July 1980

This manual describes DATATRIEVE-11 V2.0. It provides detailed reference information and describes how to use DATATRIEVE.

DATATRIEVE–11 V2.0
User's Guide

Order No. AA-C742B-TC

OPERATING SYSTEM AND VERSION:	RSX-11M	V3.2
	RSTS/E	V7.0
	IAS	V3.0
	RSX-11M-PLUS	V1.0
	VMS	V2.0
SOFTWARE VERSION:	Version 2.0	

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license, and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-paid READER'S COMMENTS form on the last page of this document requests your critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	FOCAL
DECnet	IAS
DECsystem-10	MASSBUS
DECSYSTEM-20	PDP
DECtape	RSX
DECUS	UNIBUS
DIBOL	VAX
DIGITAL	VMS

Contents

Chapter 6 Expressions

Chapter 7 Using the Report Writer

Chapter 8 Using the DATATRIEVE Editor

Chapter 9 Application Design Tool

Chapter 10 Creating and Maintaining Data Dictionaries

Chapter 11 Creating Record Definitions

Chapter 12 Procedures and Indirect Command Files

Chapter 13 Description Tables

Appendix A DATATRIEVE Error Messages

Appendix B Optimization Techniques

Appendix C Keywords

Appendix D Differences from Version 1.1

Appendix E Alignment of Fields

Appendix F DATATRIEVE Sorting Order

Glossary

Index

Figures

Tables

Commercial Engineering Publications typeset this manual using DIGITAL's TMS-11 Text Management System.

742ALL

Preface

Manual Objectives

This manual describes DATATRIEVE-11 Version 2.0 and provides usage and reference information on DATATRIEVE.

Intended Audience

This manual is intended for:

- Users who have read the *DATATRIEVE Primer*
- Advanced users who are responsible for more than data entry and update
- Applications programmers who are unfamiliar with DATATRIEVE

The *DATATRIEVE Primer* is a tutorial manual that introduces DATATRIEVE. New users of DATATRIEVE should read the *DATATRIEVE Primer* before using this manual.

Structure of This Manual

The first six chapters of this manual contain information of interest to all DATATRIEVE users. The remaining chapters explain specific DATATRIEVE features, such as description tables, and more advanced functions such as defining records.

Chapter 5 lists all DATATRIEVE commands and statements alphabetically. It is intended primarily as a reference section. Chapter 11 also contains a reference section, which lists the field definition clauses alphabetically.

Documentation Conventions

[ ]	Square brackets indicate that the enclosed item is optional. If there is more than possible choice, the possibilities are stacked vertically within the brackets.
{ }	Braces indicate that one of the items stacked within the braces must be included.
...	Ellipses, either vertical or horizontal, indicate that the preceding item in brackets may be repeated.
UPPERCASE	Uppercase indicates a DATATRIEVE language element. Inside a format, uppercase indicates keywords that you enter as shown.

lowercase — Lowercase inside a format indicates variable information that you supply.

(RET) — The symbol (RET) represents the non-printing carriage return key.

(CTRL/Z) — The symbol CTRL means that you must press the key labeled CTRL while pressing another key at the same time. Within examples a circumflex (^) is used to represent a control character. Thus ^Z is the same as CTRL/Z.

Chapter 1
Introduction to DATATRIEVE–11

DATATRIEVE–11 is an inquiry language and report writing system that provides direct, easy access to data contained in RMS files. DATATRIEVE operates under the IAS, RSTS/E, RSX–11, RSX–11M PLUS, and VMS operating systems, using RMS record management services.

DATATRIEVE lets you look at data, change it, or sort it interactively. You can add new records to a file, delete old ones, or modify existing ones to maintain an accurate, up-to-date file. In addition to its extensive facilities for file maintenance, DATATRIEVE provides report generation aids for creating simple or complex reports in a wide variety of formats.

DATATRIEVE requires no programming skills. Its simple-to-use interactive command language means that even novice users can learn to use DATATRIEVE easily and be productive in a short time.

1.1 A Full Command Language

You direct DATATRIEVE's operation using a series of commands. The command language is simple to use: each command name describes its function (such as MODIFY, SORT, and PRINT) and the syntax of each command is similar to that of English. For example, the following command requests that the data on all yachts with a price greater than $35,000 be displayed on the terminal:

```
PRINT ALL OF YACHTS WITH PRICE GREATER-THAN 35000
```

DATATRIEVE carries out each command as you enter it, so you see the results immediately. You can then continue by issuing more commands, correcting an error, or stopping. DATATRIEVE is forgiving of many common errors if your meaning is clear. And, if DATATRIEVE cannot interpret your instructions, it issues a clear, easy to understand error message. If you omit certain parts of a command, DATATRIEVE even prompts you for the missing information.

DATATRIEVE commands range from the simple, which can be invoked to produce immediate results, to the complex, which combine many functions in a single operation. Like a programming language, the DATATRIEVE command language allows you to group some commands in a loop for repeated execution. Commands which can be nested are called statements. But you do not have to be an experienced programmer to use DATATRIEVE. You can choose a number of different command sequences to produce the same results. The "right" solution to a DATATRIEVE problem is the solution that works.

