records by using a BEGIN-END block in the USING clause. To do this, define and ready a new domain that uses the same record definition as the one whose records you intend to modify. Then put a STORE statement in a BEGIN-END block in the USING clause of the MODIFY statement to store the old record in the new domain before modifying it.

Examples

Ready the EMP domain and form a collection of the first two employees in the astronomy department. Use a MODIFY ALL statement to increase the salaries by ten percent:

```
DTR> READY EMP MODIFY
DTR> FIND FIRST 2 IN EMP WITH DEPT EQ "ASTRONOMY"
[2 records found]
DTR> PRINT ALL

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY       $75,892
78923 EXPERIENCED LYDIA      HARRISON    ASTRONOMY       $40,747

DTR> MODIFY ALL USING SALARY = SALARY * 1.1
DTR> PRINT ALL

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY       $83,481
78923 EXPERIENCED LYDIA      HARRISON    ASTRONOMY       $44,821

DTR>
```

Define a procedure that finds prices in the INVENTORY domain and writes those prices to records without prices in the TRANS domain:

```
DTR> DEFINE PROCEDURE WRITE-PRICE
DFN>    READY TRANS WRITE
DFN>    READY INVENTORY
DFN>    FOR A IN INVENTORY
DFN>    FOR B IN TRANS WITH (B.NUM = A.NUM) AND
DFN>        (TOTAL-TRANS = 0)
DFN>    MODIFY USING B.UNIT-PRICE = A.UNIT-PRICE
DFN> END-PROCEDURE
DTR> PRINT TRANS WITH UNIT-PRICE EQ 0
```

```
   ITEM     TRANS   TRANS    COMPANY                      UNIT       TOTAL
  NUMBER    TYPE     DATE     NAME    FLAG QUANTITY     PRICE       TRANS

1182413601    S     7/01/83   BS        1        50
1183349988    S     6/23/83   WRA       1       500
1183349988    S     6/24/83   PPI       1       300

DTR> :WRITE-PRICE
DTR> PRINT TRANS WITH T-DATE GT "6/15/83"

   ITEM     TRANS   TRANS    COMPANY                      UNIT       TOTAL
  NUMBER    TYPE     DATE     NAME    FLAG QUANTITY     PRICE       TRANS

1182413601    S     7/01/83   BS        1        50       $1.11       $55.50
1183349988    S     6/23/83   WRA       1       500        $.79      $395.00
1183349988    S     6/24/83   PPI       1       300        $.79      $237.00

DTR>
```

DELETING RECORDS FROM AN INDEXED FILE

Use the ERASE statement to permanently delete one or more records from an indexed data file. You cannot use the ERASE statement to delete records from a sequential file.

Format

```
ERASE [ALL [OF rse ]]
```

Explanation

ERASE ALL tells PRO/DATATRIEVE to remove every record in the CURRENT collection from the data file that contains the records.

ERASE ALL OF rse tells PRO/DATATRIEVE to remove every record included in the record stream formed by the record selection expression from the data file that contains the records.

If you put the ERASE statement in a FOR statement, PRO/DATATRIEVE deletes each record specified by the RSE in the FOR statement from the data file.

If you do not put the ERASE statement in a FOR loop and type ERASE and press DO or RETURN, PRO/DATATRIEVE removes the selected record from the data file that contains it. PRO/DATATRIEVE displays the message " No context for ERASE" if you have no selected record.

Comments

You must ready a domain for WRITE access before you can remove records from it.

If you erase a collection or record stream that contains a selected record, the selected record remains available to you, even though the record has been removed from the data file by the ERASE statement. You can display the fields of that selected record, and you can use those field values in value expressions. The selected record remains available until you select another record with the SELECT statement, drop the record with a DROP statement, or until you release control with a RELEASE or FINISH command.

Use the ERASE statement with care. As a precaution, check the CURRENT collection and selected record with the SHOW CURRENT command or with a PRINT statement before typing ERASE to remove a selected record.

Examples

Erase records for all employees in the mathematics department from the EMP domain:

```
DTR> READY EMP WRITE
DTR> FIND EMP WITH DEPT CONT "MATH"
[6 records found]
DTR> PRINT ALL

                      FIRST       LAST
 ID     STATUS         NAME       NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000
00891 EXPERIENCED FRED        HOWL        MATHEMATICS      $59,594
03991 TRAINEE     JAMES       BOOLE       MATHEMATICS      $14,000
73713 TRAINEE     REED        DION        MATHEMATICS      $23,000
78375 EXPERIENCED ALBERT      EINSTEIN    MATHEMATICS      $40,095
83764 EXPERIENCED JIM         MEADER      MATHEMATICS      $41,029

DTR> ERASE ALL
DTR> PRINT ALL
DTR>
```

Define a procedure to find a particular record in the EMP domain, display the record, and erase the record if it is the right one:

```
DTR> DEFINE PROCEDURE DEL-EMP
DFN> READY EMP WRITE
DFN> FIND EMP WITH LAST-NAME EQ *.LAST-NAME AND ID = *.ID
DFN> PRINT ALL
DFN> IF *."Y to erase the record" EQ "Y"
DFN>    THEN ERASE ALL ELSE
DFN>    PRINT "Records not erased"
DFN> END-PROCEDURE
DTR> :DEL-EMP
Enter ID: 00001
Enter LAST_NAME: BOOLE

                      FIRST       LAST
 ID     STATUS         NAME       NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000

Enter Y to erase the record: Y
DTR>
```

DELETING RECORDS FROM A SEQUENTIAL FILE

You cannot delete records from a sequential file because that would change the sequence of the file. You can, however, simulate deleting a record from a sequential file by using the MODIFY statement to change all fields in the record to zero or a space.

Examples

Define a new domain, SEQ-EMP, for employee records and define a sequential file for it. Then use a STORE statement in a FOR loop to copy all records from the EMPLOYEES domain to the new domain. Display the first three records in the SEQ-EMP domain, then define and execute a procedure that deletes information from all fields in the second record. Then display the first three records again to see that the fields are now empty or equal to zero, but that the space occupied by the record remains used:

```
DTR> SET NO PROMPT
DTR> DEFINE DOMAIN SEQ-EMP USING EMPLOYEES-REC ON EMP.SEQ;
DTR> DEFINE FILE FOR SEQ-EMP
DTR> READY SEQ-EMP WRITE
DTR> READY EMPLOYEES
DTR> FOR A IN EMPLOYEES
CON>     STORE SEQ-EMP USING EMPLOYEES-REC = A.EMPLOYEES-REC
DTR> PRINT FIRST 3 SEQ-EMP

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE     BOOLE       MATHEMATICS      $2,000
00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY       $75,892
00891 EXPERIENCED FRED       HOWL        MATHEMATICS     $59,594

DTR> FINISH EMPLOYEES
DTR> DEFINE PROCEDURE DEL-REC
DFN> FOR SEQ-EMP WITH LAST-NAME EQ *.LAST-NAME
DFN>     MODIFY USING
DFN>     BEGIN
DFN>        ID = 0
DFN>        STATUS = " "
DFN>        FIRST-NAME = " "
DFN>        LAST-NAME = " "
DFN>        DEPT = " "
DFN>        SALARY = 0
DFN>     END
DFN> END-PROCEDURE
```

```
DTR> SHOW READY
Ready domains:
        SEQ_EMP: RMS SEQUENTIAL, PROTECTED WRITE
DTR> :DEL-REC
Enter LAST_NAME: SPIVA
DTR> PRINT FIRST 3 SEQ-EMP

                      FIRST        LAST
 ID      STATUS        NAME        NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000
00000
00891 EXPERIENCED FRED        HOWL        MATHEMATICS      $59,594

DTR>
```

13

Controlling PRO/DATATRIEVE Statements and Creating Loops

Chapter 13

Controlling PRO/DATATRIEVE Statements and Creating Loops

Instructions to PRO/DATATRIEVE fall into two categories: commands and statements. You can combine statements with other statements, but you cannot combine commands with other commands or statements.

This chapter describes the statements you can use to create statement blocks and loops and control statement execution:

Creating a Multistatement Block
Creating a FOR Loop
Executing Statements Conditionally
Stopping Statement Execution
Repeating Statements
Joining Statements
Creating a WHILE Loop

You can repeat statements, create a block of statements that executes as a unit, form loops to perform operations on individual records in a record stream, and establish conditions for statement execution.

Among other things, examples in this chapter show how to:

- Modify fields with prompting value expressions in BEGIN-END blocks
- Number records when printing them
- Display a record before modifying it
- Display a record and then ask whether to modify it or not
- Stop a procedure if you try to store invalid information

CREATING A MULTISTATEMENT BLOCK

Use a BEGIN-END statement to group several statements into a single unit called a BEGIN-END block. PRO/DATATRIEVE treats statements in a BEGIN-END block as one statement and executes each statement in sequential order. You can use a BEGIN-END block at any point where you can use a PRO/DATATRIEVE statement. BEGIN-END blocks are particularly useful within FOR, STORE, and REPEAT statements.

Format

```
BEGIN

    statement
        .
        .
        .

END
```

Explanation

Statement is a PRO/DATATRIEVE statement. It can be another BEGIN-END statement. Do not include FIND, SELECT, or DROP statements in BEGIN-END blocks.

A BEGIN-END block within a BEGIN-END block is called a nested block. The only limit to the number of nested BEGIN-END blocks you can form is the amount of PRO/DATATRIEVE memory available at any given time. If you exceed this limit, PRO/DATATRIEVE stops executing statements and displays an error message. You can increase available memory by finishing domains and releasing tables that you no longer need or you can reduce the number of nested BEGIN-END blocks.

You include statements in a BEGIN-END block just as you would type statements in response to the DTR> prompt. When you type the END keyword, PRO/DATATRIEVE executes statements in the block in sequential order.

Comments

Do not call a procedure that contains PRO/DATATRIEVE commands in a BEGIN-END block. You cannot put commands in a BEGIN-END block.

Do not invoke a command file in a BEGIN-END block. PRO/DATATRIEVE executes the command file as soon as you type @file-name and all the statements in the file appear on the screen and become part of the BEGIN-END block just as though you had typed them yourself.

A DECLARE statement in a BEGIN-END block creates a local variable. A local variable exists only within the BEGIN-END block that contains the DECLARE statement, so you cannot refer to a local variable from outside the BEGIN-END block. PRO/DATATRIEVE automatically releases all local variables when the BEGIN-END block that declares the variables finishes executing.

When a BEGIN-END block that defines a local variable is in a FOR loop or REPEAT statement, PRO/DATATRIEVE initializes the variable to zero (numeric) or blank (string or date) each time it executes the BEGIN-END block. See Chapter 15 for information on creating and using local variables.

When you type a BEGIN-END statement in response to the DTR> prompt, PRO/DATATRIEVE prompts with the CON> prompt for the elements needed to complete a statement or to complete the block. After you type END, PRO/DATATRIEVE executes all the statements in the BEGIN-END block and returns to the DTR> prompt. If you are nesting BEGIN-END blocks within BEGIN-END blocks, PRO/DATATRIEVE does not execute any statements in the nested blocks until you type the END that completes the outermost BEGIN-END block.

To stop execution of a BEGIN-END block, press INTERRUPT and then DO. PRO/DATATRIEVE treats a BEGIN-END block as one statement, regardless of the number of statements it contains. If the BEGIN-END block contains BEGIN-END blocks, INTERRUPT/DO stops the first BEGIN-END block.

If a statement in a BEGIN-END block prompts for a value and you press the CANCEL key, PRO/DATATRIEVE prompts again for the value. If you press the INTERRUPT and DO keys in response to the prompt, PRO/DATATRIEVE returns you to the Main Menu. To end the BEGIN-END block and return to the DTR> prompt, press the EXIT key in response to the prompt.

To repeat a sequence of statements, put the statements in a BEGIN-END block and put the BEGIN-END block in a REPEAT statement.

To repeat all statements in a procedure, call the procedure in a BEGIN-END block, and put the BEGIN-END block in a REPEAT statement. If you execute a procedure in a REPEAT statement (REPEAT n :procedure-name), PRO/DATATRIEVE repeats the first statement of the procedure n times and then executes the other statements in the procedure once each.

Use BEGIN-END blocks to include more than one PRO/DATATRIEVE statement in STORE, MODIFY, FOR, and WHILE statements and in the THEN and ELSE clauses of the IF-THEN-ELSE statement.

Examples

Use a BEGIN-END statement to store two new records in the WORK-EMP domain. Since both employees are trainees in the mathematics department and have the same salary, provide these values in assignment statements. Have PRO/DATATRIEVE prompt for values for the ID, FIRST-NAME, and LAST-NAME fields. The SET NO PROMPT suppresses the "Looking for statement" prompts:

```
DTR> READY WORK-EMP WRITE
DTR> SET NO PROMPT
DTR> REPEAT 2 STORE WORK-EMP USING
CON> BEGIN
CON>     ID = *.ID
CON>     STATUS = "TRAINEE"
CON>     LAST-NAME = *.LAST-NAME
CON>     FIRST-NAME = *.FIRST-NAME
CON>     DEPT = "MATHEMATICS"
CON>     SALARY = "20000"
CON> END
Enter ID: 11111
Enter LAST_NAME: SMITH
Enter FIRST_NAME: ALLEN
Enter ID: 11112
Enter LAST_NAME: JONES
Enter FIRST_NAME: AMY
DTR> PRINT WORK-EMP WITH SALARY = 20000 AND
CON> DEPT CONT "MATH"

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

11111 TRAINEE      ALLEN      SMITH      MATHEMATICS      $20,000
11112 TRAINEE      AMY        JONES      MATHEMATICS      $20,000

DTR>
```

Ready the WORK-EMP domain for MODIFY access. Use a BEGIN-END block in the USING clause of a MODIFY statement to print an employee record, modify the salary field, and print the record again:

```
DTR> READY WORK-EMP MODIFY
DTR> SET NO PROMPT
DTR> FOR WORK-EMP WITH SALARY LT 18000
CON> MODIFY USING
CON>    BEGIN
CON>        PRINT LAST-NAME, SALARY
CON>        SALARY = *,"NEW-SALARY"
CON>        PRINT EMPLOYEES-REC, SKIP
CON>    END

   LAST
   NAME     SALARY

BOOLE        $2,000
Enter NEW-SALARY: 3000

                    FIRST        LAST
 ID     STATUS      NAME         NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $3,000

BOOLE       $14,000
Enter NEW-SALARY: 18000
03991 TRAINEE     JAMES       BOOLE       MATHEMATICS      $18,000
```

Define a procedure and call it from a BEGIN-END statement in a REPEAT statement:

```
DTR> SET NO PROMPT
DTR> DEFINE PROCEDURE LOOP-EXAMPLE
DFN>    PRINT "Show how BEGIN-END works with REPEAT"
DFN>    PRINT "and more than one statement"
DFN> END-PROCEDURE
DTR> REPEAT 2 :LOOP-EXAMPLE
Show how BEGIN-END works with REPEAT
Show how BEGIN-END works with REPEAT

and more than one statement

DTR> REPEAT 2
CON> BEGIN
CON>    :LOOP-EXAMPLE
CON> END
Show how BEGIN-END works with REPEAT
and more than one statement
Show how BEGIN-END works with REPEAT
and more than one statement
```

CREATING A FOR LOOP

Use the FOR statement to repeat sequences of PRO/DATATRIEVE statements and to operate on individual records in a record stream. PRO/DATATRIEVE executes each statement or statement block in the FOR statement once for each record in the record stream specified in the record selection expression.

Format

```
FOR rse statement
```

Explanation

Rse is a record selection expression that forms a record stream. The record stream controls the number of times PRO/DATATRIEVE executes the *statement*, and establishes the single record context for the statement.

Statement is a simple or compound statement you want PRO/DATATRIEVE to execute once for each record in the record stream formed by the RSE. You can form compound statements with the BEGIN-END, IF-THEN-ELSE, and THEN statements.

Do not include PRO/DATATRIEVE commands or FIND, SELECT, SORT, DROP or RELEASE statements in a FOR statement.

PRO/DATATRIEVE executes the *statement* once for each record in the record stream, unless you press the EXIT key in response to a prompt to end the FOR loop. If you specify a BEGIN-END statement, PRO/DATATRIEVE executes the statements in the BEGIN-END statement in sequential order for each record in the record stream.

Comments

You must have the domain specified in the RSE readied for READ, WRITE, or MODIFY access.

You can exit from a FOR loop by pressing the EXIT key in response to a prompt for a value or while PRO/DATATRIEVE is executing statements. To return to the Main Menu, press INTERRUPT and then DO in response to a prompt for a value.

You can also force an exit from a FOR loop by using a variable as a counter. To do this:

1. Declare the variable outside the FOR loop.
2. Inside the FOR loop, increase the value of the variable by one for each repetition of the loop.
3. Use an IF-THEN-ELSE statement to specify conditions for the exit based on the value of the variable and include an ABORT statement in the THEN or ELSE clause to tell PRO/DATATRIEVE to exit from the loop when the conditions are true.

Examples

Ready the WORK-EMP domain and increase the salary for all employees in the literature department by 10%:

```
DTR> SET NO PROMPT
DTR> READY WORK-EMP MODIFY
DTR> PRINT SALARY OF WORK-EMP WITH DEPT CONT "LITER"

SALARY
$29,908
$32,918
$32,000
$54,000
$23,908
$55,407
$34,125

DTR> FOR WORK-EMP WITH DEPT CONT "LITER"
CON>    MODIFY USING SALARY = SALARY * 1.1
DTR> PRINT SALARY OF WORK-EMP WITH DEPT CONT "LITER"

SALARY

$32,898
$36,209
$35,200
$59,400
$26,298
$60,947
$37,537

DTR>
```

Use a variable to number employee records and to end a FOR loop after five records have been displayed:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES
DTR> DECLARE COUNTER PIC 9.
DTR> PRINT COUNTER

COUNTER

   0

DTR> FOR EMPLOYEES
CON> BEGIN
CON>     COUNTER = COUNTER + 1
CON>     PRINT COUNTER, EMPLOYEES-REC
CON>     IF COUNTER = 5 THEN ABORT "End of loop"
CON> END

                              FIRST        LAST
COUNTER  ID     STATUS         NAME        NAME          DEPT         SALARY

   1     00001 EXPERIENCED GEORGE      BOOLE       MATHEMATICS       $2,000
   2     00012 EXPERIENCED CHARLOTTE   SPIVA       ASTRONOMY        $75,892
   3     00891 EXPERIENCED FRED        HOWL        MATHEMATICS      $59,594
   4     02943 EXPERIENCED CASS        TERRY       LITERATURE       $29,908
   5     03991 TRAINEE     JAMES       BOOLE       MATHEMATICS      $14,000
ABORT: End of loop
Execution terminated by "ABORT" statement
DTR>
```

EXECUTING STATEMENTS CONDITIONALLY

Use the IF-THEN-ELSE to execute one of two statements depending on the evaluation of a conditional expression.

Format

```
IF boolean-expression THEN statement [ELSE statement]
```

Explanation

Boolean-expression is a Boolean expression that establishes conditions for execution of statements in the THEN or ELSE clause. See Chapter 2 for information on forming Boolean expressions.

THEN introduces the statement you want PRO/DATATRIEVE to execute when the Boolean expression is true.

ELSE introduces the statement you want PRO/DATATRIEVE to execute when the Boolean expression is false.

Statement in either the THEN or ELSE clause can be either a simple or a compound statement. You can execute more than one statement at a time by putting statements in a BEGIN-END, THEN, or another IF-THEN-ELSE statement.

Comments

Although the THEN keyword is optional, use it to clarify the logic of the IF statement. If you include an ELSE clause, you must include the ELSE keyword.

When you specify an ELSE clause, PRO/DATATRIEVE executes the statement in the ELSE clause only when the Boolean expression is false. If you do not specify an ELSE clause and the Boolean expression is false, PRO/DATATRIEVE does not execute the statement in the THEN clause and goes on to the next command or statement.

You can continue an IF-THEN-ELSE statement over multiple lines by pressing DO or RETURN before any statement element that does not complete the statement, except before typing ELSE. PRO/DATATRIEVE will prompt you to continue the statement. If you press DO or RETURN before typing ELSE,

PRO/DATATRIEVE considers the IF statement complete and executes or ignores the THEN clause, depending on the evaluation of the Boolean expression. PRO/DATATRIEVE then tries to execute the ELSE clause as though it were a separate statement and displays an error message because the ELSE clause is not a statement. When you have to break lines in an IF-THEN-ELSE statement, put ELSE at the end of a line rather than at the beginning of the next line.

You can use an IF-THEN-ELSE statement to force an exit from a BEGIN-END block or from a FOR loop. Put an ABORT statement in either the THEN clause or the ELSE clause, and put the IF-THEN-ELSE statement in a BEGIN-END block or FOR loop.

Examples

Print each record in the CUSTOMERS domain and ask whether to modify the LAST-CONTACT field. Use one IF-THEN-ELSE statement to modify the field if the response to a prompt is Y and to skip to the next record if the response is N. Use another IF-THEN-ELSE statement to end the FOR loop if the response to another prompt is N:

```
DTR> SET NO PROMPT
DTR> READY CUSTOMERS MODIFY
DTR> FOR CUSTOMERS WITH LAST-CONTACT LT "TODAY"
CON>    BEGIN
CON>    PRINT COMPANY, CONTACT, PHONE, LAST-CONTACT
CON>    PRINT SKIP
CON>    IF *,"Y to modify Last Contact, N to skip" CONT "Y" THEN
CON>       BEGIN
CON>       MODIFY LAST-CONTACT
CON>       PRINT SKIP
CON>       END ELSE PRINT "No change made to this record", SKIP
CON>    IF *,"Y to continue, N to quit" CONT "N" THEN
CON>       ABORT "End of price changes"
CON>    END

      COMPANY              CONTACT          PHONE        LAST
        NAME               PERSON           NUMBER       CONTACT

BASIC SYSTEMS          ANNE DUGGAN      617-555-8495 18-May-83


Enter Y to modify Last Contact, N to skip: Y
Enter LAST_CONTACT: TODAY


Enter Y to continue, N to quit: Y
INFO ANALYSIS          WILMA SWAYER     617-555-7114  2-Mar-83
```

(continued on next page)

```
Enter Y to modify Last Contact, N to skip: N
No change made to this record

Enter Y to continue, N to quit: N
ABORT: End of price changes
Execution terminated by "ABORT" statement
DTR>
```

Ready the SUPPLIES domain and use nested IF statements to print the COMPANY field for suppliers of paper and of pens:

```
DTR> SET NO PROMPT
DTR> READY SUPPLIES
DTR> FOR SUPPLIES WITH ANY VENDORS
CON> IF ITEM EQ "PAPER" THEN
CON>     PRINT "Paper Suppliers:", ALL COMPANY OF VENDORS ELSE
CON>     IF ITEM EQ "PENS" THEN
CON>        PRINT "Pen Suppliers:", ALL COMPANY OF VENDORS

                  COMPANY

Paper Suppliers: ABC SUPPLY
                 MORRIS
                 XYZ DATA

                 COMPANY

Pen Suppliers: ABC SUPPLY
               MORRIS
               HARRIDON

DTR>
```

STOPPING STATEMENT EXECUTION

Use the ABORT statement to stop a single statement or an entire procedure or command file. Use the SET ABORT and SET NO ABORT commands to indicate whether you want the the ABORT statement to stop a single statement or an entire procedure or command file. SET ABORT tells PRO/DATATRIEVE to stop the procedure or command file

Format

```
SET [NO] ABORT

ABORT value-expression
```

Explanation

The SET ABORT and SET NO ABORT commands determine the effect of the ABORT statement when it executes in a procedure or command file:

- The SET ABORT command tells PRO/DATATRIEVE to stop the procedure or command file that contains the ABORT statement when the ABORT statement executes. With SET ABORT in effect, PRO/DATATRIEVE returns you to the DTR> prompt.
- The SET NO ABORT command tells PRO/DATATRIEVE to stop only the statement in the procedure or command file that contains the ABORT statement. With SET NO ABORT in effect, PRO/DATATRIEVE executes the next command or statement in the procedure or command file.

SET NO ABORT is in effect when you first enter PRO/DATATRIEVE.

Value-expression is usually a character string literal that tells why the statement was halted.

The ABORT statement does not stop a PRO/DATATRIEVE session. Rather, PRO/DATATRIEVE stops executing the statement that contains the ABORT statement and displays a message containing the value specified by *value-expression*.

The ABORT statement affects the outermost statement that contains it. When a statement contains nested FOR loops, you cannot use an ABORT statement to transfer control from an inner loop to an outer loop. Similarly, when a statement contains nested BEGIN-END blocks, you cannot use an ABORT statement to transfer control from an inner block to an outer one.

Comments

The SET ABORT command has no effect on an ABORT statement that is not in a procedure or command file. PRO/DATATRIEVE displays the message specified by the value expression and returns you to the DTR> prompt.

Use an IF-THEN-ELSE statement to establish conditions for stopping a statement. The Boolean expression in the IF clause establishes the conditions that control the ABORT statement. The ABORT statement executes when:

- The Boolean expression is true and the ABORT statement is in the THEN clause.
- The Boolean expression is false and the ABORT statement is in the ELSE clause.

Examples

Declare a variable and assign a value to it. Put an ABORT statement in an IF statement to display a message and return to the DTR> prompt if the variable does not equal ABC.

```
DTR> SET NO PROMPT
DTR> DECLARE X PIC XXX.
DTR> X = "DEF"
DTR> IF X = "ABC" THEN PRINT X ELSE
CON>     ABORT "X does not equal ABC"
ABORT: X does not equal ABC
Execution terminated by "ABORT" statement
DTR>
```

Define a procedure to store a record in the INVENTORY domain. Use an ABORT statement in an IF-THEN statement to stop the store operation if the value of PRODUCT-YEAR is not 83. You do not need to specify SET NO ABORT because the BEGIN-END block that contains the ABORT statement is the last statement in the procedure. When the ABORT statement executes, the procedure ends:

```
DTR> DEFINE PROCEDURE INV-STORE
DFN> READY INVENTORY WRITE
DFN> STORE INVENTORY USING
DFN> BEGIN
DFN>     ITEM-NAME = *.ITEM-NAME
DFN>     PRODUCT-GROUP = *."Product group - 2 digits"
DFN>     IF *."Product year - 2 digits" NE 83 THEN
DFN>        ABORT "Product year must be 83"
DFN>     ASSEMBLY-CODE = *."assembly code - 1 digit"
DFN>     SUP-ASSEMBLY = *."supervisor ID - 5 digits"
DFN>     ON-HAND = *.ON-HAND
DFN>     UNIT-PRICE = *.UNIT-PRICE
DFN> END
DFN> END-PROCEDURE
DTR> :INV-STORE
Enter ITEM_NAME: FINCHES
Enter product group - 2 digits: 23
Enter product year - 2 digits: 84
ABORT: Product year must be 83
Execution terminated by "ABORT" statement
DTR>
```

Define a procedure to write a report on the current collection. Use the ABORT statement to stop the procedure if there is no current collection to report on. Include a SET ABORT statement so that PRO/DATATRIEVE does not execute the report specification:

```
DTR> DEFINE PROCEDURE CUR-REPORT
DFN> SET ABORT
DFN> PRINT "Have you established a current collection"
DFN> IF *."YES or NO" CONT "N" THEN
DFN>     ABORT "Sorry, no collection, no report."
DFN> REPORT
DFN> PRINT EMPLOYEES-REC
DFN> END-REPORT
DFN> END-PROCEDURE
DTR> :CUR-REPORT
Have you established a current collection

Enter YES or NO: N
ABORT: Sorry, no collection, no report.
Execution terminated by "ABORT" statement
DTR>
```

Define a procedure that writes new records to the TRANS domain. Declare a variable for the item number and an ABORT statement to stop the procedure if the value entered for the variable does not match an ITEM-NUMBER in the INVENTORY domain. Include a SET ABORT statement before the ABORT statement so that the procedure stops when the ABORT statement executes. If you do not specify SET ABORT, PRO/DATATRIEVE stops only the IF statement when the ABORT statement executes. The rest of the procedure then executes, creating a transaction record with an incorrect inventory number:

```
DTR> DEFINE PROCEDURE TRANS-WRITE
DFN> READY TRANS WRITE
DFN> READY INVENTORY
DFN> SET ABORT
DFN> DECLARE T-ITEM PIC 9(10),
DFN> T-ITEM = *,"Item number - 10 digits"
DFN> IF NOT ANY INVENTORY WITH ITEM-NUMBER = T-ITEM THEN
DFN>     ABORT "Bad number - check inventory codes"
DFN> STORE TRANS USING
DFN>     BEGIN
DFN>     ITEM-NUMBER = T-ITEM
DFN>     TRANS-TYPE = *,"Type of transaction - S or O"
DFN>     TRANS-DATE = *,"Date - dd/mm/yy or TODAY"
DFN>     COMPANY = *,"Three-character company code"
DFN>     QUANTITY = *,"Quantity"
DFN>     END
DFN> END-PROCEDURE
DTR> :TRANS-WRITE
Enter Item number - 10 digits: 9999999999
ABORT: Bad number - check inventory codes
Execution terminated by "ABORT" statement
DTR>
```

REPEATING STATEMENTS

Use the REPEAT statement to execute a simple or compound statement a specified number of times.

Format

```
REPEAT numeric-value statement
```

Explanation

Numeric-value is a number or a numeric variable that specifies the number of times you want PRO/DATATRIEVE to execute the statement. It must be a positive integer less than or equal to 32,767.

Statement is any simple or compound PRO/DATATRIEVE statement except a FIND, SELECT, DROP, or SORT statement. You cannot repeat a command. If the REPEAT statement calls a procedure (for example, REPEAT n :procedure-name), make sure the procedure does not contain a command or a FIND, SELECT, DROP, or SORT statement as its first element.

Comments

Do not use a FIND, SELECT, DROP, or SORT statement in a REPEAT statement.

PRO/DATATRIEVE executes *statement* the number of times specified by the numeric value. Then PRO/DATATRIEVE executes the command or statement following the REPEAT statement.

If a REPEAT statement calls a procedure, PRO/DATATRIEVE executes only the first statement, either simple or compound, in the procedure the number of times specified in the value expression. Each succeeding statement in the procedure is executed only once.

To end a REPEAT statement, press the EXIT key in response to any prompt within the loop or press INTERRUPT and DO while a statement is executing. If you press INTERRUPT/DO in response to a prompt, PRO/DATATRIEVE returns you to the Main Menu.

To control a REPEAT statement loop, use an IF-THEN-ELSE statement with an ABORT statement in the THEN or ELSE clause. Put the IF statement in a BEGIN-END block in the REPEAT statement.

You can nest REPEAT statements. PRO/DATATRIEVE executes each inner REPEAT statement the specified number of times each time it loops through the outer REPEAT statement.

Examples

Print the string "TEST REPEAT" three times:

```
DTR> REPEAT 3 PRINT "TEST REPEAT"
TEST REPEAT
TEST REPEAT
TEST REPEAT

DTR>
```

Exit from a REPEAT statement by pressing the EXIT key in response to a prompt:

```
DTR> READY CUSTOMERS WRITE
DTR> REPEAT 3 STORE CUSTOMERS
Enter COMPANY_NAME: HODGE-PODGE
Enter CONTACT_PERSON: (EXIT)
Execution terminated by operator
DTR>
```

Show the effect of nesting REPEAT statements in procedures. The procedure NUM1 contains two PRINT statements. The procedure NUM2 contains two REPEAT statements, one nested in the other. The inner REPEAT statement causes PRO/DATATRIEVE to execute the first PRINT statement in NUM1 twice each time PRO/DATATRIEVE loops through the outer REPEAT statement:

```
DTR> SHOW NUM1
PROCEDURE NUM1
PRINT SKIP, "ONE, TWO, THREE"
PRINT "ONE, TWO, THREE, FOUR, FIVE"
END_PROCEDURE
DTR> :NUM1

ONE, TWO, THREE

ONE, TWO, THREE, FOUR, FIVE
```

```
DTR> SHOW NUM2
PROCEDURE NUM2
REPEAT 2
        BEGIN
            REPEAT 2 :NUM1
        END
END_PROCEDURE
DTR> :NUM2

ONE, TWO, THREE

ONE, TWO, THREE
ONE, TWO, THREE, FOUR, FIVE

ONE, TWO, THREE

ONE, TWO, THREE
ONE, TWO, THREE, FOUR, FIVE

DTR>
```

JOINING STATEMENTS

Use the THEN statement to join two or more PRO/DATATRIEVE statements into a compound statement. You can use a compound statement anywhere you can use a single statement.

Format

```
statement { THEN statement } ...
```

Explanation

Statement is a simple or compound statement. PRO/DATATRIEVE executes each statement in a THEN statement in sequential order.

If any statement in a THEN statement contains an error, PRO/DATATRIEVE does not execute any of the statements in the THEN statement.

Do not include a command or call a procedure that contains a command in the THEN statement.

Comments

Use THEN statements rather than BEGIN-END blocks to form short compound statements.

If you use any statement that creates a context in the compound statement, do not use any other statement in that compound statement that refers to or depends on that context information. Be careful using FIND, SELECT, SORT, and DROP in a THEN statement. For example, do not include FIND and SELECT, FIND and SORT, or SELECT and DROP in the same THEN statement.

Example

Ready the EMPLOYEES domain for MODIFY access and use a THEN statement to print and then modify records for trainees in the math department. Press the EXIT key to end the THEN statement after modifying the first record in the record stream:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES MODIFY
DTR> FOR EMPLOYEES WITH STATUS EQ "TRAINEE" AND
CON>     DEPT CONT "MATH"
CON>     PRINT THEN MODIFY STATUS, SALARY

                    FIRST       LAST
 ID      STATUS      NAME       NAME          DEPT        SALARY

03991  TRAINEE     JAMES      BOOLE      MATHEMATICS     $14,000
Enter EMPLOYEE_STATUS: EXPERIENCED
Enter SALARY: 15000
11111  TRAINEE     ALLEN      SMITH      MATHEMATICS     $20,000
Enter EMPLOYEE_STATUS: (EXIT)
Execution terminated by operator
DTR>
```

CREATING A WHILE LOOP

Use the WHILE statement to form loops and control their execution. The WHILE statement executes as long as a specified Boolean expression is true.

Format

```
WHILE boolean-expression statement
```

Explanation

Boolean-expression is a Boolean expression and must conform to the following format:

```
{ variable-name }
{ *.prompt      }   boolean-operator value-expression
```

You cannot use a field name as the value expression in a Boolean expression that is in a WHILE statement, but you can use a prompting value expression. See Chapter 2 for information on forming Boolean expressions and prompting value expressions.

Statement is a statement that you want PRO/DATATRIEVE to execute while the Boolean expression is true. You can execute more than one statement at a time by putting statements in a BEGIN-END, THEN, or another IF-THEN-ELSE statement.

Comments

PRO/DATATRIEVE repeats the statement you specify as long as the Boolean expression evaluates to true. If the relationship between the two values in the Boolean expression never changes, the loop repeats forever until you end it by pressing the INTERRUPT and DO keys.

Example

Define a domain for two-digit integers and their squares, and use a WHILE statement to control the number of records stored in the domain:

```
DTR> SET NO PROMPT
DTR> DEFINE DOMAIN SQUARES USING SQUARES-REC ON SQUARES.DAT;
DTR> DEFINE RECORD SQUARES-REC USING
DFN> 01 SQUARES-REC.
DFN>    03 NUMBER PIC 99
DFN>       EDIT-STRING IS Z9.
DFN>    03 ITS-SQUARE PIC 9(4)
DFN>       EDIT-STRING IS Z(3)9.
DFN> ;
DTR> DEFINE FILE FOR SQUARES KEY = NUMBER;
DTR> READY SQUARES WRITE
DTR> DECLARE N PIC 99.
DTR> WHILE N LE 10
DTR>    STORE SQUARES USING
DTR>       BEGIN
DTR>       NUMBER = N
DTR>       ITS-SQUARE = N
DTR>       N = N + 1
DTR>       END
DTR> PRINT SQUARES

         ITS
NUMBER SQUARE

    0      0
    1      1
    2      4
    3      9
    4     16
    5     25
    6     36
    7     49
    8     64
    9     81
   10    100

DTR>
```

14

Forming and Using Collections

Chapter 14

Forming and Using Collections

This chapter describes the statements you can use to create, sort, and release collections and to drop records from collections:

Creating Collections
Sorting Records in a Collection
Releasing Collections
Selecting a Record from a Collection
Dropping a Record from a Collection

A collection is a group of records gathered together from a domain. You can name a collection, refer to it by name, and display, modify, and delete records from it just as you would records in the domain. You can also refine a collection by dropping records from it.

Unlike temporary record streams formed by including an RSE in a statement, a collection remains available to you until you form another collection with the same name, until you release it, or until you finish the domain from which you created the collection.

This chapter describes how to form and use collections and how to select a record in a collection as the target for a DROP, PRINT, MODIFY, or ERASE statement. Among other things, examples in this chapter show how to:

- Sort records from top to bottom and bottom to top
- Select and display a record, then ask whether to modify it or not
- Refine collections to contain only the records you want

Chapter 4 explains how to use collections and selected records to establish a list field as the target of a PRINT or MODIFY statement.

CREATING COLLECTIONS

Use the FIND statement to establish a collection of records from a data domain, a view domain, or another collection. A FIND statement establishes a collection and names it CURRENT, in addition to any other name you specify.

Format

```
FIND rse
```

Explanation

Rse is a record selection expression that identifies the records you want to include in the collection.

When the FIND statement executes, PRO/DATATRIEVE forms a collection consisting of the records specified in the RSE and displays the message "n records found", where n is the number of records in the collection, unless the FIND statement is inside a procedure. When the FIND statement is in a procedure, PRO/DATATRIEVE does not display this message unless the FIND statement is the last statement in the procedure.

If you specify a collection name in the RSE, the collection has two names: CURRENT and the one you specify. You can refer to the collection by the name you gave it or by the name CURRENT.

When you use a FIND statement to form another collection, the new collection becomes the CURRENT collection and you can refer to the old collection only by the name you gave it. If you did not name the collection, it disappears when you form a new CURRENT collection.

Comments

You must ready a domain before you can form a collection from it.

Do not use the FIND statement in a BEGIN-END, FOR, REPEAT, or THEN statement.

PRO/DATATRIEVE collects records in the order it finds them in the data file. There is no order to the collection unless you include a SORTED BY clause in the record selection expression, or unless the domain uses an indexed file. If the domain uses an indexed file, PRO/DATATRIEVE collects records according to the order of the primary key.

Examples

Form a collection of all employees in the mathematics department from the EMPLOYEES domain. Name the collection MATH-PROFS so that you can refer to it later to identify record context. Use the SHOW COLLECTIONS and SHOW BIG-ONES commands to display information about the collection:

```
DTR> READY EMPLOYEES
DTR> FIND MATH-PROFS IN EMPLOYEES WITH DEPT CONT "MATH"
[6 records found]
DTR> SHOW COLLECTIONS
Collections:
        MATH_PROFS (also CURRENT)
DTR> SHOW MATH-PROFS
Collection MATH_PROFS
        Domain: EMPLOYEES
        Number of records: 6
        No selected record
DTR>
```

Form a collection of the 10 most highly-paid employees, sorted by descending price. Do not name this collection and use the SHOW COLLECTIONS command to see that the collection's only name is CURRENT:

```
DTR> FIND FIRST 10 EMPLOYEES SORTED BY DESC SALARY
[10 records found]
DTR> SHOW COLLECTIONS
Collections:
        CURRENT
        MATH-PROFS
DTR>
```

To name the CURRENT collection, type a new FIND statement that names the collection HIGH-ROLLERS and specifies the CURRENT collection as the record source. This collection is identical to the last CURRENT collection except that it has been named, which means that it will not disappear when you form a new CURRENT collection:

```
DTR> FIND HIGH-ROLLERS IN CURRENT
[10 records found]
DTR> SHOW COLLECTIONS
Collections:
        HIGH_ROLLERS (also CURRENT)
        MATH_PROFS
DTR>
```

SORTING RECORDS IN A COLLECTION

Use the SORT statement to change the way records in an established collection are sorted. You can use the SORT statement only to sort collections already established with a FIND statement.

Format

```
SORT [collection-name] BY [sort-order] field-name,...

sort-order: { ASCENDING  }
            { DESCENDING }
```

Explanation

Collection-name is the optional name of the collection you want to sort. If you omit the collection name, PRO/DATATRIEVE sorts the CURRENT collection.

Field-name is the name of a field you want PRO/DATATRIEVE to use as a sort key. You can specify any number of sort keys by separating field names with commas.

Sort-order is a keyword that specifies the way in which you want records sorted:

- *ASCENDING* tells PRO/DATATRIEVE to put the record with the lowest value in the specified sort field first in the collection and the one with the highest value last.
- *DESCENDING* tells PRO/DATATRIEVE to put the record with the highest value in the specified sort field first in the collection and the one with the lowest value last.

If you do not specify a sort order, PRO/DATATRIEVE sorts the collection using the specified sort fields in ASCENDING order. Appendix C shows the order PRO/DATATRIEVE uses to sort printing characters.

Comments

To save typing time, you can omit the BY keyword from the SORT statement and you can abbreviate ASCENDING to ASC and DESCENDING to DESC.

Do not include a SORT statement in a FOR, WHILE, or REPEAT statement.

PRO/DATATRIEVE sorts numeric fields by number and character fields according to the sort order and fields specified as follows:

- If you specify more than one sort field, PRO/DATATRIEVE uses the first field name as the major sort key and each successive field name as an increasingly minor key.
- If you do not specify DESCENDING or ASCENDING for the first sort field, PRO/DATATRIEVE sorts the collection in the ascending order for that field.
- If you do not specify a specific order for the second or subsequent sort fields, PRO/DATATRIEVE uses the sort order implied or specified for the preceding sort key. Thus, if you specified DESCENDING for the first sort field, PRO/DATATRIEVE sorts all subsequent keys in descending order. To change the sort order for a specific field, specify either ASCENDING or DESCENDING before the field name.

When PRO/DATATRIEVE executes a SORT statement, the selected record for the collection, if one exists, is released.

You cannot use the SORT statement to sort record streams. To sort record streams, include a SORTED BY clause in the record selection expression that creates the record stream.

To sort a collection when you form it, include a SORTED BY clause in the RSE in the FIND statement that forms the collection. Note, though, that for particularly large collections, this may use more memory than your system has available. In this case, form the collection and use the SORT statement to sort records in the collection.

Examples

Form a named collection of CUSTOMERS with LAST-CONTACT prior to January 1, 1983. Sort the collection by descending date and print the collection to see which customer you contacted last:

```
DTR> READY CUSTOMERS
DTR> FIND CONTACT IN CUSTOMERS WITH LAST-CONTACT LT "1/1/83"
[3 records found]
DTR> SORT CONTACT BY DESC LAST-CONTACT
DTR> PRINT COMPANY, CONTACT-PERSON, PHONE, LAST-CONTACT OF CURRENT

      COMPANY              CONTACT          PHONE        LAST
        NAME               PERSON           NUMBER      CONTACT

KEY SPECIALISTS        DANE REED        603-555-0570 11-Dec-82
PRODUCTS UNLIMITED     DANA MCMANUS     603-555-3886 13-Apr-82
LOGIC SYSTEMS CO.      GEORGE BOOLE     617-555-9981 17-Apr-81

DTR>
```

Form an unnamed collection of employees in the astronomy department and sort the collection by LAST-NAME and FIRST-NAME. Display the ID numbers and names of employees included in the collection:

```
DTR> READY EMPLOYEES
DTR> FIND EMPLOYEES WITH DEPT CONT "MATH"
[6 records found]
DTR> SORT BY LAST-NAME, FIRST-NAME
DTR> PRINT ALL ID, LAST-NAME, FIRST-NAME OF CURRENT

          LAST       FIRST
  ID      NAME        NAME

00001 BOOLE       GEORGE
03991 BOOLE       JAMES
73713 DION        REED
78375 EINSTEIN    ALBERT
00891 HOWL        FRED
83764 MEADER      JIM

DTR>
```

RELEASING COLLECTIONS

Use the RELEASE command to end your control over one or more collections. This frees the computer memory they occupy.

Format

```
RELEASE collection-name,...
```

Explanation

Collection-name is the name of the collection you want to release. To release more than one collection, separate collection names with commas.

Comments

You can also use the RELEASE command to release tables and variables. Simply list them as you would multiple collection names.

When the RELEASE command executes, PRO/DATATRIEVE releases the specified collections, tables, and variables by removing them from memory. The effect is very much like the FINISH command.

The domain from which the released collection was formed remains ready. You must use the FINISH command to remove readied domains from memory.

When you specify more than one collection, table, or variable in the RELEASE command, PRO/DATATRIEVE releases the items in left-to-right order. When PRO/DATATRIEVE cannot release an item, the RELEASE command stops executing. PRO/DATATRIEVE displays a message indicating which item it could not release and does not release any items that follow.

Before you use the RELEASE command, use the SHOW COLLECTIONS command to see what collections you have established and the order in which you created them.

When you have two or more existing collections and release the CURRENT one, the collection you formed most recently becomes the new CURRENT collection.

You do not have to use the RELEASE command in the following cases:

- When a FIND statement forms a collection, PRO/DATATRIEVE releases any existing collection with the same name. If the CURRENT collection has no other name, a new collection formed by a FIND statement releases and replaces the previous CURRENT collection.
- When you use a FINISH command, PRO/DATATRIEVE releases all collections associated with the domains specified in the FINISH command.
- When you end a PRO/DATATRIEVE session, PRO/DATATRIEVE automatically releases all collections.

Example

Release one of two named collections, then release the other:

```
DTR> SET NO PROMPT
DTR> READY EMPLOYEES
DTR> FIND MATH-PROFS IN EMPLOYEES WITH DEPT CONT "MATH"
[6 records found]
DTR> FIND ASTRONOMERS IN EMPLOYEES WITH DEPT CONT "ASTR"
[5 records found]
DTR> SHOW COLLECTIONS
Collections:
        ASTRONOMERS (also CURRENT)
        MATH-PROFS
DTR> RELEASE ASTRONOMERS
DTR> SHOW COLLECTIONS
Collections:
        MATH-PROFS (also CURRENT)
DTR> RELEASE MATH-PROFS
DTR> SHOW COLLECTIONS
No established collections
DTR>
```

SELECTING A RECORD FROM A COLLECTION

Use the SELECT statement to establish a selected record for a collection. One of the easiest ways to modify or erase one or a few records is to form a collection that contains the record(s) you want to change. Then you can use the SELECT statement to choose a record as the target of a MODIFY or ERASE statement.

Format

```
SELECT [record-specification] [collection-name]

record-specification: { FIRST
                      { NEXT
                      { LAST
                      { value
```

Explanation

If you type SELECT and press DO or RETURN, PRO/DATATRIEVE selects the next record in the CURRENT collection. If you have not yet selected a record, PRO/DATATRIEVE selects the first record in the CURRENT collection.

Record-specification is a keyword or numerical value that specifically identifies the record to select:

- *FIRST* selects the first record in the target collection. If you have not yet selected a record, SELECT FIRST is the same as SELECT.
- *NEXT* selects the next record in the target collection. SELECT NEXT is the same as typing SELECT and pressing DO or RETURN.
- *LAST* selects the last record in the target collection.
- *Value* selects the record in the numerical position specified by the value. SELECT 5, for example, selects the fifth record in the target collection, not five records in the collection. If you specify an expression for the value, the expression must evaluate to a positive number. If you specify a value greater than the number of records in the collection, PRO/DATATRIEVE displays an error message and does not select a new record.

Collection-name is the name of the collection that contains the record you want to select. If you do not specify a collection, PRO/DATATRIEVE selects the specified record from the CURRENT collection.

If you have already selected a record in the collection, SELECT NEXT selects the record following the previously selected record. When the selected record is the last record in the collection and you type SELECT, PRO/DATATRIEVE displays an error message and keeps the last record in the collection as the selected record.

Comments

You can select records only from collections. You must establish the target collection with a FIND statement before you can use the SELECT statement.

A collection can have only one selected record at a time.

You can select a record in a collection other than the CURRENT collection by specifying the collection name. If you select a record from the CURRENT collection and then select a record from another collection, the CURRENT collection and its selected record remain unchanged.

PRO/DATATRIEVE displays an error message if you try to select a record from a collection that does not exist or from a collection that contains no records.

You can select a record that you have previously dropped from the collection, but you cannot retrieve any data from the record that occupied that position. To use the dropped record, you must form a new collection and then select the record.

Do not put a SELECT statement in a BEGIN-END, THEN, FOR, REPEAT, or WHILE statement. If you want to perform the same set of statements on each record in a collection, use the following sequence of statements to loop through the collection:

```
FOR collection-name
BEGIN
    statement
        .
        .
        .
END
```

To show information about a collection, use the SHOW CURRENT or SHOW collection-name command.

When you use a DROP, ERASE, MODIFY, or PRINT statement to operate on the selected record, PRO/DATATRIEVE uses the "nearest" selected record as the target record.

The "nearness" of selected records corresponds to the order in which you created collections, not the order in which you selected records. If the CURRENT collection has a selected record, that record is the "nearest" selected record. If the CURRENT collection does not have a selected record, the next most recently created collection with a selected record is the "nearest" selected record, and so on.

Use the SHOW COLLECTIONS command to see the order in which you created collections. The most recently created collection (the CURRENT) is always at the top of the list, while the "oldest" collection is always at the bottom of the list.

To see the "nearest" selected record, type PRINT and press DO or RETURN. If the CURRENT collection has no selected record, PRO/DATATRIEVE displays the selected record from the most recently established named collection that has a selected record. If no collection has a selected record, PRO/DATATRIEVE displays a message and the entire CURRENT collection.

Having a selected record also allows you to retrieve values from the fields of a selected record without specifying a target record stream. When you use a field name by itself, PRO/DATATRIEVE retrieves the field value from the "nearest" selected record with a field of that name.

To display all fields of the selected record in a named collection that is not the CURRENT collection, you must qualify the top-level field name in the PRINT statement.

- If you created the CURRENT collection and the target collection from the same domain, use the target collection name as the qualifier. Type PRINT MATH-PROFS.EMPLOYEES-REC, for example, to display the selected record in a collection of EMPLOYEES called MATH-PROFS.

(continued on next page)

- If you created the CURRENT collection and the target collection from different domains, you do not have to qualify the top-level field names unless they are the same. If the top-level field names are the same, use the domain name from which you created the target collection to qualify the top-level field name. For example, the EMPLOYEES domain and the WORK-EMP domain use the record definition EMPLOYEES-REC. If you form collections from both domains, you can differentiate between the two selected records by specifying WORK-EMP.EMPLOYEES-REC and EMPLOYEES.EMPLOYEES-REC.

To refer to a field name in a selected record in a named collection that is not the CURRENT collection, use the name of the collection containing the target record to qualify the field name.

Use context variables to distinguish between two or more selected records referred to in one PRO/DATATRIEVE statement. Context variables are particularly useful if you created the collections from the same domain or if the field names of the selected records are identical.

See Chapter 3 for more information on establishing record context.

Examples

Form an unnamed CURRENT collection from the EMPLOYEES domain. Select the last record in the collection and use PRINT statements to display the whole record and the FIRST-NAME field:

```
DTR> READY EMPLOYEES
DTR> FIND FIRST 8 EMPLOYEES
[8 records found]
DTR> SELECT LAST
DTR> PRINT

                    FIRST       LAST
 ID      STATUS      NAME       NAME          DEPT        SALARY

12643 TRAINEE      JEFF       TASHKENT   LITERATURE      $14,000

DTR> PRINT FIRST-NAME

  FIRST
   NAME

JEFF

DTR>
```

Form a collection named MATH-PROFS from the EMPLOYEES domain and select the third record in the collection. Display the selected record with a PRINT statement, then form a new collection named ASTRONOMERS. Select the third record from this collection and display it. Then select the second record in the MATH-PROFS collection and display it. Use the SHOW COLLECTIONS command to see the order of your collections and the SHOW collection-name command to see information about the position of the selected record:

```
DTR> READY EMPLOYEES
DTR> FIND MATH-PROFS IN EMPLOYEES WITH DEPT CONT "MATH"
[6 records found]
DTR> SELECT 3
DTR> PRINT

                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT         SALARY

03991 TRAINEE      JAMES       BOOLE       MATHEMATICS      $14,000

DTR> FIND ASTRONOMERS IN EMPLOYEES WITH DEPT CONT "ASTR"
[5 records found]
DTR> SELECT 3
DTR> PRINT

                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT         SALARY

83771 EXPERIENCED GERALD      BOOLE       ASTRONOMY        $21,000

DTR> SHOW COLLECTIONS
Collections:
        ASTRONOMERS (also CURRENT)
        MATH_PROFS
DTR> SHOW ASTRONOMERS
Collection ASTRONOMERS
        Domain: EMPLOYEES
        Number of records: 5
        Selected record: 3
DTR> SHOW MATH-PROFS
Collection MATH_PROFS
        Domain: EMPLOYEES
        Number of records: 6
        Selected record: 3
DTR> SELECT 2 MATH-PROFS
DTR> PRINT

                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT         SALARY

83771 EXPERIENCED GERALD      BOOLE       ASTRONOMY        $21,000
```

(continued on next page)

```
DTR> PRINT MATH-PROFS.EMPLOYEES-REC

                    FIRST       LAST
 ID      STATUS      NAME       NAME          DEPT        SALARY

00891 EXPERIENCED FRED        HOWL        MATHEMATICS      $59,594

DTR> SHOW MATH-PROFS
Collection MATH_PROFS
        Domain: EMPLOYEES
        Number of records: 6
        Selected record: 2
DTR>
```

Do not release any previously formed collections. Form a collection named BIOLOGISTS from the EMPLOYEES domain and use the SHOW CURRENT command to see that the BIOLOGISTS collection has no selected record. Use the SHOW COLLECTIONS command to see the names of established collections and a PRINT statement to display the "nearest" selected record. Release all collections and type SELECT to see the message PRO/DATATRIEVE displays. Type PRINT to see that PRO/DATATRIEVE looks first for a selected record to display, and then for the CURRENT collection:

```
DTR> FIND BIOLOGISTS IN EMPLOYEES WITH DEPT CONT "BIO"
[4 records found]
DTR> SHOW CURRENT
Collection CURRENT
        Domain: EMPLOYEES
        Number of records: 4
        No selected record
DTR> SHOW COLLECTIONS
Collections:
        BIOLOGISTS (also CURRENT)
        ASTRONOMERS
        MATH_PROFS
DTR> PRINT

                    FIRST       LAST
 ID      STATUS      NAME       NAME          DEPT        SALARY

83771 EXPERIENCED GERALD      BOOLE       ASTRONOMY       $21,000

DTR> RELEASE BIOLOGISTS, ASTRONOMERS, MATH-PROFS
DTR> SELECT
No collection for select
DTR> PRINT
No record selected, printing whole collection
A current collection has not been established
DTR>
```

Form a collection of trainees in the math department. Display each record, and modify the STATUS field based on the response to a prompting value expression:

```
DTR> SET NO PROMPT
DTR> READY WORK-EMP MODIFY
DTR> FIND TRAINEES IN WORK-EMP WITH STATUS CONT "TRAIN" AND
CON>         DEPT CONT "MATH"
[2 records found]
DTR> FOR TRAINEES
CON> BEGIN
CON>     PRINT EMPLOYEES-REC
CON>     IF *."Y to modify, N to keep current status" = "Y" THEN
CON>         MODIFY STATUS
CON>     PRINT SKIP
CON> END

                    FIRST       LAST
 ID      STATUS      NAME       NAME          DEPT        SALARY

03991 TRAINEE     JAMES      BOOLE      MATHEMATICS      $14,000
Enter Y to modify, N to keep current status: Y
Enter EMPLOYEE_STATUS: EXPERIENCED


73713 TRAINEE     REED       DION       MATHEMATICS      $23,000
Enter Y to modify, N to keep current status: N


DTR>
```

DROPPING A RECORD FROM A COLLECTION

Use the DROP statement to remove records from a collection. When PRO/DATATRIEVE removes records from a collection, it does not delete the record from the data file.

Format

```
DROP [collection-name]
```

Explanation

Collection-name names the collection with the selected record you want to drop. If the specified collection has no selected record, or if the selected record has been previously erased or dropped, PRO/DATATRIEVE displays an error message and does not execute the DROP statement.

If you do not specify a *collection-name*, PRO/DATATRIEVE drops the selected record in the nearest single record context:

- If the CURRENT collection has a selected record, PRO/DATATRIEVE drops that record.
- If the CURRENT collection does not have a selected record, PRO/DATATRIEVE drops the selected record from the most recently formed named collection.
- If no collection has a selected record, or if the selected record in the most recently formed collection has been erased or dropped, PRO/DATATRIEVE displays an error message and does not drop any record.

Comments

Use the DROP statement to refine a collection until it contains exactly the records you want. A dropped record is *not* erased from the data file. You can retrieve a dropped record by forming a record stream or another collection that contains it.

You can use the DROP statement only to remove records from a collection.

Before using the DROP statement to drop the selected record in the nearest single record context, it is a good idea to display that record by typing PRINT and pressing DO or RETURN. See the section on selecting records for more information on single record context and the selected record.

To drop more than one record from a collection, put the DROP statement in a FOR loop. The RSE in the FOR statement identifies the records you want to drop.

Examples

Form an unnamed collection from the EMPLOYEES domain, then select and drop the first record from the collection. Try to select that record again to see the message PRO/DATATRIEVE displays. Form the same collection again and display the first record to see that PRO/DATATRIEVE has not deleted the record from the data file:

```
DTR> READY EMPLOYEES
DTR> FIND FIRST 3 EMPLOYEES
[3 records found]
DTR> SELECT
DTR> PRINT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE     BOOLE       MATHEMATICS      $2,000

DTR> DROP
DTR> PRINT
No record selected, printing whole collection

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY       $75,892
00891 EXPERIENCED FRED       HOWL        MATHEMATICS     $59,594

DTR> SELECT FIRST
Record has been dropped from collection
Execution failed
DTR> FIND FIRST 3 EMPLOYEES
[3 records found]
DTR> PRINT FIRST 1 CURRENT

                    FIRST       LAST
 ID     STATUS      NAME        NAME          DEPT        SALARY

00001 EXPERIENCED GEORGE     BOOLE       MATHEMATICS      $2,000

DTR>
```

Ready the WORK-EMP domain and form a series of collections. Use the SELECT statement to select the first record in each collection. Use DROP and PRINT statements to see how the DROP statement operates in a single record context:

```
DTR> READY WORK-EMP
DTR> FIND MATH IN EMPLOYEES WITH DEPT CONT "MATH"
[6 records found]
DTR> SELECT
DTR> FIND BIOS IN EMPLOYEES WITH DEPT CONT "BIOL"
[4 records found]
DTR> SELECT
DTR> FIND PHILOS IN EMPLOYEES WITH DEPT CONT "PHILOS"
[5 records found]
DTR> SELECT
DTR> PRINT

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

34456 TRAINEE      HANK        MORRISON    PHILOSOPHY      $30,000

DTR> DROP
DTR> PRINT

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

90342 EXPERIENCED BRUNO       DONCHIKOV   BIOLOGY         $35,952

DTR> SHOW PHILOS
Collection PHILOS
        Domain: EMPLOYEES
        Number of records: 5
        Selected record: 1
DTR> DROP PHILOS
No collection with selected record for DROP
DTR> PRINT

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

90342 EXPERIENCED BRUNO       DONCHIKOV   BIOLOGY         $35,952

DTR> DROP
DTR> PRINT

                    FIRST       LAST
 ID     STATUS       NAME       NAME          DEPT        SALARY

00891 EXPERIENCED FRED        HOWL        MATHEMATICS     $59,594
```

```
DTR> DROP
DTR> PRINT
No record selected, printing whole collection

                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT        SALARY

39485 EXPERIENCED DEE          TERRICK     PHILOSOPHY       $55,829
48573 TRAINEE     SY           KELLER      PHILOSOPHY       $31,546
49843 TRAINEE     BART         HAMMER      PHILOSOPHY       $26,392
84375 EXPERIENCED MARY         NALEVO      PHILOSOPHY       $56,847

DTR>
```

Form a collection of ASTRONOMERS from the EMPLOYEES domain. Put a DROP statement in a FOR statement that identifies records with STATUS equal to EXPERIENCED to drop three records. Display the collection. Put a DROP statement in a FOR statement that contains no RSE to drop the remaining records:

```
DTR> READY EMPLOYEES
DTR> FIND ASTRONOMERS IN EMPLOYEES WITH DEPT CONT "ASTR"
[5 records found]
DTR> PRINT ASTRONOMERS

                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT        SALARY

00012 EXPERIENCED CHARLOTTE  SPIVA       ASTRONOMY        $75,892
78923 EXPERIENCED LYDIA      HARRISON    ASTRONOMY        $40,747
83771 EXPERIENCED GERALD     BOOLE       ASTRONOMY        $21,000
87701 TRAINEE     NATHANIEL  CHONTZ      ASTRONOMY        $24,502
93811 TRAINEE     MARTHA     BOOLE       ASTRONOMY        $20,000

DTR> FOR ASTRONOMERS WITH STATUS EQUAL "EXPERIENCED" DROP
DTR> PRINT ASTRONOMERS

                      FIRST       LAST
 ID      STATUS        NAME       NAME          DEPT        SALARY

87701 TRAINEE     NATHANIEL  CHONTZ      ASTRONOMY        $24,502
93811 TRAINEE     MARTHA     BOOLE       ASTRONOMY        $20,000

DTR> FOR ASTRONOMERS DROP
DTR> PRINT ASTRONOMERS
DTR>
```

Use the DROP statement to remove records from a domain to see the error message PRO/DATATRIEVE displays:

```
DTR> FOR EMPLOYEES WITH DEPT CONT "MATH" DROP
DROP can only be used for collections
DTR>
```

15

Creating, Using, and Releasing Variables

Chapter 15

Creating, Using, and Releasing Variables

This chapter describes the statements you can use to create variables, assign values to variables, and release variables from memory when you finish with them:

Creating Variables
Assigning Values to Variables
Releasing Variables

Variables are similar to fields in a record definition, except that they exist only while you are using PRO/DATATRIEVE. Variables disappear when you exit from PRO/DATATRIEVE. To use a variable again, you need to create it again. When you create a variable, you name the variable and define its characteristics with field definition clauses described in Chapter 8.

Among other things, examples in this chapter show how to use variables to calculate salary and price increases, how to use date variables, and how to translate table codes with variables.

CREATING VARIABLES

Use the DECLARE statement to create and define a variable.

Format

```
DECLARE variable-name field-definition-clause(s).
```

Explanation

Variable-name names the variable you want to define.

Field-definition-clause(s) define the characteristics of the variable and specify how PRO/DATATRIEVE should store and interpret information stored in the variable. If you include more than one field definition clause on a line, separate them with spaces or tabs.

You must include a COMPUTED BY, PICTURE, or USAGE field definition clause in a variable definition to specify the data type and length of the variable. You should include an EDIT-STRING clause to tell PRO/DATATRIEVE how to display the variable. If you like, you can include any other field definition clauses, except OCCURS or REDEFINES. A variable can have only the properties of an elementary, single-item field.

If you press DO or RETURN to put a field definition clause on a new line, PRO/DATATRIEVE prompts with the DFN> prompt. To end the variable definition, type a period after the last field definition clause or in response to the DFN> prompt.

Comments

A variable name, like other PRO/DATATRIEVE names, can consist of from 1 to 31 letters, numbers, hyphens, and underscores and should begin with a letter and end with a letter or number.

When the DECLARE statement executes, PRO/DATATRIEVE creates the variable, initializing numeric variables to zero and alphanumeric and date variables to blanks.

When you declare a variable in a DECLARE statement that is not nested in another PRO/DATATRIEVE statement, you create a global variable. A global variable exists until you declare another global variable with the same name, exit from PRO/DATATRIEVE, or explicitly release the variable with the RELEASE command.

When you put a DECLARE statement in another PRO/DATATRIEVE statement, such as a BEGIN-END or THEN statement, PRO/DATATRIEVE creates a local variable. A local variable exists only within the statement that contains the DECLARE statement. When the statement that contains the DECLARE statement finishes executing, PRO/DATATRIEVE releases local variables created within that statement block.

If, for example, you define a local variable in the third BEGIN-END block in a series of four nested BEGIN-END blocks, you can refer to, assign values to, and retrieve values from the variable only in the third and fourth BEGIN-END blocks. That local variable, however, has no meaning for any statement outside those two BEGIN-END blocks.

Unless you include a COMPUTED BY clause when you define a global variable, the global variable retains the value you assign it until you assign a new value to it, declare another global variable with the same name, exit from PRO/DATATRIEVE, or explicitly release the variable with a RELEASE command. The value of a global variable defined with a COMPUTED BY clause depends on the expression that controls the computation. If the expression is based on the value of a field in a record, the variable has a value only when there is a valid single record context for the field value.

If you declare a variable in a FOR loop or REPEAT statement, PRO/DATATRIEVE initializes the variable each time it executes the loop or statement.

If you define a variable as a date value, you can assign today's date to it with the "TODAY" value function. See the next section in this chapter for more information on assigning a value to a variable and on using the "TODAY" value function.

Examples

Declare a global variable, NEW-SALARY, computed from the SALARY field. Include an EDIT-STRING clause to display NEW-SALARY as a money value. Ready the EMPLOYEES domain and try to display a value for NEW-SALARY. PRO/DATATRIEVE displays a message because there is no valid single record context for the SALARY field used by the NEW-SALARY variable. To compute a value for NEW-SALARY, create a context with a PRINT statement that displays the ID, SALARY, and NEW-SALARY for all employees in the math department:

```
DTR> SET NO PROMPT
DTR> DECLARE NEW-SALARY COMPUTED BY SALARY * 1.1
CON>     EDIT-STRING IS $$$,$$$.
DTR> READY EMPLOYEES
DTR> PRINT NEW-SALARY
Field "SALARY" is undefined or used out of context
DTR> PRINT ID, SALARY, NEW-SALARY OF EMPLOYEES WITH
CON>      DEPT CONT "MATH"

                 NEW
 ID    SALARY   SALARY

00001   $2,000   $2,200
00891  $59,594  $65,553
03991  $14,000  $15,400
73713  $23,000  $25,300
78375  $40,095  $44,104
83764  $41,029  $45,131
```

Declare a numeric variable, an alphanumeric variable, and a date variable, then use a PRINT statement to see how PRO/DATATRIEVE initializes the variables. Use the SHOW FIELDS command to see information about existing global variables and their data type:

```
DTR> DECLARE NUMBER PIC 9(2).
DTR> DECLARE CHARACTERS PIC X(5).
DTR> DECLARE T-DATE USAGE IS DATE.
DTR> PRINT NUMBER, CHARACTERS, T-DATE

                          T
NUMBER CHARACTERS     DATE

  00

DTR> SHOW FIELDS
Global variables:
    T_DATE  [Date]
    CHARACTERS  [Character string]
    NUMBER  [Number]
```

Declare a date variable, DUE-DATE, and assign a future date to it. Subtract today's date, using the "TODAY" date value expression, from DUE-DATE to find out how many more days until the due date for a project:

```
DTR> DECLARE DUE-DATE USAGE IS DATE.
DTR> DUE-DATE = "9-1-83"
DTR> PRINT DUE-DATE - "TODAY"
           76

DTR>
```

Declare a numeric global variable named TEST. Assign a value to the variable and display the variable. In a BEGIN-END block, display the value of TEST, declare an alphanumeric local variable named TEST, assign a value to TEST, and display TEST. After the BEGIN-END statement executes, display TEST again to see that the local variable has disappeared and that the global variable has not changed:

```
DTR> DECLARE TEST PIC 99.
DTR> TEST = 25
DTR> PRINT TEST

TEST

 25

DTR> SET NO PROMPT
DTR> BEGIN
CON>     PRINT TEST
CON>     DECLARE TEST PIC X.
CON>     TEST = "A"
CON>     PRINT TEST
CON> END

TEST

 25

TEST

 A

DTR> PRINT TEST

TEST

 25

DTR>
```

ASSIGNING VALUES TO VARIABLES

Use an assignment statement to assign a value to a variable.

Format

```
variable-name = value-expression
```

Explanation

Variable-name is the name of a variable defined with a DECLARE statement. You must declare a variable before you can assign a value to it.

Value-expression is the value you want to assign to the variable.

The equal sign (=) is *not* the same as the relational operators EQ or EQUAL; it is the assignment operator that tells PRO/DATATRIEVE to store the value on the right in the variable specified on the left.

Comments

To make PRO/DATATRIEVE prompt for the value of a variable, use a prompting value expression (*.prompt-string). PRO/DATATRIEVE then prompts for a value for the variable, using the prompt string you specified.

PRO/DATATRIEVE displays a message and prompts again for a variable value if you try to assign a value to the variable that does not match the variable definition.

If you do not use a prompting value expression in the assignment statement, PRO/DATATRIEVE accepts the value you specify as follows:

- If you specify too many digits for a numeric variable, PRO/DATATRIEVE truncates the leftmost digits and stores the remaining digits in the variable.
- If you specify too many characters for an alphanumeric variable, PRO/DATATRIEVE truncates the rightmost characters and stores the remaining characters in the variable.

- If you specify a value that a VALID IF clause in the variable definition does not allow, PRO/DATATRIEVE does not execute the assignment statement.

Use the assignment statement to assign today's date to a date variable. First, define a variable with the USAGE IS DATE clause. Then, assign today's date to the variable with the PRO/DATATRIEVE date value expression, "TODAY":

```
DTR> DECLARE CURRENT-DATE USAGE IS DATE.
DTR> CURRENT-DATE = "TODAY"
DTR> PRINT CURRENT-DATE

  CURRENT
   DATE

11-Jul-83

DTR>
```

Note that if you use a prompting value expression to prompt for the date, you do not need to use quotation marks with TODAY:

```
DTR> DECLARE TEST-DATE USAGE IS DATE.
DTR> TEST-DATE = *,"TODAY'S DATE"
Enter TODAY'S DATE: TODAY
DTR> PRINT TEST-DATE

   TEST
   DATE

11-Jul-83

DTR>
```

Examples

Declare the global variable NEW-PRICE, assign a value to it, and display the variable:

```
DTR> DECLARE NEW-PRICE PIC 9(5)
DFN>    EDIT-STRING IS $$$,$$$.
DTR> NEW-PRICE = 35000
DTR> PRINT NEW-PRICE

  NEW
 PRICE

$35,000
DTR>
```

Declare two global variables, OLD-COST and NEW-COST. Assign the value stored in the COST field for the first item in the INVENTORY domain to OLD-COST, then assign the value of OLD-PRICE * 1.1 to NEW-PRICE:

```
DTR> READY INVENTORY
DTR> DECLARE NEW-COST PIC 9(3)V99 EDIT-STRING IS $$$$.$$.
DTR> DECLARE OLD-COST PIC 9(3)V99 EDIT-STRING IS $$$$.$$.
DTR> FIND FIRST 1 INVENTORY
[1 Record found]
DTR> SELECT
DTR> PRINT

   ITEM        ITEM       ON     UNIT      TOTAL
   NAME       NUMBER     HAND    PRICE     VALUE

BAFFLES     0981414899    2353    $2.19   $5,153.07

DTR> OLD-COST = COST
DTR> PRINT OLD-COST

  OLD
 COST

  $2.19

DTR> NEW-COST = OLD-COST * 1.1
DTR> PRINT NEW-COST

  NEW
 COST

  $2.40

DTR>
```

RELEASING VARIABLES

Use the RELEASE command to end your control over one or more global variables and to free the computer memory they occupy.

Format

```
RELEASE variable-name,...
```

Explanation

Variable-name is the name of the global variable you want to release. To release more than one variable, separate variable names with commas.

Comments

You can also use the RELEASE command to release collections and tables. Simply list them as you would list multiple variable names.

When the RELEASE command executes, PRO/DATATRIEVE releases the specified collections, tables, and variables by removing them from memory. The effect is very much like the FINISH command.

When you specify more than one collection, table, or variable in the RELEASE command, PRO/DATATRIEVE releases the items in left-to-right order. When PRO/DATATRIEVE cannot release an item, the RELEASE command stops executing. PRO/DATATRIEVE displays a message indicating which item it could not release and does not release any items that follow.

Before you use the RELEASE command, use the SHOW FIELDS to see what variables you have created.

You do not have to use the RELEASE command in the following cases:

- When you end a PRO/DATATRIEVE session, PRO/DATATRIEVE automatically releases all collections, tables, and global variables.
- When you declare a global variable with the same name as an existing variable, PRO/DATATRIEVE replaces the old global variable declaration with the new declaration.

You cannot assign a value to or retrieve the value from a global variable that you have released. You can redefine the variable with the DECLARE statement, but the previous value is lost. PRO/DATATRIEVE initializes the variable to zero if numeric or blank if alphanumeric.

Example

Declare two global variables, DEPT and NEW-DEPT. Assign values to them, display translation values for the variables using the DEPT-TABLE table, then release the variables. Include an edit string in the PRINT statement so that PRO/DATATRIEVE displays all of the translation. PRO/DATATRIEVE displays only the first 10 characters of the translation if you do not include an edit string:

```
DTR> DECLARE DEPT PIC XX.
DTR> DECLARE NEW-DEPT PIC XX.
DTR> DEPT = "SD"
DTR> NEW-DEPT = "RD"
DTR> PRINT DEPT VIA DEPT-TABLE USING T(25)

        DEPT

Sales Department

DTR> PRINT NEW-DEPT VIA DEPT-TABLE USING T(25)

        NEW
        DEPT

Research and Development

DTR> SHOW TABLES
Tables loaded:
        DEPT_TABLE
Tables:
        DEPT_TABLE      RIG_TABLE

DTR> SHOW FIELDS
Global variables:
    NEW_DEPT    [Character string]
    DEPT    [Character string]
DTR> RELEASE DEPT
DTR> SHOW FIELDS
Global variables:
    NEW_DEPT    [Character string]
DTR> RELEASE NEW-DEPT
DTR> SHOW FIELDS
No Domains Readied or Global Variables Declared
DTR>
```

16

Creating and Managing Dictionaries

Chapter 16

Creating and Managing Dictionaries

This chapter describes the commands you can use to create and manage a PRO/DATATRIEVE dictionary. It also shows how to display information about your current dictionary and the PRO/DATATRIEVE environment:

Creating a Dictionary
Displaying Dictionary and
    Environment Information
Changing Dictionaries
Deleting Dictionary Definitions
Copying Dictionary Definitions

A dictionary contains domain, record, procedure, and table definitions. PRO/DATATRIEVE supplies you with a default dictionary called PRODTR.DIC. When you select PRO/DATATRIEVE from the Main Menu, your current dictionary is PRODTR.DIC. You can keep all your data definitions in PRODTR.DIC, but it is more orderly and efficient to store unrelated definitions in separate dictionaries.

You can change from one dictionary to another, and you can display general and specific information about your current dictionary. You can also display information about readied domains, established collections, tables currently in memory, and selected records.

You can transfer definitions stored in one dictionary to another dictionary by copying the definitions you want to a command file. You then set the destination dictionary as the current dictionary and execute the command file to store the definitions.

You can also edit dictionary definitions. If you need to make only a few changes to an existing definition, use the PRO/DATATRIEVE Editor, as described in Chapter 17. If you need to make extensive modifications, copy the definition to a command file with the EXTRACT command as described in this chapter, exit from PRO/DATATRIEVE, edit the command file with PROSE, and return to PRO/DATATRIEVE to execute the command file and store the new definition.

Among other things, examples in this chapter show how to:

- Create dictionaries on diskettes
- Change to a dictionary in a directory other than your current directory
- Move definitions from one dictionary to another

CREATING A DICTIONARY

Use the DEFINE DICTIONARY command to create a dictionary file and name a dictionary. PRO/DATATRIEVE sets the new dictionary as your current dictionary.

Format

```
DEFINE DICTIONARY file-spec
```

Explanation

File-spec names the dictionary and causes PRO/DATATRIEVE to create an empty indexed file with that name to hold dictionary definitions.

If you want the dictionary file to reside in your current directory, you need specify only a file name. PRO/DATATRIEVE creates the dictionary and gives the file a .DIC file type. Like all Professional file names, a dictionary file name can consist of from one to nine letters or numbers.

To create a dictionary file in a different directory or on a different device, or to specify a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

Comments

If you do not type a file specification on the same line as DEFINE DICTIONARY, PRO/DATATRIEVE prompts with the DFN> prompt. After you specify a file, PRO/DATATRIEVE creates the specified dictionary and sets the newly created dictionary as your current dictionary. You can then use the data definition commands as described in Chapter 7 to store definitions in the new dictionary.

Readied domains and tables from your previous dictionary that are currently in memory remain available for your use.

To verify the creation of the new dictionary, use the SHOW DICTIONARY command. PRO/DATATRIEVE displays the name of the new dictionary as the new current dictionary and displays information on domains previously readied and tables previously in memory. To return to your default dictionary or to establish another dictionary as the current dictionary, use the SET DICTIONARY command.

When you define a dictionary on a different volume or in a different directory from your current directory, PRO/DATATRIEVE does not set the new directory as your current directory. Thus, it is a good idea to store data files in the same directory as the dictionary that contains the domain definitions. To do this, specify the same volume name and directory for the file named in the DEFINE DOMAIN command as specified in the DEFINE DICTIONARY command. By doing this, you can access domains in different directories by simply setting the dictionary.

It is a good idea to make periodic copies of your PRO/DATATRIEVE dictionaries on diskettes. To copy a dictionary file, follow these steps:

1. Use the SHOW DICTIONARY command to obtain the full file specification of your current dictionary.
2. Exit from PRO/DATATRIEVE and choose "File services" from the Main Menu.
3. Choose "Back up disk file" or "Copy file" from the File Services Menu and follow the directions to back up or copy the dictionary file.

Examples

Define a dictionary file named NEWDIC in your current directory, USERFILES. PRO/DATATRIEVE gives the file an extension of .DIC and sets NEWDIC.DIC as your current dictionary:

```
DTR> SHOW DICTIONARY
The current dictionary is DW1:[ZZAP00014]PRODTR.DIC;1
DTR> DEFINE DICTIONARY NEWDIC
DTR> SHOW DICTIONARY
The current dictionary is DW1:[USERFILES]NEWDIC.DIC;1
DTR>
```

Define a dictionary named CUSTOMERS in a directory named CUSTOMERS on a diskette with the volume name CUSTINFO. Confirm the creation of the dictionary and reset the current dictionary to the PRO/DATATRIEVE default dictionary:

```
DTR> DEFINE DICTIONARY CUSTINFO:[CUSTOMERS]CUSTOMERS
DTR> SHOW DICTIONARY
The current dictionary is DZ2:[CUSTOMERS]CUSTOMERS.DIC;1
DTR> SET DICTIONARY
DTR> SHOW DICTIONARY
The current dictionary is DW1:[ZZAP00014]]PRODTR.DIC;1
DTR>
```

Create a new dictionary on a volume named MAIL, then create a data file stored on that volume:

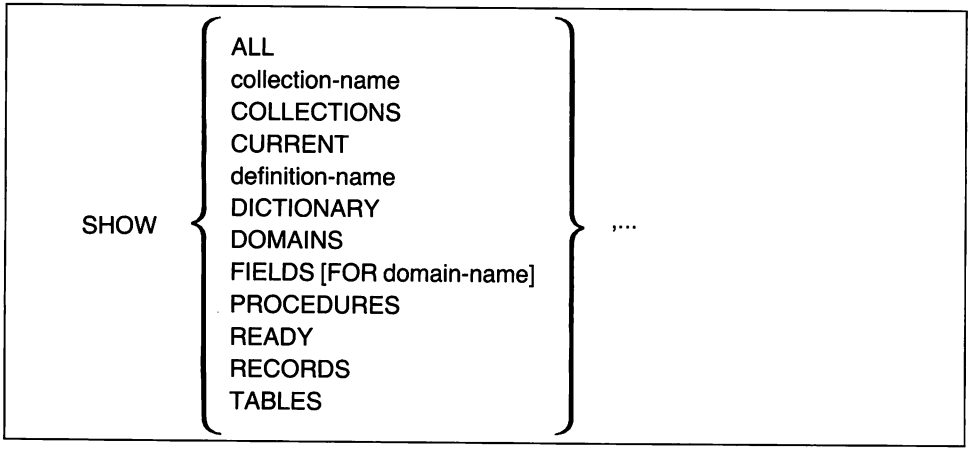
```
DTR> SHOW DICTIONARY
The current dictionary is DW1:[ZZAP00014]PRODTR.DIC;1
DTR> DEFINE DICTIONARY MAIL:[MAILLIST]NAMES
DTR> SHOW DICTIONARY
The current dictionary is DZ2:[MAILLIST]NAMES.DIC;1
DTR> DEFINE DOMAIN PHONES USING PHONE-REC
DFN>    ON MAIL:[MAILLIST]PHONES.IND;
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>    03 NAME PIC X(20).
DFN>    03 NUMBER PIC 9(7)
DFN>       EDIT-STRING IS XXX-XXXX.
DFN>   03 AREA-CODE PIC XXX VALID IF AREA-CODE IN AREA-TABLE.
DFN> ;
[Record PHONE_REC is 30 bytes long]
DTR> DEFINE FILE FOR PHONES KEY = NAME (DUP);
DTR>
```

You can access the PHONES domain from any current directory by setting your dictionary to MAIL:[MAILLIST]NAMES.

DISPLAYING DICTIONARY AND ENVIRONMENT INFORMATION

Use the SHOW command to display information about the contents of your current dictionary and about your current PRO/DATATRIEVE session.

Format



Explanation

You tell PRO/DATATRIEVE what information to display by specifying one of the elements shown in the brackets:

- *ALL* displays the names of all the definitions in your current dictionary, the name of your current dictionary, the names of established collections, the names of and information about readied domains, and the names of tables currently loaded in memory.
- *Collection-name* and *CURRENT* display information about a collection. PRO/DATATRIEVE displays the collection name (CURRENT, if you specify CURRENT), the name of the domain from which the collection was formed, the number of records in the collection, the status of the selected record within the collection, and, if you have sorted the collection, the names of the keys on which the collection was sorted.
- *COLLECTIONS* displays the names of all collections currently established and, if the CURRENT collection is a named collection, indicates which collection is the CURRENT collection.

- *Definition-name* is the name of a domain, record, procedure, or table stored in your current dictionary. PRO/DATATRIEVE displays the definition of the dictionary object you specify.
- *DICTIONARY* displays the full file specification of your current dictionary.
- *DOMAINS* displays the names of all domains defined in your current dictionary.
- *FIELDS* displays the names, data types, and information about index keys for fields of all readied domains, or for the domain named in the optional *FOR* clause. The SHOW FIELDS command also displays the names and data types of all global variables. See Chapter 15 for information on defining and using variables.
- *PROCEDURES* displays the names of all procedures defined in your current dictionary.
- *READY* displays the names of all readied domains, the file type of the data file associated with the domain, and the access mode specified when you readied the domain.
- *RECORDS* displays the names of all records defined in your current dictionary.
- *TABLES* displays the names of all tables currently loaded in memory and the names of all tables defined in your current dictionary.

You can specify multiple SHOW command elements by separating each element from the next with a comma. PRO/DATATRIEVE displays information in the order specified.

If you just type SHOW and press DO, PRO/DATATRIEVE prompts you with "[Looking for element to SHOW]".

Comments

You can use the SHOW command to display information only about definitions stored in your current dictionary. To display information about another dictionary, use the SET DICTIONARY command to change your current dictionary.

When you have more than one existing collection, the SHOW COLLECTIONS command displays the names of existing collections as a list with the most recently established collection at the top and the oldest existing collection at the bottom. Knowing this order can help you see how PRO/DATATRIEVE resolves field names in value expressions and searches for a selected record to establish a single record context for MODIFY, PRINT, and ERASE statements.

Examples

Use the SHOW ALL command to display all available information about your current dictionary and PRO/DATATRIEVE environment:

```
DTR> SHOW ALL
Domains:
        CUSTOMERS       EMPLOYEES       INVENTORY       PAY
        PHONES          STOCKS          TRANS
Records:
        CUSTOMERS_REC   EMPLOYEES_REC   INVENTORY_REC   PAY_REC
        PHONE-REC       STOCKS_REC      TRANS_REC
Procedures:
        DEPT_SAL        MAIL_FORMAT     MAIL_REPORT     SALARY_REPORT
        SUM_REPORT      UPDATE_MASTER   VENDORS_MOD     WRITE_PRICE
Tables:
        AREA_TABLE      COMPANIES       DEPT_TABLE
The current dictionary is DW1:[EXAMPLES]EXAMPLES.DIC;1
No established collections
No ready domains
DTR>
```

Ready the TRANS domain for READ access and the EMPLOYEES domain for WRITE access. Then use the SHOW READY command to display information about readied domains:

```
DTR> READY TRANS
DTR> READY EMPLOYEES WRITE
DTR> SHOW READY
Ready domains:
        EMPLOYEES: RMS INDEXED, PROTECTED WRITE
        TRANS: RMS INDEXED, PROTECTED READ
DTR>
```

Load the COMPANIES table into memory by using it to display the COMPANY field of records stored in the TRANS domain. Use the SHOW TABLES command

to display the names of tables loaded in memory and of tables defined in the current dictionary:

```
DTR> READY TRANS
DTR> PRINT COMPANY VIA COMPANIES USING X(20) OF FIRST 1 TRANS

      COMPANY
        NAME

TACTICAL AID, INC.

DTR> SHOW TABLES
Tables loaded:
        COMPANIES
Tables:
        AREA_TABLE       COMPANIES       DEPT_TABLE
DTR>
```

Use the SHOW command to display the TRANS domain definition:

```
DTR> SHOW TRANS
DOMAIN TRANS
 USING TRANS-REC ON TRANS;
DTR>
```

CHANGING DICTIONARIES

Use the SET DICTIONARY command to specify a new current dictionary.

Format

```
SET DICTIONARY [file-spec]
```

Explanation

File-spec is the file specification of the dictionary you want to set as your current dictionary. When you start a PRO/DATATRIEVE session, your current dictionary is PRODTR.DIC. If you type SET DICTIONARY without a file specification, PRO/DATATRIEVE sets your dictionary back to PRODTR.DIC.

If the dictionary you name resides in your current directory, you need to specify only a file name. PRO/DATATRIEVE searches for a file with that name and a file type of .DIC in the current directory.

To access a dictionary file in a different directory or on a different device, or to specify a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

Comments

If PRO/DATATRIEVE does not find the dictionary file you specify, it displays a message and does not change the current dictionary.

When PRO/DATATRIEVE finds the dictionary file you specify, it sets that dictionary as your current dictionary. To display the names of definitions stored in the new current dictionary, use the SHOW ALL command.

Note that if you specify a dictionary in a different directory, your current directory does not change to correspond to the directory where the new current dictionary resides. Unless you have specified complete file specifications for data files in DEFINE DOMAIN commands, you cannot ready domains defined in the new dictionary. For more information, see the section on defining domains in Chapter 7 and the section on defining dictionaries in this chapter.

Examples

Set your current dictionary to CUSTOMERS.DIC in a directory named CUSTOMERS on a diskette with the volume name CUSTINFO. Then use the SHOW DICTIONARY command to verify the change:

```
DTR> SET DICTIONARY CUSTINFO:[CUSTOMERS]CUSTOMERS
DTR> SHOW DICTIONARY
The current dictionary is DZ1:[CUSTOMERS]CUSTOMERS.DIC;1
DTR>
```

Return to the default PRO/DATATRIEVE dictionary, PRODTR.DIC, by typing SET DICTIONARY:

```
DTR> SET DICTIONARY
DTR> SHOW DICTIONARY
The current dictionary is DW1:[ZZAP00014]PRODTR.DIC;1
DTR>
```

DELETING DICTIONARY DEFINITIONS

Use the DELETE command to delete a domain, record, procedure, or table definition from your current dictionary.

Format

```
DELETE definition-name;
```

Explanation

Definition-name is the name of a domain, record, procedure, or table definition you want to remove from the dictionary. If you type DELETE and press DO, PRO/DATATRIEVE prompts you for a definition name.

You must type a semicolon (;) after the definition name to end the DELETE command. Otherwise, PRO/DATATRIEVE displays an error message and does not make the deletion.

Comments

You can delete only definitions stored in a dictionary, not the dictionary itself, with the DELETE command. You must use P/OS File Services to delete the dictionary file. See *Professional 300 Series User's Guide: Hard Disk System* for information on deleting files from your Professional.

When you delete a domain definition, PRO/DATATRIEVE does not delete the associated record definition or the associated data file. When you delete a record definition, PRO/DATATRIEVE does not delete the associated domain definition.

A readied domain is not affected by the deletion of the domain definition or its associated record definition. However, after you release a domain whose definition or record definition has been deleted from the dictionary, you cannot ready the domain again.

A table in memory remains there until you release it, even if you delete its definition. However, after you have released the table, you cannot use it again.

Examples

Delete a procedure from your current dictionary:

```
DTR> SHOW HELO
PROCEDURE HELO
PRINT "Good morning"
END_PROCEDURE
DTR> DELETE HELO;
DTR> SHOW HELO
"HELO" has not been defined in the dictionary
DTR>
```

Delete the domain and record definitions for a domain named XYZ:

```
DTR> READY XYZ
DTR> SHOW XYZ
DOMAIN XYX
 USING XYZ_REC ON XYZ;
DTR> SHOW XYZ-REC
RECORD XYZ_REC
 USING
01 XYZ_REC.
   03 LAST_NAME PIC X(12).
   03 FIRST_NAME PIC X(10).
;
DTR> FIND FIRST 1 XYZ
[1 Record found]
DTR> DELETE XYZ;
DTR> SHOW XYZ
"XYZ" has not been defined in the dictionary
DTR> DELETE XYZ-REC;
DTR> SHOW XYZ-REC
"XYZ_REC" has not been defined in the dictionary
DTR> SHOW READY
Ready domains:
        XYZ: RMS INDEXED, PROTECTED READ
DTR> SHOW FIELDS
XYZ
    XYZ_REC
        LAST_NAME   [Character string, indexed key]
        FIRST_NAME  [Character string]
DTR> SHOW CURRENT
Collection CURRENT
        Domain: XYZ
        Number of records: 1
        No selected record
DTR> FINISH XYZ
DTR> SHOW READY
No ready domains
DTR>
```

COPYING DICTIONARY DEFINITIONS

Use the EXTRACT command to copy one or more existing dictionary definitions from your current dictionary to a command file. You can then change dictionaries with the SET DICTIONARY command and execute the command file to store the copied definition in the new current dictionary or you can exit from PRO/DATATRIEVE and edit the command file with PROSE. When you have made the changes you want, you can return to PRO/DATATRIEVE and execute the command file to store the modified definition in a dictionary.

Format

```
EXTRACT ON file-spec definition-name,...
```

Explanation

File-spec is the file specification of the command file you want to contain the specified definition or definitions.

Definition-name is the name of the domain, record, procedure, or table definition you want copied to the command file.

If you want the command file to reside in your current directory, you need to specify only a file name for *file-spec*. PRO/DATATRIEVE creates the file in your current directory and gives it a .CMD file type. Like all Professional file names, a command file name can consist of from one to nine letters or numbers.

To create a command file in a different directory or on a different device, or to specify a different file type, you must use an extended file name. See your *Professional 300 Series User's Guide: Hard Disk System* for information on naming files and using extended file names.

Comments

You can omit the ON keyword from the EXTRACT command, but you cannot omit the file specification.

If you turn word wrapping off when using PROSE to edit a command file, do not save the word wrap setting when you exit from PROSE. If you save the command file with word wrapping off, PRO/DATATRIEVE cannot execute the file and displays an error message.

To execute the command file, either in a new dictionary or after you have edited the file, type the *at* symbol (@) and the command file name in response to the DTR> prompt. If you specified a file type other than .CMD in the EXTRACT command that created the file, you must specify the file type when you type the @ command. If you do not specify a file type, PRO/DATATRIEVE automatically searches for a .CMD file with the file name you specify in the @ command.

For more information on executing command files, see Chapter 10.

The EXTRACT command does not delete a dictionary definition. When the EXTRACT command executes, PRO/DATATRIEVE inserts a DELETE command and a DEFINE command for each definition name you specified. Each DEFINE command written to the command file contains an exact copy of the specified definition. Extracted definitions are not actually deleted until you execute the command file.

Each DELETE command tells PRO/DATATRIEVE to delete the specified dictionary definition from the current dictionary when you execute the command file. Each DEFINE command then creates a new definition with the specified name and stores it in your current dictionary.

Use the EXTRACT command to create backup copies of your dictionary definitions, to transport definitions to other dictionaries, and to compress a dictionary so that it takes up less space on your disk or diskette. If you have made extensive changes to a dictionary, you should compress it so that it takes up as little space on the disk or diskette as possible.

To compress a PRO/DATATRIEVE dictionary, take the following steps:

1. Use the SHOW ALL command to see the names of all definitions stored in the dictionary, then use the EXTRACT command to write all dictionary definitions to a command file:

```
DTR> SHOW ALL
Domains:
        CUSTOMERS       EMPLOYEES       INVENTORY       SALES_VIEW
        SUPPLIES        TRANSACTIONS
Records:
        CUSTOMERS_REC   EMPLOYEES_REC   INVENTORY_REC   SUPPLIES_REC
        TRANS_REC
Procedures:
        CUR_REPORT      CUST_REP        INV_STORE       MAIL_FORMAT
        MAIL_REPORT     TRANS_WRITE     UPDATE_MASTER   WRITE_PRICE
```

(continued on next page)

```
Tables:
        AREA_TABLE        COMPANIES
The current dictionary is DW1:[INVENTORY]INVENTORY.DIC;1
No established collections
No ready domains
DTR> EXTRACT ON NEWDIC CUSTOMERS, EMPLOYEES,
CON> INVENTORY, SALES-VIEW, SUPPLIES, TRANSACTIONS,
CON> CUSTOMERS-REC, EMPLOYEES-REC, INVENTORY-REC, SUPPLIES-REC,
CON> TRANS-REC, CUR-REPORT, CUST-REP, INV-STORE, MAIL-FORMAT,
CON> MAIL-REPORT, TRANS-WRITE, UPDATE-MASTER, WRITE-PRICE,
CON> AREA-TABLE, COMPANIES
DTR>
```

2. Use the DEFINE DICTIONARY command to define the dictionary again:

```
DTR> DEFINE DICTIONARY [INVENTORY]INVENTORY.DIC
DTR> SHOW DICTIONARY
The current dictionary is DW1:[INVENTORY]INVENTORY.DIC;2
DTR>
```

3. Execute the command file created with the EXTRACT command to store all your definitions in the new dictionary by typing @NEWDIC.

You can move selected definitions to another dictionary by following the same steps. When you move a definition from one dictionary to another, use the SHOW command to see whether a definition with the same name as the one specified in the DELETE and DEFINE commands exists in the new dictionary. SHOW TRANS for example, shows you whether a definition named TRANS exists in the destination dictionary.

If the definition exists, you must decide whether you want to delete the existing definition and replace it with the definition in the command file. If you do not want to delete the existing definition, edit the command file to change the names in the DELETE and DEFINE commands.

Try to anticipate problems that might arise when you change current dictionaries and execute command files. Remember that no two objects in the same dictionary can have the same name, but if you shift to a new current dictionary, that new one may legitimately contain an object with the same name. You might, for example, extract the definition of procedure ABC from the dictionary WIDGETS.DIC and then set your current dictionary to THINGS.DIC, which contains a dictionary table called ABC. When you execute the command file, the DELETE ABC command in the command file deletes the ABC table definition from THINGS.DIC, and the DEFINE PROCEDURE ABC command enters the procedure definition.

You can avoid this problem by giving domain, records, tables, and procedures names that identify their function, such as ABC, ABC-REC, ABC-PRINT, ABC-TABLE.

Example

Extract definitions for MAIL-FORMAT and MAIL-REPORT from the current dictionary. Change the current dictionary to CUSTOMERS.DIC, and use the SHOW command to check for definitions with the names MAIL-FORMAT and MAIL-REPORT. Execute the command file created with the EXTRACT command to copy the MAIL-FORMAT and MAIL-REPORT definitions to CUSTOMERS.DIC:

```
DTR> EXTRACT ON MOVING MAIL-FORMAT, MAIL-REPORT
DTR> SET DICTIONARY CUSTOMERS
DTR> SHOW MAIL-FORMAT
"MAIL_FORMAT" has not been defined in the dictionary
DTR> SHOW MAIL-REPORT
"MAIL_REPORT" has not been defined in the dictionary
DTR> @MOVING
DELETE MAIL_FORMAT;
"MAIL_FORMAT" has not been defined in the dictionary
DEFINE PROCEDURE MAIL_FORMAT
        PRINT SKIP 2,  COL 10, COMPANY(-),
            COL 10, CONTACT(-),
            COL 10, STREET(-),
            COL 10, CITY!!", "!STATE!", "!ZIP
END_PROCEDURE
DELETE MAIL_REPORT;
"MAIL_REPORT" has not been defined in the dictionary
DEFINE PROCEDURE MAIL_REPORT
        REPORT CUSTOMERS
        SET REPORT_NAME = "MAILING LIST"
        SET COLUMNS-PAGE = 40
        SET NO DATE, NO NUMBER
        :MAIL_FORMAT
        END_REPORT
END_PROCEDURE
DTR>
```

17

Using the PRO/DATATRIEVE Editor

Chapter 17

Using the PRO/DATATRIEVE Editor

This chapter describes the EDIT command that calls the PRO/DATATRIEVE Editor and the editing commands you can use to edit dictionary definitions:

Calling the PRO/DATATRIEVE Editor
Deleting Lines from a Definition
Inserting Lines in a Definition
Replacing Lines
Making Substitutions in a Definition

The PRO/DATATRIEVE Editor, sometimes referred to as QED because of the QED> prompt it uses, allows you to modify existing domain, record, procedure, and table definitions stored in your current dictionary without exiting from PRO/DATATRIEVE.

By using the Editor to make changes to a dictionary definition, you do not have to exit from PRO/DATATRIEVE as you do to edit a command file. When you leave the Editor, PRO/DATATRIEVE returns to the DTR> prompt. All readied domains and established collections remain unchanged by your editing session.

With the PRO/DATATRIEVE Editor, you can substitute characters, insert lines, replace lines, delete lines, and display lines in a dictionary definition. For extensive modifications to a dictionary definition, you may want to write the definition to a command file with the EXTRACT command and exit from PRO/DATATRIEVE to edit the command file.

CALLING THE PRO/DATATRIEVE EDITOR

Use the EDIT command to call the PRO/DATATRIEVE Editor.

Format

```
EDIT definition-name [ADVANCED]
```

Explanation

Definition-name is the name of the domain, record, procedure, or table definition you want to edit.

ADVANCED indicates that you want to edit a domain or record definition. You must specify ADVANCED before you can edit a domain or record definition.

When you use the EDIT command, PRO/DATATRIEVE invokes the Editor, loads the specified definition into the Editor's buffer, and prompts with QED>, the editing command mode prompt. You can type any Editor command in response to the QED> prompt.

Comments

For quick reference, Table 17-1 summarizes Editor commands. The remaining sections of this chapter describe Editor commands in detail.

To end an editing session, press the EXIT key or type EXIT or QUIT in response to the QED> prompt:

- The EXIT key or EXIT command tells PRO/DATATRIEVE to make the specified changes to the definition and to replace the old definition with the modified version.
- The QUIT command tells PRO/DATATRIEVE to display the message "Execution failed", leave the dictionary definition unchanged by the editing session, and return to the DTR> prompt.

The PRO/DATATRIEVE Editor does not check the syntax of any commands or statements in the edited definition. If you make an error when editing a definition, PRO/DATATRIEVE does not identify the error until you ready the modified domain, refer to the modified table, or execute the modified procedure.

Table 17-1: Summary of PRO/DATATRIEVE Editor Commands

Command	Function
D or DELETE	Deletes the current line or a specified range of lines and sets the current line pointer to the line following last deleted line.
(EXIT) or (CTRL/Z)	In response to the QED> prompt, ends an editing session, replaces the dictionary definition with the edited version, and returns you to the DTR> prompt; in response to the IN> prompt, ends an insert or replace operation and returns you to the QED> prompt.
EX or EXIT	Ends an editing session, replaces the dictionary definition with the edited version, and returns you to the DTR> prompt. Do not type this command to exit from insert mode; the Editor interprets the EXIT command as a line to insert in the definition.
I or INSERT	Begins insert mode and displays the IN> prompt. Lines you type are inserted before the current line. Press the EXIT key or CTRL/Z to exit from insert mode and return to the QED> prompt. The current line does not change when you leave insert mode.
QUIT	Returns you to the DTR> prompt and leaves the dictionary definition unchanged by the editing session.
R or REPLACE	Deletes the current line or the specified range of lines, begins insert mode, and displays the IN> prompt. Press the EXIT key or CTRL/Z to exit from insert mode. Do not type the EXIT command; the Editor interprets the EXIT command as a line to insert in the definition. When you exit from insert mode, the Editor sets the current line to the line following the last deleted line.
S	Substitutes one string of characters (the substitution string) for another (the target string) in the current line or in a specified range of lines. The Editor sets the line in which the last substitution occurred as the current line.

The PRO/DATATRIEVE Editor has two different prompts that indicate the type of input it expects:

- When you call the Editor with the EDIT command, the Editor prompts with QED>, indicating that it expects you to type in an Editor command. This is called command mode. In command mode, the Editor interprets all your inputs as commands, and you can display and alter the text of the dictionary object.

(continued on next page)

- After you type an INSERT or REPLACE command, the Editor prompts with IN>, indicating that it expects you to type text you want inserted in the definition. This is called insert mode. To exit from insert mode and return to the QED> prompt, press the EXIT key or press the CTRL and Z keys simultaneously (CTRL/Z).

You can modify a specific line or a range of lines with some Editor commands by specifying a range keyword or symbol. Table 17-2 lists range keywords and symbols and describes their effect.

Table 17-2: Range Keywords and Symbols

Range Specifier	Lines Specified
ALL "string"	All lines containing the specified string. You can enclose the string in either single or double quotation marks. The search starts from the first line of the definition and continues to the end of the text buffer. The Editor does not distinguish between upper- and lowercase letters.
BEFORE	All lines preceding and including the current line.
BEGIN	First line of the definition.
END	The line after the last line of the definition.
REST	Current line and all remaining lines in the definition.
"string"	The current line or the next line containing the specified string. You can enclose the string in either single or double quotation marks. The search starts with the current line and continues toward the end of the text buffer. The Editor does not distinguish between upper- and lowercase letters.
WHOLE	All lines of the definition.
.	Current line.

Some editing commands move you through the text from one line to another, thus changing the current line. Other commands display or alter lines at various places in the text but leave the current line unchanged.

The Editor uses a line pointer to keep track of the current line. The line pointer points to the entire current line, not to any part of the line. There can be only one current line. The current line can be a blank line at the end of the definition, as well as a line of text, thus allowing you to add text at the end of the definition. The symbol [EOB] marks the end of the definition.

To display a line or a range of lines, type a range specification keyword or symbol in response to the QED> prompt. For example:

- To display the current line, type a period (.) in response to the QED> prompt.
- To display the entire definition and set the current line to the first line of the definition, type WH or WHOLE in response to the QED> prompt.
- To move through the definition, changing the current line, press DO or RETURN in response to the QED> prompt. The Editor moves to the next line and sets it as the current line.
- To search for a text string in a definition, type the string, enclosed in single or double quotation marks, in response to the QED> prompt. The Editor searches for the string from the current line.

When you display a range of lines, the first line displayed becomes the current line, with one exception: if you specify END, the line pointer points to the end of the definition as indicated by the [EOB] symbol.

Table 17-3 shows the effect of range keywords and symbols on the current line when typed in response to the QED> prompt to display lines. Each of the following sections indicates which range keywords and symbols you can use with particular commands and describes their effect on the current line.

Table 17-3: Setting the Current Line with Range Keywords and Symbols

Range Specifier	New Current Line
ALL "string"	First line containing "string".
BEFORE	First line of the definition.
BEGIN	First line of the definition.
END	Empty line after the last line of the definition.
REST	No new current line.
"string"	First line containing "string".
WHOLE	First line of the definition.
DO or RETURN	Line just displayed.
.	No new current line.

Be careful when editing record definitions. If you change the length of the record definition, you have to create a new data file and transfer the old information from the old file to the new one. If you change the name of any fields in a record definition, you may have to edit the procedures and reports that refer to those fields to reflect the changes.

Changes made with the Editor affect only definitions stored in the dictionary. They do not affect readied domains and tables currently in memory. If, for example, you change the record definition of a readied domain, you cannot see the effect of this change until you finish the associated domain with the FINISH command and ready the domain again. Similarly, to use an edited table, you must release the version of the table currently in memory and refer to it again to obtain the edited version.

Examples

Edit a table definition, display the definition by typing WHOLE, search for the line containing "ELSE", display that line as the current line, and use the R command to replace the line. Press the EXIT key once to exit from insert mode and once again to exit from the Editor:

```
DTR> EDIT DEPT-TABLE
QED> WHOLE
        PE : "Plant Engineering",
        CS : "Customer Support",
        RD : "Research and Development",
        SD : "Sales Department",
        ELSE "UNKNOWN DEPARTMENT"
        END_TABLE
QED> "ELSE"
        ELSE "UNKNOWN DEPARTMENT"
QED> R
IN>     ELSE "Unknown Department"
IN>     EXIT
QED> EXIT
DTR>
```

Edit the TRANS-REC record definition and use the S command to change the edit string for the TRANS-DATE field from NN/DD/YY to DD-MMM-YYYY. Use an asterisk (*) as the delimiter to separate the format you want to change from the new format. Press the EXIT key to save the modified definition and exit from the Editor:

```
DTR> EDIT TRANS-REC ADVANCED
QED> S *NN/DD/YY*DD-MMM-YYYY*
                EDIT-STRING IS DD-MMM-YYYY
QED> EXIT
DTR>
```

DELETING LINES FROM A DEFINITION

Use the DELETE command to delete one or more lines from the dictionary definition.

Format

```
DELETE [range]
```

Explanation

Range specifies which line or range of lines you want to delete. If you omit a range keyword or symbol, PRO/DATATRIEVE deletes only the current line. You can use all range keywords except END with the DELETE command. Table 17-2 lists range keywords.

Comments

To save typing time, you can abbreviate the DELETE command to D.

After the Editor deletes the current line or the specified range of lines, it sets the line pointer to the line following the last deleted line.

Examples

Edit the DEPT-SAL procedure and delete all the AT TOP and AT BOTTOM statements in the procedure. To do this, use two DELETE commands to delete lines that contain the string "AT" and lines that contain the string "TOTAL". Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT DEPT-SAL
QED> WH
        READY EMPLOYEES READ
        FIND EMPLOYEES WITH (DEPT CONT "MATH") OR
            (DEPT CONT "ASTR") SORTED BY STATUS, DEPT
        REPORT CURRENT
        SET REPORT_NAME = "DETAILED SALARY REPORT"
        AT TOP OF DEPT PRINT DEPT
        AT TOP OF STATUS PRINT STATUS
        PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
```

(continued on next page)

```
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE,
            "TOTAL:", TOTAL SALARY USING $,$$$,$$$, SKIP
        AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
            TOTAL SALARY USING $,$$$,$$$,
            COL 42, "----------", SKIP
        AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
            TOTAL SALARY USING $,$$$,$$$
        SET NO DATE
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> D ALL "AT"
QED> D ALL "TOTAL"
QED> WH
        READY EMPLOYEES READ
        SET REPORT_NAME = "DETAILED SALARY REPORT"
        PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
            COL 42, "----------", SKIP
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> QUIT
Execution failed
DTR>
```

Establish the PRINT statement that precedes the first AT BOTTOM statement as the current line, then delete all lines from the beginning of the DEPT-SAL procedure through the current line:

```
DTR> EDIT DEPT-SAL
QED> "PRINT ID"
        PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
QED> .
        PRINT ID, FIRST_NAME!!" "!LAST_NAME ("NAME"), SALARY
QED> DELETE BEFORE
QED> WHOLE
        AT BOTTOM OF STATUS PRINT SKIP, COL 42, STATUS, SPACE,
            "TOTAL:", TOTAL SALARY USING $,$$$,$$$, SKIP
        AT BOTTOM OF DEPT PRINT COL 42, DEPT, SPACE, "TOTAL:",
            TOTAL SALARY USING $,$$$,$$$,
            COL 42, "----------", SKIP
        AT BOTTOM OF REPORT PRINT COL 42, "GRAND TOTAL SALARY:",
            TOTAL SALARY USING $,$$$,$$$
        SET NO DATE
        SET COLUMNS_PAGE = 80
        END_REPORT
QED> QUIT
Execution failed
DTR>
```

INSERTING LINES IN A DEFINITION

Use the INSERT command to add lines to a dictionary definition and to enter insert mode.

Format

```
INSERT [range]
```

Explanation

Range specifies which line you want to follow the inserted lines. That is, you insert lines before the specified line. If you do not specify a range, the Editor inserts the new lines before the current line. Table 17–2 lists range keywords.

After you type an INSERT command, the Editor prompts with the IN> prompt to indicate that you are in insert mode. The Editor inserts lines you type in response to the IN> prompt and sets the current line as follows:

- No range - Inserts lines before the current line. The current line does not change.
- BEGIN - Inserts lines before the first line of the definition and sets the current line to the old first line of the definition.
- END - Inserts lines at the end of the definition and sets the current line to the end of the definition, as indicated by the [EOB] symbol.
- "String" - Inserts lines before the first line containing "string" and sets the current line to the line containing "string".

Do not use the ALL, BEFORE, REST, or WHOLE range specifiers with the INSERT command.

Comments

To save typing time, you can abbreviate the INSERT command to I.

If you do not specify a range, the current line does not change when you exit from insert mode. If you do specify a range keyword or symbol, the line specified by the range becomes the current line.

To exit from insert mode, press the EXIT key or press the CTRL and Z keys simultaneously (CTRL/Z).

Examples

Insert comments in a procedure before the current line. Press the EXIT key to exit from insert mode, display the modified procedure with the WHOLE range specifier, then press the EXIT key to exit from the Editor and replace the procedure with the modified version:

```
DTR> EDIT MAIL-REPORT
QED> INSERT
IN>      !
IN>      ! This is a comment
IN>      !
IN>      (EXIT)
QED> WHOLE
         !
         ! This is a comment
         !
         REPORT CUSTOMERS
         SET REPORT_NAME = "MAILING LIST"
         SET COLUMNS-PAGE = 40
         SET NO DATE, NO NUMBER
         :MAIL_FORMAT
         END_REPORT
QED> (EXIT)
DTR>
```

Insert comments after the last line of the MAIL-REPORT procedure and press the EXIT key to exit from insert mode. Use the BEGIN range keyword to go back to the beginning of the definition and insert lines before the :MAIL-FORMAT line. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT MAIL-REPORT
QED> END
         [EOB]
QED> INSERT
IN>      !
IN>      ! This is the end of the procedure.
IN>      !
IN>      (EXIT)
QED> BEGIN
         REPORT CUSTOMERS
QED> ":mail"
         :MAIL_FORMAT
```

```
QED> INSERT
IN>      !
IN>      ! This invokes the formatting procedure.
IN>      !
IN>      EXIT
QED> WHOLE
         REPORT CUSTOMERS
         SET REPORT_NAME = "MAILING LIST"
         SET COLUMNS_PAGE = 40
         SET NO DATE, NO NUMBER
         !
         ! This invokes the formatting procedure.
         !
         :MAIL_FORMAT
         END_REPORT
         !
         ! This is the end of the procedure.
         !
QED> QUIT
Execution failed
DTR>
```

REPLACING LINES

Use the REPLACE command to delete and insert lines in a dictionary definition and to enter insert mode.

Format

```
REPLACE [range]
```

Explanation

Range specifies the line or range of lines you want to delete and replace. If you do not specify a range, the Editor replaces the current line. Table 17–2 lists range keywords.

After you type a REPLACE command, the Editor deletes the specified line or lines and prompts with the IN> prompt to indicate that you are in insert mode. The Editor replaces lines with text you type in response to the IN> prompt and sets the current line as follows:

- No range - Replaces the current line and sets the current line to the line following the deleted line.
- BEGIN - Replaces the first line of the definition and sets the current line to the second line of the definition.
- BEFORE - Replaces all lines preceding and including the current line. The current line is set to the line following the deleted current line.
- END - Deletes no lines and inserts text at the end of the definition. The current line is set to the end of the definition, as indicated by the [EOB] symbol.
- REST - Replaces the current line and all following lines. The current line is set to the end of the definition, as indicated by the [EOB] symbol.
- "String" - Replaces the first line containing "string" and sets the current line to the line following the deleted line.
- WHOLE - Replaces the entire definition and sets the current line to the end of the definition as indicated by the [EOB] symbol.

Do not use the ALL "string" range specifier with the REPLACE command.

Comments

To save typing time, you can abbreviate the REPLACE command to R.

To exit from insert mode, press the EXIT key or press the CTRL and Z keys simultaneously (CTRL/Z).

Examples

Delete all lines in a record definition and type in a new line to replace the definition. Exit from insert mode by pressing the EXIT key and display the current line. Then use the QUIT command to exit from the Editor without changing the record definition:

```
DTR> EDIT TRANS-REC ADVANCED
QED> R WHOLE
IN>     This should replace the whole definition
IN>     (EXIT)
QED> .
        [EOB]
QED> WHOLE
        This should replace the whole definition
QED> QUIT
Execution failed
DTR>
```

Edit the SALARY-REPORT procedure and replace the first line containing "COL 40" with a new line. Exit from insert mode by pressing the EXIT key and use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT SALARY-REPORT
QED> R "COL 40"
IN>     ! This replaces the first line containing COL 40
IN>     (EXIT)
QED> .
        COL 40, "TOTAL SALARY:", TOTAL SALARY USING $$$,$$$,$$$
QED> WHOLE
        PRINT ID, STATUS, FIRST_NAME!!" "!LAST_NAME, DEPT, SALARY
        AT BOTTOM OF REPORT PRINT SKIP,
        "NUMBER OF RECORDS:  "!COUNT,
        ! This replaces the first line containing COL 40
        COL 40, "TOTAL SALARY:", TOTAL SALARY USING $$$,$$$,$$$
        END_REPORT
QED> QUIT
Execution failed
DTR>
```

MAKING SUBSTITUTIONS IN A DEFINITION

Use the S command to substitute one character string (the substitution string) for another character string (the target string) in the current line or in a specified range of lines.

Format

```
S /target-string/substitution-string [/range]
```

Explanation

Target-string is the string you want to delete and *substitution-string* is the string you want to insert. If you do not specify a substitution string, the Editor deletes the target string.

Slashes (/) mark the beginning and end of the target and substitution strings and are called delimiters. If you do not specify a range, you need to specify only the first two delimiters. If you specify a range, you must include a delimiter after the substitution string.

Range specifies the line or range of lines where you want the substitution to take place. If you do not specify a range, the Editor makes the substitution in the first line that contains the target string.

Do not use the END range specifier with the S command.

The Editor displays all lines in which a substitution occurs. If the target string does not exist in the specified range, the Editor displays the message "No such string in this range".

When a substitution takes place, the Editor sets the current line to the last line in which a substitution occurred. If no substitution takes place, the current line remains unchanged.

Comments

The Editor recognizes only the letter S as the SUBSTITUTE command. If you type SUBSTITUTE, the Editor displays an error.

You may use any printing character not in the target or substitution string as a delimiter, as long as all delimiters in the command are identical. The Editor interprets the first character after the S command as the delimiter and treats all characters that follow it as part of the target string until it encounters a character identical to the first character.

The Editor searches for the target string from the current line towards the end of the definition except when you specify WHOLE for the range. When you specify WHOLE, the Editor begins the search at the beginning of the definition rather than at the current line.

If you specify a range, the Editor replaces all occurrences of the target string in that range with the substitution string. If you do not specify a range, the Editor replaces only the first occurrence of the target string:

```
DTR> EDIT TRANS-REC ADVANCED
QED> "$$$"
                EDIT_STRING IS $$$,$$$.$$
QED> S /$/Z/"EDIT"
                EDIT_STRING IS ZZZ,ZZZ.ZZ
QED> S/Z/$
                EDIT_STRING IS $ZZ,ZZZ.ZZ
QED> QUIT
Execution failed
DTR>
```

The first occurrence of the target string need not be in the current line.

Examples

Substitute the string HEADER for the string NAME in all lines of the TRANS-REC definition. Use the QUIT command to exit from the Editor without changing the record definition:

```
DTR> EDIT TRANS-REC ADVANCED
QED> S /NAME/HEADER/WHOLE
                QUERY_HEADER IS NUM
                QUERY_HEADER IS TYPE.
                QUERY_HEADER IS T_DATE.
             15 COMPANY_HEADER PIC IS X(3)
                VALID IF COMPANY_HEADER IN COMPANIES
                QUERY_HEADER IS COMPANY.
                QUERY_HEADER IS QUANT.
                QUERY_HEADER IS PRICE.
                QUERY_HEADER IS T_TOTAL
QED> QUIT
Execution failed
DTR>
```

Substitute five exclamation points for all occurrences of one exclamation point in the SALARY-REPORT procedure. Use the QUIT command to exit from the Editor without changing the procedure:

```
DTR> EDIT SALARY-REPORT
QED> S /!/!!!!!/REST
        !!!!! By preceding a line with an exclamation point, you
        !!!!! can include comments that tell what the procedure
        !!!!! does.  You can also use exclamation points to include
        !!!!! blank lines to make a procedure easier to read.
        !!!!!
        !!!!! This procedure prints records from the EMPLOYEES
        !!!!! domain, then prints the number of records, the
        !!!!! average salary, and the total salary at the bottom
        !!!!! of a report.
QED> QUIT
Execution failed
DTR>
```

A

Sample Definitions and Report

Appendix A

Sample Definitions and Report

RECORD DEFINITIONS

CUST Domain: CUST-REC

```
DEFINE RECORD CUST-REC USING
01 CUST-REC.
   15 COMPANY-NAME PIC IS X(20)
      QUERY-NAME IS COMPANY.
   15 ADDRESS.
      20 STREET PIC IS X(15).
      20 CITY PIC IS X(10).
      20 S-Z.
         25 STATE PIC IS X(2).
         25 ZIP-CODE PIC IS X(5)
            QUERY-NAME IS ZIP.
;
```

CUSTOMERS Domain: CUSTOMERS-REC

```
DEFINE RECORD CUSTOMERS-REC USING
01 CUSTOMERS-REC.
   15 COMPANY-NAME PIC IS X(20)
      QUERY-NAME IS COMPANY.
   15 CONTACT-PERSON PIC IS X(15)
      QUERY-NAME IS CONTACT.
   15 ADDRESS.
      25 STREET PIC IS X(15).
      25 CITY PIC IS X(10).
      25 STATE PIC IS X(2).
      25 ZIP-CODE PIC IS X(5)
         QUERY-NAME IS ZIP.
```

(continued on next page)

```
    15 PHONE-NUMBER PIC IS 9(10)
       EDIT-STRING IS XXX-XXX-XXXX
       QUERY-NAME IS PHONE.
    15 SALES-THIS-YEAR PIC IS  S9(5)V99
       EDIT-STRING IS $$$,$$$.$$
       QUERY-NAME IS SALES.
    15 LAST-PURCHASE USAGE IS DATE
       EDIT-STRING IS DD-MMM-YY.
    15 LAST-CONTACT USAGE IS DATE
       EDIT-STRING IS  DD-MMM-YY
       QUERY-NAME IS LAST-C.
;
```

EDIT-TEST Domain: EDIT-TEST-REC

```
DEFINE RECORD EDIT-TEST-REC USING
01 EDIT-TEST-REC.
    03 ITEM-NUMBER PIC IS 9(6).
    03 QUANTITY PIC IS 9(5)
       EDIT-STRING IS Z(5).
    03 UNIT-PRICE PIC IS 99V99
       EDIT-STRING IS $$$.$$
       QUERY-NAME IS PRICE.
    03 TOTAL-VALUE EDIT-STRING IS $$$$,$$$.$$
       COMPUTED BY QUANTITY * PRICE.
    03 LAST-SALE USAGE IS DATE
       EDIT-STRING IS MMMBDDBYYYY.
;
```

EMPLOYEES and WORK-EMP Domains: EMPLOYEES-REC

```
DEFINE RECORD EMPLOYEES-REC USING
01 PERSON.
    05 ID                        PIC IS 9(5).
    05 EMPLOYEE-STATUS           PIC IS X(11)
                                 QUERY-NAME IS STATUS
                                 QUERY-HEADER IS "STATUS".
    05 FIRST-NAME                PIC IS X(10)
                                 QUERY-NAME IS F-NAME.
    05 LAST-NAME                 PIC IS X(10)
                                 QUERY-NAME IS L-NAME.
    05 DEPT                      PIC IS X(15).
    05 SALARY                    PIC IS 9(5)
                                 EDIT-STRING IS $*$,$$$.
;
```

EVALUATION Domain: EVAL-REC

```
DEFINE RECORD EVAL-REC USING
01 PERSON.
   03 LAST-NAME PIC IS X(10).
   03 FIRST-NAME PIC IS X(10).
   03 EVAL-DATE USAGE IS DATE.
   03 SUMMARY PIC X(100)
      EDIT-STRING IS T(25).
;
```

FAMILIES Domain: FAMILY

```
DEFINE RECORD FAMILY USING
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC IS X(10).
      06 MOTHER PIC IS X(10).
   03 NUMBER-KIDS PIC 99
      EDIT-STRING IS Z9.
    03 KIDS OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER-KIDS.
      06 EACH-KID.
         09 KID-NAME PIC IS X(10)
            QUERY-NAME IS KID.
         09 AGE PIC IS 99
            EDIT-STRING IS Z9.
;
```

FLAT-SUPPLIES Domain: FLAT-SUPPLIES-REC

```
DEFINE RECORD FLAT-SUPPLIES-REC USING
01 WHAT.
   05 ITEM PIC X(15).
   05 ORDER-UNIT PIC X(10).
   05 VENDORS.
      07 COMPANY PIC X(10).
      07 PRICE PIC 99V99
         EDIT-STRING IS $$$.$$.
      07 DISCOUNT.
         09 MIN-ORDER PIC 9(5)
            EDIT-STRING IS Z(5).
         09 PERCENT PIC 99
            EDIT-STRING IS Z9%.
         09 WE-PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
            EDIT-STRING IS $$$.$$.
;
```

INVENTORY Domain: INVENTORY-REC

```
DEFINE RECORD INVENTORY-REC USING
01 INVENTORY-REC.
   15 ITEM-NAME PIC IS X(10)
      QUERY-NAME IS NAME.
   15 ITEM-NUMBER PIC IS 9(10)
      QUERY-NAME IS NUM.
   15 ITEM-NUMBER-PARTS REDEFINES ITEM-NUMBER
      QUERY-NAME IS NUM-PARTS.
      17 PRODUCT-GROUP PIC 99.
      17 PRODUCT-YEAR PIC 99.
      17 ASSEMBLY-CODE PIC 9.
      17 SUB-ASSEMBLY PIC 9(5).
   15 ITEM-NUMBER-GROUPS REDEFINES ITEM-NUMBER
      QUERY-NAME IS NUM-GROUPS.
      17 PRODUCT-GROUP-ID PIC 9(4).
      17 PART-DETAIL-ID PIC 9(6).
   15 ON-HAND PIC IS S9(5)
      EDIT-STRING IS -Z(5).
   15 UNIT-PRICE PIC IS  S9(3)V99
      EDIT-STRING IS $$$$.$$
      QUERY-NAME IS COST.
   15 TOTAL-VALUE EDIT-STRING IS $$$,$$$.$$
      QUERY-NAME IS TOTAL
      COMPUTED BY ON-HAND * UNIT-PRICE.
;
```

V1.0

KIDS Domain: KIDS-REC

```
DEFINE RECORD KIDS-REC USING
01 KIDS.
   03 LAST-NAME PIC X(10).
   03 KIDS-NAMES OCCURS 3 TIMES.
         05 FIRST-NAME PIC X(10).
         05 MIDDLE-INITIAL PIC X(1).
         05 AGE PIC 99
            EDIT-STRING IS Z9.
         05 GRADE PIC 99
            EDIT-STRING IS Z9.
         05 AVG-GRADES OCCURS 5 TIMES.
            07 SUBJECT PIC X(10).
            07 AVERAGE PIC 999
               EDIT-STRING IS Z99.
;
```

PARTS Domain: PART-REC

```
DEFINE RECORD PART-REC USING
01 PART-RECORD.
   05 PART-NAME PIC X(10).
   05 PART-NUMBER PIC 9(10).
```

```
   05 PART-NUMBER-PARTS REDEFINES PART-NUMBER.
      07 PRODUCT-GROUP PIC 99.
      07 PRODUCT-YEAR PIC 99.
      07 ASSEMBLY-CODE PIC 9.
      07 SUP-ASSEMBLY PIC 9(5).
   05 PART-NUMBER-GROUPS REDEFINES PART-NUMBER.
      07 PRODUCT-GROUP-ID PIC 9(4).
      07 PART-DETAIL-ID PIC 9(6).
;
```

PAY Domain: PAY-REC

```
DEFINE RECORD PAY-REC USING
01 PERSON.
   03 WHOLE-NAME QUERY-NAME IS NAME.
      05 LAST-NAME PIC IS X(10)
         QUERY-NAME IS LAST.
      05 FIRST-NAME PIC IS X(10)
         QUERY-NAME IS FIRST.
   03 WAGE PIC IS 99V99
      EDIT-STRING IS $$$.$$.
   03 WEEK-ENDING USAGE IS DATE.
   03 HOURS PIC IS 99V99
      EDIT-STRING IS ZZ.ZZ.
   03 SALARY EDIT-STRING IS $$$$.$$
      COMPUTED BY WAGE * HOURS.
   03 DEPT PIC XXX.
   03 AREA EDIT-STRING IS T(25)
      COMPUTED BY DEPT VIA DEPT-TABLE.
;
```

PHONES Domain: PHONE-REC

```
DEFINE RECORD PHONE-REC USING
01 PHONE.
   03 NAME PIC X(20).
   03 NUMBER PIC 9(7)
      EDIT-STRING IS XXX-XXXX.
   03 AREA-CODE PIC XXX VALID IF AREA-CODE IN AREA-TABLE.
;
```

REPS Domain: REPS-REC

```
DEFINE RECORD REPS-REC USING
01 JOB.
   03 TERRITORY PIC X(15).
   03 NUMBER-IN-JOB PIC 9.
   03 REP OCCURS 0 TO 8 TIMES DEPENDING ON NUMBER-IN-JOB.
      05 LAST-NAME PIC X(10).
      05 FIRST-NAME PIC X(10).
      05 CUSTOMERS PIC X(15) OCCURS 5 TIMES.
;
```

SQUARES Domain: SQUARES-REC

```
DEFINE RECORD SQUARES-REC USING
01 SQUARES-REC.
   03 NUMBER PIC 99
      EDIT-STRING IS Z9.
   03 ITS-SQUARE PIC 9(4)
      EDIT-STRING IS Z(3)9.
;
```

STOCKS Domain: STOCKS-REC

```
DEFINE RECORD STOCKS-REC USING
01 STOCKS-REC.
   15 COMPANY PIC IS X(15).
   15 PURCHASE-DATE USAGE IS DATE
      EDIT-STRING IS DD-MMM-YY
      QUERY-NAME IS P-DATE.
   15 PURCHASE-PRICE PIC IS S9(3)V99
      EDIT-STRING IS $$$$.$$
      QUERY-NAME IS P-PRICE.
   15 SHARES PIC IS S9(5)
      EDIT-STRING IS -Z(5).
   15 CURRENT-PRICE PIC IS S9(3)V99
      EDIT-STRING IS $$$$.$$
      QUERY-NAME IS C-PRICE.
   15 CURRENT-VALUE PIC IS  S9(6)V99
      EDIT-STRING IS $$$$,$$$.$$
      QUERY-NAME IS C-VALUE.
;
```

SUPPLIERS Domain: SUPPLIERS-REC

```
DEFINE RECORD SUPPLIERS-REC USING
01 WHAT.
   03 ITEM PIC X(15).
   03 ORDER-UNIT PIC X(10).
   03 SUPPLIERS OCCURS 3 TIMES.
      07 COMPANY PIC X(10).
      07 PRICE PIC 99V99
         EDIT-STRING IS $$$.$$.
      07 DISCOUNT OCCURS 3 TIMES.
         09 MIN-ORDER PIC X(10).
         09 PERCENT PIC 99
            EDIT-STRING IS Z9%.
         09 WE-PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
            EDIT-STRING IS $$$.$$.
;
```

SUPPLIES Domain: SUPPLIES-REC

```
DEFINE RECORD SUPPLIES-REC USING
01 WHAT.
   05 ITEM PIC X(15).
   05 ORDER-UNIT PIC X(10).
   05 HOW-MANY PIC 9.
   05 VENDORS OCCURS 0 TO 10 TIMES DEPENDING ON HOW-MANY.
      07 COMPANY PIC X(10).
      07 PRICE PIC 99V99
         EDIT-STRING IS $$$.$$.
      07 DISCOUNT OCCURS 3 TIMES.
         09 MIN-ORDER PIC 9(5)
            EDIT-STRING IS Z(5).
         09 PERCENT PIC 99
            EDIT-STRING IS Z9%.
         09 WE-PAY COMPUTED BY PRICE - (PERCENT/100 * PRICE)
            EDIT-STRING IS $$$.$$.
;
```

TRANS Domain: TRANS-REC

```
DEFINE RECORD TRANS-REC USING
01 TRANS-REC.
   15 ITEM-NUMBER PIC IS 9(10)
      QUERY-NAME IS NUM
      VALID IF ITEM-NUMBER NE 0.
   15 TRANS-TYPE PIC IS X(1)
      VALID IF TYPE EQ "S" OR TYPE EQ "O"
      QUERY-NAME IS TYPE.
   15 TRANS-DATE USAGE IS DATE
      EDIT-STRING IS NN/DD/YY
      QUERY-NAME IS T-DATE.
   15 COMPANY-NAME PIC IS X(3)
      VALID IF COMPANY-NAME IN COMPANIES
      QUERY-NAME IS COMPANY.
   15 FLAG PIC IS S9(1)
      EDIT-STRING IS -Z(1)
      VALID IF FLAG EQ 0 OR FLAG EQ 1.
   15 QUANTITY PIC IS S9(5)
      EDIT-STRING IS -Z(5)
      VALID IF QUANTITY NE 0
      QUERY-NAME IS QUANT.
   15 UNIT-PRICE PIC IS S9(5)V99
      EDIT-STRING IS $$$,$$$.$$
      QUERY-NAME IS PRICE.
   15 TOTAL-TRANS EDIT-STRING IS $$$,$$$.$$
      QUERY-NAME IS T-TOTAL
      COMPUTED BY QUANTITY * UNIT-PRICE.
;
```

WORK Domain: WORK-REC

```
DEFINE RECORD WORK-REC USING
01 WORK-REC.
    05 ID PIC IS 9(5).
    05 EMPLOYEE-STATUS PIC IS X(11)
       QUERY-NAME IS STATUS
       QUERY-HEADER IS "STATUS".
    05 EMPLOYEE-NAME QUERY-NAME IS NAME.
       10 FIRST-NAME PIC IS X(10)
          QUERY-NAME IS F-NAME.
       10 LAST-NAME PIC IS X(10)
          QUERY-NAME IS L-NAME.
    05 NORMAL-NAME COMPUTED BY FIRST-NAME!!" "!LAST-NAME
       QUERY-NAME IS NORMAL.
    05 ALPHA-NAME COMPUTED BY LAST-NAME!!", "!FIRST-NAME
       QUERY-NAME IS ALPHA.
;
```

VIEW DOMAIN DEFINITIONS

CUSTOMERS View: SALES-VIEW

```
DEFINE DOMAIN SALES-VIEW OF CUSTOMERS USING
01 CUSTOMER OCCURS FOR CUSTOMERS WITH SALES NE 0.
   03 COMPANY FROM CUSTOMERS.
   03 LAST-CONTACT FROM CUSTOMERS.
   03 SALES FROM CUSTOMERS.
;
```

INVENTORY and TRANS View: INV-VIEW

```
DEFINE DOMAIN INV-VIEW OF INVENTORY, TRANS USING
01 INV-DATA OCCURS FOR INVENTORY.
   03 ITEM-NAME FROM INVENTORY.
   03 ON-HAND FROM INVENTORY.
   03 SALES-DATA OCCURS FOR TRANS WITH
      (ITEM-NUMBER EQ INVENTORY-REC.ITEM-NUMBER) AND
      (TRANS-DATE GE VIEW-DATE) AND
      (TYPE EQ "S") SORTED BY TRANS-DATE.
      05 TRANS-DATE FROM TRANS.
      05 QUANTITY FROM TRANS.
;
```

PROCEDURE DEFINITIONS

CLEAN-DATA

```
DEFINE PROCEDURE CLEAN-DATA
   DEFINE FILE FOR WORK-EMP KEY = ID
   READY WORK-EMP WRITE
   READY EMPLOYEES
   FOR A IN EMPLOYEES
   STORE WORK-EMP USING EMPLOYEES-REC = A.EMPLOYEES-REC
END-PROCEDURE
```

MAIL-FORMAT and MAIL-REPORT

```
DEFINE PROCEDURE MAIL-FORMAT
   PRINT SKIP 2, COL 10, COMPANY(-), COL 10,
         CONTACT(-), COL 10, STREET(-), COL 10,
         CITY|!", "|STATE|", "|ZIP
END-PROCEDURE

DEFINE PROCEDURE MAIL-REPORT
   REPORT CUSTOMERS
   SET REPORT-NAME = "MAILING LIST"
   SET COLUMNS-PAGE = 40
   SET NO DATE, NO NUMBER
   :MAIL-FORMAT
   END-REPORT
END-PROCEDURE
```

NEW-SALARIES

```
DEFINE PROCEDURE NEW-SALARIES
   DECLARE PLUS10 COMPUTED BY SALARY * 1.1
      EDIT-STRING IS $$$$,$$$.
   DECLARE PLUS15 COMPUTED BY SALARY * 1.15
      EDIT-STRING IS $$$$,$$$.
   READY EMPLOYEES
   REPORT EMPLOYEES
   SET NO DATE
   SET NO NUMBER
   PRINT LAST-NAME|!", "|FIRST-NAME ("NAME"),
      SALARY ("CURRENT"/"SALARY"), PLUS10, PLUS15
   END-REPORT
END-PROCEDURE
```

SALARIES

```
DEFINE PROCEDURE SALARIES
   READY EMPLOYEES
   FIND EMPLOYEES SORTED BY DEPT
   REPORT CURRENT
   SET COLUMNS-PAGE = 60
   SET REPORT-NAME = "SALARIES"/"BY DEPARTMENT"
   AT BOTTOM OF DEPT PRINT COL 5, DEPT,
      COL 20, COUNT ("NUMBER"/"EMPLOYEES"),
      COL 35, TOTAL SALARY ("TOTAL"/"SALARY") USING $$$,$$$,$$$,
      COL 50, AVERAGE SALARY ("AVERAGE"/"SALARY") USING $$$,$$$,$$$
   AT BOTTOM OF REPORT PRINT SKIP, COL 10,
      "* * * * * * * * * * * * * * * * * * * * * * * * * *",
      SKIP 2, COL 5, "ALL DEPARTMENTS:", COL 22, COUNT,
      COL 35, TOTAL SALARY USING $$$,$$$,$$$,
      COL 50, AVERAGE SALARY USING $$$,$$$,$$$
   END-REPORT
END-PROCEDURE
```

SALARY-INFO

```
DEFINE PROCEDURE SALARY-INFO
   READY EMPLOYEES
   FIND EMPLOYEES SORTED BY DEPT
   REPORT CURRENT ON *,"Report file spec - TI: for screen"
   SET COLUMNS-PAGE = 50
   SET REPORT-NAME = "SALARY SUMMARY"
   AT BOTTOM OF DEPT PRINT COL 10,
      DEPT!!" ("!COUNT!!")", SKIP 2,
         COL 12, "MAXIMUM SALARY:", COL 30,
            MAX SALARY (-) :MONEY-EDIT, SKIP,
         COL 12, "MINIMUM SALARY:", COL 30,
            MIN SALARY (-) :MONEY-EDIT, SKIP,
         COL 12, "AVERAGE SALARY:", COL 30,
            AVERAGE SALARY (-) :MONEY-EDIT, SKIP,
         COL 12, "TOTAL SALARY:", COL 30,
            TOTAL SALARY (-) :MONEY-EDIT, SKIP
   AT BOTTOM OF PAGE PRINT SKIP 2, COL 28,
      "CONFIDENTIAL", SKIP, COL 28, "**********"
   AT BOTTOM OF REPORT PRINT SKIP,
      COL 10, "* * * * * * * * * * * * * * * * * *", SKIP 2,
      COL 10, "SUMMARY ("!!COUNT!")", SKIP 2,
      COL 12, "MAXIMUM SALARY:", COL 30,
         MAX SALARY (-) :MONEY-EDIT, SKIP,
      COL 12, "MINIMUM SALARY:", COL 30,
         MIN SALARY (-) :MONEY-EDIT, SKIP,
      COL 12, "AVERAGE SALARY:", COL 30,
         AVERAGE SALARY (-) :MONEY-EDIT, SKIP,
      COL 12, "TOTAL SALARY:", COL 30,
         TOTAL SALARY (-) :MONEY-EDIT
    END-REPORT
END-PROCEDURE
```

TOTAL-SAL

```
DEFINE PROCEDURE TOTAL-SAL
   READY EMPLOYEES
   FIND EMPLOYEES SORTED BY DEPT
   REPORT CURRENT
   SET COLUMNS-PAGE = 70
   AT BOTTOM OF DEPT PRINT COL 1, DEPT, COL 15,
      MAX SALARY ("MAXIMUM"/"SALARY") :MONEY-EDIT, COL 28,
      MIN SALARY ("MINIMUM"/"SALARY") :MONEY-EDIT, COL 41,
      TOTAL SALARY ("TOTAL"/"SALARY")  :MONEY-EDIT, COL 54,
      AVERAGE SALARY ("AVERAGE"/"SALARY") :MONEY-EDIT
   END-REPORT
END-PROCEDURE
```

TRANS-WRITE

```
DEFINE PROCEDURE TRANS-WRITE
   READY TRANS WRITE
   READY INVENTORY
   SET ABORT
   DECLARE T-ITEM PIC 9(10),
   T-ITEM = *,"Item number - 10 digits"
   IF NOT ANY INVENTORY WITH ITEM-NUMBER = T-ITEM THEN
      ABORT "Bad number - check inventory codes"
   STORE TRANS USING
      BEGIN
         ITEM-NUMBER = T-ITEM
         TRANS-TYPE = *,"Type of transaction - S or O"
         TRANS-DATE = *,"Date - dd/mm/yy or TODAY"
         COMPANY = *,"Three-character company code"
         QUANTITY = *,"Quantity"
      END
END-PROCEDURE
```

UPDATE-MASTER

```
DEFINE PROCEDURE UPDATE-MASTER
   FINISH
   READY TRANS WRITE
   READY INVENTORY WRITE
   FOR A IN TRANS WITH FLAG EQ 0
      BEGIN
         FOR B IN INVENTORY WITH B.NUM EQ A.NUM
            BEGIN
               IF TYPE EQ "S" THEN
               MODIFY USING B.ON-HAND = B.ON-HAND - A.QUANTITY ELSE
               MODIFY USING B.ON-HAND = B.ON-HAND + A.QUANTITY
            END
         MODIFY USING A.FLAG = 1
      END
END-PROCEDURE
```

VIEW-OF-INV

```
DEFINE PROCEDURE VIEW-OF-INV
   DECLARE VIEW-DATE USAGE IS DATE.
   VIEW-DATE = *."What date for records in the view"
   FINISH
   READY INV-VIEW
   PRINT INV-VIEW
   FINISH
END-PROCEDURE
```

WRITE-PRICE

```
DEFINE PROCEDURE WRITE-PRICE
   FINISH
   READY TRANS WRITE
   READY INVENTORY
   FOR A IN INVENTORY
      FOR B IN TRANS WITH (B.NUM EQ A.NUM) AND
          (TOTAL-TRANS EQ 0)
      MODIFY USING B.UNIT-PRICE = A.UNIT-PRICE
END-PROCEDURE
```

EMP-REPORT PROCEDURE AND REPORT

```
DEFINE PROCEDURE EMP-REPORT
   READY EMPLOYEES
   DECLARE DATE-TODAY USAGE IS DATE.
   DATE-TODAY = "TODAY"
   FIND EMPLOYEES SORTED BY DEPT, LAST-NAME, FIRST-NAME
   REPORT CURRENT ON *,"Report file spec - TI: for terminal"
   SET LINES-PAGE = 50
   SET COLUMNS-PAGE = 70
   AT TOP OF REPORT PRINT SKIP 15, COL 20,
      "* * * * * * * * * * * * * * * * * * * * *",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "*", COL 32, "SALARY REPORT", COL 56, "*",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "*", COL 30, "FOR ALL EMPLOYEES", COL 56, "*",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "*", COL 33, DATE-TODAY, COL 56, "*",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "* * * * * * * * * * * * * * * * * * * *",
      SKIP 5, COL 24, "AN EQUAL OPPORTUNITY EMPLOYER",
      NEW-PAGE
   AT TOP OF PAGE PRINT REPORT-HEADER,
      SKIP, COL 20, "* * * * * * * * * * * * * * * * * * * *",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "*", COL 32, "SALARY REPORT:", COL 56, "*",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "*", COL 32, "CONFIDENTIAL", COL 56, "*",
      SKIP, COL 20, "*", COL 56, "*",
      SKIP, COL 20, "* * * * * * * * * * * * * * * * * * * *",
      SKIP 2
   AT TOP OF DEPT PRINT COL 30, DEPT (-),
      SKIP, COL 28, "---------------", SKIP
   PRINT COL 15, LAST-NAME||", "|FIRST-NAME ("EMPLOYEE NAME"),
      COL 40, STATUS, COL 55, SALARY
   AT BOTTOM OF DEPT PRINT SKIP,
      COL 30, "NUMBER OF EMPLOYEES:  "|COUNT, SKIP,
      COL 30, "TOTAL SALARIES:",
         TOTAL SALARY (-) USING $$$$,$$$,$$$ , SKIP,
      COL 30, "AVERAGE SALARY:",
         AVERAGE SALARY (-) USING $$$$,$$$,$$$ , SKIP 2,
      COL 20, "* * * * * * * * * * * * * * * * * * * *", SKIP
   AT BOTTOM OF PAGE PRINT SKIP, COL 28, "CONFIDENTIAL",
      SKIP, COL 28, "------------"

   AT BOTTOM OF REPORT PRINT NEW-PAGE, COL 10, "GRAND TOTALS:",
      COL 30, "NUMBER OF EMPLOYEES:  "|COUNT, SKIP,
      COL 30, "TOTAL SALARIES:",
         TOTAL SALARY (-) USING $$$$,$$$,$$$ , SKIP,
      COL 30, "AVERAGE SALARY:",
         AVERAGE SALARY (-) USING $$$$,$$$,$$$ , SKIP,
      COL 30, "MINIMUM SALARY:",
         MIN SALARY (-) USING $$$,$$$,$$$ , SKIP,
      COL 30, "MAXIMUM SALARY:",
         MAX SALARY (-) USING $$$,$$$,$$$
   END-REPORT
   FINISH EMPLOYEES
END-PROCEDURE
```

The following pages show a reduced copy of the report produced by the EMP-REPORT procedure, as printed on an LA50 printer.

```
* * * * * * * * * * * * * * * * * * * *
*                                       *
*            SALARY REPORT              *
*                                       *
*          FOR ALL EMPLOYEES            *
*                                       *
*             21-Jul-83                 *
*                                       *
* * * * * * * * * * * * * * * * * * * *

     AN EQUAL OPPORTUNITY EMPLOYER

          CONFIDENTIAL
          ------------
```

21-Jul-83
Page 2

```
* * * * * * * * * * * * * * * * * * *
*                                     *
*           SALARY REPORT:            *
*                                     *
*           CONFIDENTIAL              *
*                                     *
* * * * * * * * * * * * * * * * * * *
```

AGRICULTURE

SCHWEIK, THOMAS	TRAINEE	$26,723

NUMBER OF EMPLOYEES:	1
TOTAL SALARIES:	$26,723
AVERAGE SALARY:	$26,723

* * * * * * * * * * * * * * * * * * * *

ASTRONOMY

BOOLE, GERALD	EXPERIENCED	$21,000
BOOLE, MARTHA	TRAINEE	$20,000
CHONTZ, NATHANIEL	TRAINEE	$24,502
HARRISON, LYDIA	EXPERIENCED	$40,747
SPIVA, CHARLOTTE	EXPERIENCED	$75,892

NUMBER OF EMPLOYEES:	5
TOTAL SALARIES:	$182,141
AVERAGE SALARY:	$36,428

* * * * * * * * * * * * * * * * * * * *

BIOLOGY

DONCHIKOV, BRUNO	EXPERIENCED	$35,952
IACOBONE, ANTHONY	EXPERIENCED	$58,462
PODERESIAN, RANDY	EXPERIENCED	$33,738
ROBERTS, DAN	EXPERIENCED	$41,395

NUMBER OF EMPLOYEES:	4
TOTAL SALARIES:	$169,547
AVERAGE SALARY:	$42,386

* * * * * * * * * * * * * * * * * * * *

CONFIDENTIAL

21-Jul-83
Page 3

```
* * * * * * * * * * * * * * * * * * *
*                                     *
*           SALARY REPORT:            *
*                                     *
*           CONFIDENTIAL              *
*                                     *
* * * * * * * * * * * * * * * * * * *
```

GEOGRAPHY

CARON, LESLIE	TRAINEE	$21,000

NUMBER OF EMPLOYEES:	1
TOTAL SALARIES:	$21,000
AVERAGE SALARY:	$21,000

* * * * * * * * * * * * * * * * * * * *

LITERATURE

CASSIDY, GAIL	EXPERIENCED	$55,407
FREIBURG, JOANNE	EXPERIENCED	$23,908
STONE, PAUL	EXPERIENCED	$32,000
SWAY, BILL	EXPERIENCED	$54,000
TASHKENT, JEFF	TRAINEE	$32,918
TERRY, CASS	EXPERIENCED	$29,908
WINSLOW, JOANNE	EXPERIENCED	$34,125

NUMBER OF EMPLOYEES:	7
TOTAL SALARIES:	$262,266
AVERAGE SALARY:	$37,466

* * * * * * * * * * * * * * * * * * * *

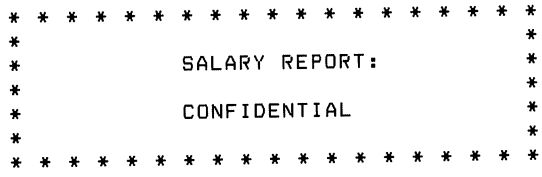
MATHEMATICS

BOOLE, GEORGE	EXPERIENCED	$20,000
BOOLE, JAMES	TRAINEE	$14,000
DION, REED	TRAINEE	$23,000
EINSTEIN, ALBERT	EXPERIENCED	$40,095
HOWL, FRED	EXPERIENCED	$59,594

CONFIDENTIAL

```
                                                    21-Jul-83
                                                    Page 4

      * * * * * * * * * * * * * * * * * * *
      *                                     *
      *           SALARY REPORT:            *
      *                                     *
      *           CONFIDENTIAL              *
      *                                     *
      * * * * * * * * * * * * * * * * * * *


JONES, AMY                  TRAINEE              $20,000
MEADER, JIM                 EXPERIENCED          $41,029
SMITH, ALLEN                TRAINEE              $20,000

              NUMBER OF EMPLOYEES:  8
              TOTAL SALARIES:                   $237,718
              AVERAGE SALARY:                    $29,714

     * * * * * * * * * * * * * * * * * * * *

              PHILOSOPHY
             ---------------

HAMMER, BART                TRAINEE              $26,392
KELLER, SY                  TRAINEE              $31,546
MORRISON, HANK              TRAINEE              $30,000
NALEVO, MARY                EXPERIENCED          $56,847
TERRICK, DEE                EXPERIENCED          $55,829

              NUMBER OF EMPLOYEES:  5
              TOTAL SALARIES:                   $200,614
              AVERAGE SALARY:                    $40,122

     * * * * * * * * * * * * * * * * * * * *

              PSYCHOLOGY
             ---------------

DEPALMA, LOUISE             EXPERIENCED          $57,598
LITELLA, DAVID              EXPERIENCED          $34,933
WITTGEN, STAN               TRAINEE              $25,023

              NUMBER OF EMPLOYEES:  3
              TOTAL SALARIES:                   $117,554
              AVERAGE SALARY:                    $39,184

     * * * * * * * * * * * * * * * * * * * *


             CONFIDENTIAL
             ------------
```

21-Jul-83
Page 5



GRAND TOTALS:	NUMBER OF EMPLOYEES:	34
	TOTAL SALARIES:	$1,217,563
	AVERAGE SALARY:	$35,810
	MINIMUM SALARY:	$14,000
	MAXIMUM SALARY:	$75,892

CONFIDENTIAL

B

PRO/DATATRIEVE Keywords

Appendix B

PRO/DATATRIEVE Keywords

ABORT
ADT
ADVANCED
ALL
ALLOCATION
AND
ANY
AS
ASC
ASCENDING
AT
AVERAGE
BEGIN
BETWEEN
BOTTOM
BT
BUT
BY
CHANGE
CHARACTER
CHARACTERS
CLOSE
COL
COLLECTIONS
COLUMN
COLUMNS_PAGE
COLUMN_HEADER
COMP
COMPUTED
COMP_1
COMP_2
COMP_3
COMP_5
COMP_6
CONNECT
CONT
CONTAINING
COUNT
DATATYPE
DATE
DECIMAL
DECLARE
DECREASING
DEFINE
DEFINEP
DELETE
DELETEP
DEPENDING
DESC
DESCENDING
DICTIONARY
DIGIT
DIGITS
DISPLAY
DOMAIN
DOMAINS
DOUBLE
DROP
DUP
D_FLOATING
EDIT
EDIT_STRING
ELSE
END
END_PROCEDURE
END_REPORT
END_TABLE
EQ
EQUAL
ERASE
EXCLUSIVE
EXIT
EXTEND
EXTRACT
FIELDS

FILE
FILL
FIND
FINISH
FIRST
FN1
FN2
FN3
FOR
FORM
FROM
F_FLOATING
GE
GREATER_EQUAL
GREATER_THAN
GT
GUIDE
HELP
IF
IN
INCREASING
INTEGER
IS
KEY
LAST
LE
LEADING
LEFT
LESS_EQUAL
LESS_THAN
LINES_PAGE
LONGWORD
LT
MATCH
MAX
MAX_LINES
MAX_PAGES
MEDIAN
MEMBERS
MIN
MODIFY
NE
NEW_PAGE
NEW_SECTION
NEXT
NO
NOT
NOT_EQUAL
NO_DATE
N0_NUMBER
NUMBER
NUMERIC
OCCURS
OF
ON
OPEN
OR
OVERPUNCHED
OWNER
PACKED
PAGE
PIC
PICTURE
PLOT
PORT
PRINT
PROCEDURE
PROCEDURES
PROMPT
PROTECTED
PW
QUADWORD
QUERY_HEADER
QUERY_NAME
READ
READY
REAL
RECORD
RECORDS
REDEFINES
RELEASE
REPEAT
REPORT
REPORT_HEADER
REPORT_NAME
RIGHT
SCALE
SELECT
SEPARATE
SET
SETS
SHARED
SHOW
SHOWP
SIGN
SIGNED
SIZE
SKIP
SORT
SORTED
SPACE
STORE
STRING
STRUCTURE
SUBSCHEMA
SUM
SUPERCEDE
SUPERSEDE
SYNC
TAB
TABLE
TABLES
TEXT
THE
THEN
TIMES
TO
TOP
TOTAL

TRAILING
UIC
UNSIGNED
USAGE
USING
VALID
VARYING
VERIFY
VIA
WHILE
WITH
WITHIN
WORD
WRITE
ZONED

PRODATATREVE Character Set and Sort Order

Appendix C

PRO/DATATRIEVE Character Set and Sort Order

Table C–1: ASCII Codes

DECIMAL VALUE	CHARACTER	REMARKS
32	SP	Space
33	!	Exclamation point
34	"	Double quotation mark
35	#	Number sign
36	$	Dollar sign
37	%	Percent sign
38	&	Ampersand
39	'	Apostrophe
40	(	Left (open) parenthesis
41	)	Right (close) parenthesis
42	*	Asterisk
43	+	Plus sign
44	,	Comma
45	-	Minus sign (hyphen)
46	.	Period (decimal point)
47	/	Slash (slant)
48	0	Zero
49	1	One
50	2	Two
51	3	Three
52	4	Four
53	5	Five

(continued on next page)

Table C–1: ASCII Codes (Cont.)

DECIMAL VALUE	CHARACTER	REMARKS
54	6	Six
55	7	Seven
56	8	Eight
57	9	Nine
58	:	Colon
59	;	Semicolon
60	<	Less than (left angle bracket)
61	=	Equal sign
62	>	Greater than (right angle bracket)
63	?	Question mark
64	@	Commercial at sign
65	A	Uppercase A
66	B	Uppercase B
67	C	Uppercase C
68	D	Uppercase D
69	E	Uppercase E
70	F	Uppercase F
71	G	Uppercase G
72	H	Uppercase H
73	I	Uppercase I
74	J	Uppercase J
75	K	Uppercase K
76	L	Uppercase L
77	M	Uppercase M
78	N	Uppercase N
79	O	Uppercase O
80	P	Uppercase P
81	Q	Uppercase Q
82	R	Uppercase R
83	S	Uppercase S
84	T	Uppercase T
85	U	Uppercase U
86	V	Uppercase V
87	W	Uppercase W
88	X	Uppercase X
89	Y	Uppercase Y
90	Z	Uppercase Z
91	[	Left square bracket
92	\	Backslash (reverse slant)
93	]	Right square bracket
94	^	Circumflex (caret)

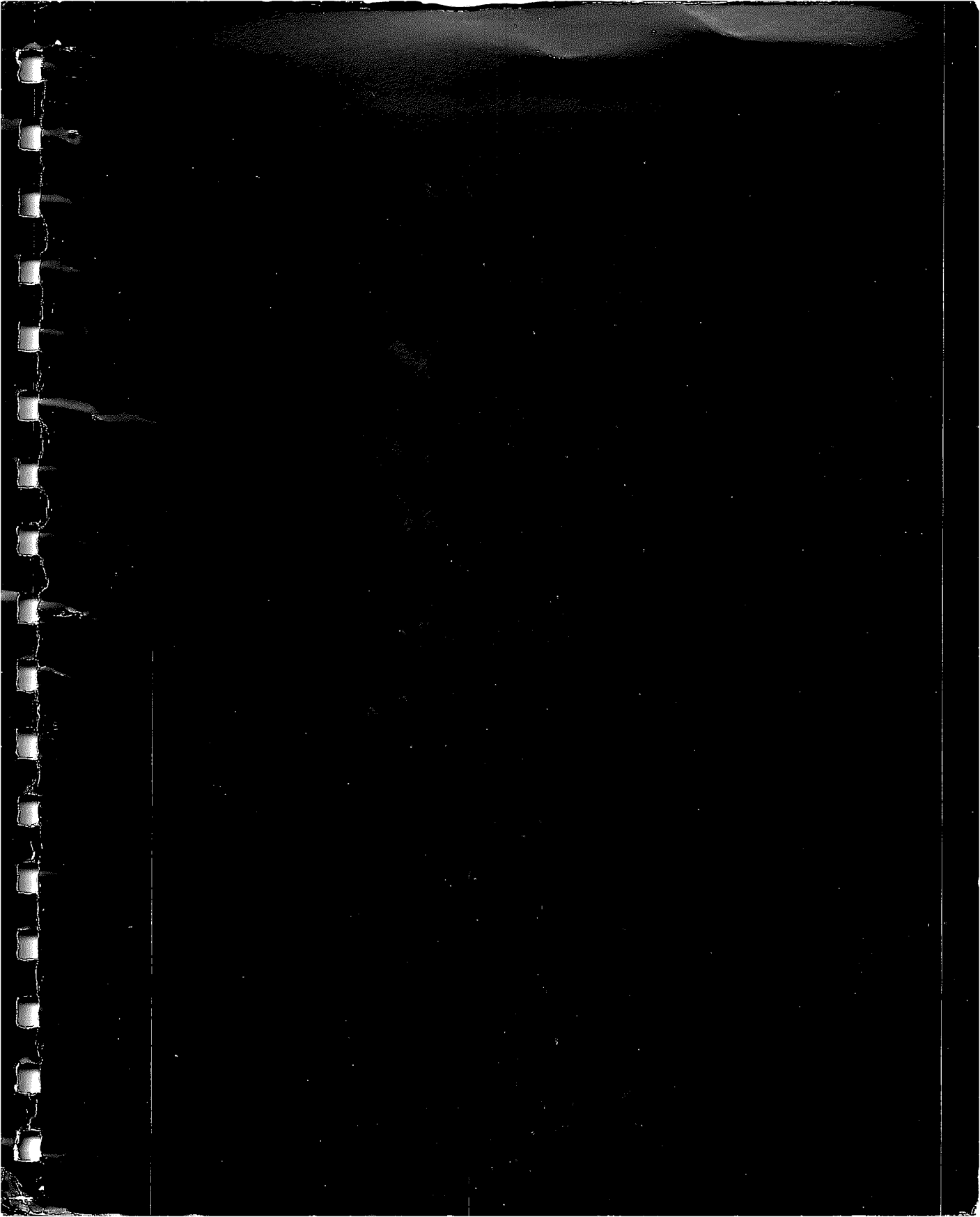
(continued on next page)

Table C–1: ASCII Codes (Cont.)

DECIMAL VALUE	CHARACTER	REMARKS
95	_	Underscore (underline)
96	`	Left single quote (grave accent)
97	a	Lowercase a
98	b	Lowercase b
99	c	Lowercase c
100	d	Lowercase d
101	e	Lowercase e
102	f	Lowercase f
103	g	Lowercase g
104	h	Lowercase h
105	i	Lowercase i
106	j	Lowercase j
107	k	Lowercase k
108	l	Lowercase l
109	m	Lowercase m
110	n	Lowercase n
111	o	Lowercase o
112	p	Lowercase p
113	q	Lowercase q
114	r	Lowercase r
115	s	Lowercase s
116	t	Lowercase t
117	u	Lowercase u
118	v	Lowercase v
119	w	Lowercase w
120	x	Lowercase x
121	y	Lowercase y
122	z	Lowercase z
123	{	Left brace
124	|	Vertical line
125	}	Right brace
126	~	Tilde

Index

Index



Printed in U.S.A.
AA-V440A-TH