Commands for retrieving records let you select just the data you want. You can retrieve all records in a file or only those that meet the selection criteria you specify. You can look at any field in a record or any combination of fields.

Commands for manipulating records let you sort them based on the contents of one or more fields. The records can be sorted in ascending or descending order. And, the data entry and modification statements let you store data in new records or change data in existing ones.

1.2 The Report Writer and PRINT

The DATATRIEVE Report Writer creates and prints reports in a variety of formats. You select not only the data and records to be included in the report, but also the report's title, headings, and if desired, summary lines. DATATRIEVE can calculate some common statistics (such as averages, totals, and high and low values) and include them where you specify. Every page in the report can be numbered and dated. You can display the report immediately on your terminal, print it on a line printer, or store it in a file (for printing at a later time).

You can also produce simpler reports using a single DATATRIEVE statement: PRINT. While PRINT does not provide the full formatting and statistical capabilities of the Report Writer, it allows you to display selective data on your terminal. While you are updating files, you can use PRINT to display intermediate results or to verify the contents of records.

1.3 Procedures

At every installation, some sequences of commands and statements will be issued on a recurring basis. To avoid retyping the command sequence each time it is needed, you can create a procedure. A procedure is a sequence of commands stored under a procedure name. To execute the commands, you merely call out the procedure by its name. A procedure can save a great deal of time because you need not reenter commonly used commands or "reinvent" complex command sequences.

A procedure can be made available to all users at an installation or only an authorized few. Thus, by invoking a procedure created by someone else, even less experienced users can perform complex operations.

You can use the DATATRIEVE editor, a subset of the DEC Standard Editor, to modify procedures once they are stored. So, you can change a procedure as the needs of your installation change.

1.4 Description Tables

A description table is a group of code-and-description pairs you create and store under a description table name. Using the description table facility, you can store a code in a record, but when you print the record you can print a descriptive character string, and not the code itself. The character string is the description that you stored in the description table. Because the code can be a single character and its description a much longer character string, description tables can save a great deal of record space.

For example, you could create a description table containing the following code-and-description pair:

S: "Single, widowed, or divorced"

Then you could store the code S in a field containing marital status. If you print the field without referring to the description table, DATATRIEVE prints the letter S. But if you use the description table to refer to the field, DATATRIEVE prints the description "Single, widowed, or divorced".

Description tables can save the time involved in creating some complex command sequences. And, description tables can be expanded to accommodate new code-and-description pairs. The DATATRIEVE editor is available for modifying description tables.

1.5 Defining Records

DATATRIEVE uses data stored in an RMS file. The data file organization can be relative, indexed, or sequential.

DATATRIEVE provides automatic conversion between data types used in other languages. Files created in other languages can be used by DATATRIEVE and files created with DATATRIEVE can be accessed by other programs.

You define the format of the records in a data file with a series of COBOL-like statements. But you need not be familiar with COBOL to create record definitions for DATATRIEVE's use. Complete instructions for creating record definitions are included in Chapter 11 of this manual.

DATATRIEVE also provides an alternative method for defining records: the Application Design Tool (ADT). ADT is an easy-to-use part of DATATRIEVE that creates the data file and its record definition using your responses to its questions. ADT lets you create the necessary definitions quickly so that you can begin storing your data in the file.

1.6 Password Tables

DATATRIEVE provides for data protection, beyond the protection offered by your operating system, through password tables. A password table restricts the use of data, procedures, and description tables to authorized users only. Even authorized users can be prevented from performing highly sensitive

operations such as modifying data or deleting records. The system manager is responsible for overall system security, but all users can protect their own private data, procedures, and description tables with the password facility.

1.7 Hierarchies and Views

DATATRIEVE provides two ways of looking at data in a manner different from the way you normally see it: hierarchies and views. A hierarchy lets you define a record so that its data is structured in a tree-like fashion. A hierarchy arranges data in a way that shows the subordination of one item of data to another.

A view lets you look at data in related records that have different definitions. A view provides a "window" into selected records and selected data of these records, allowing you to work with them as if they were stored in the same file.

1.8 HELP, GUIDE Mode, and More

When you need help using DATATRIEVE, use the HELP command. On-line assistance with DATATRIEVE commands and some concepts is available as soon as you invoke DATATRIEVE and remains available throughout a session.

GUIDE Mode, available if you use a VT52 or VT100 terminal, prompts you for every operation you want to perform and leads you through a session. To start GUIDE Mode, use the command SET GUIDE.

DATATRIEVE's clear error messages and prompting for missing statement elements also help you form complete, syntactically correct statements. The sample data provided with DATATRIEVE gives you the chance to practice using DATATRIEVE before you use it on your own data.

Chapter 2
DATATRIEVE Concepts

DATATRIEVE comes with three groups of sample data:

- Data on yachts
- Data on the owners of yachts
- Data on families

Most of this manual uses the data on yachts to explain the use of DATATRIEVE. (The data on yacht owners and families is used only with the more advanced topics of hierarchies and views.) This chapter uses the yachts data to introduce some basic concepts essential to the use of DATATRIEVE.

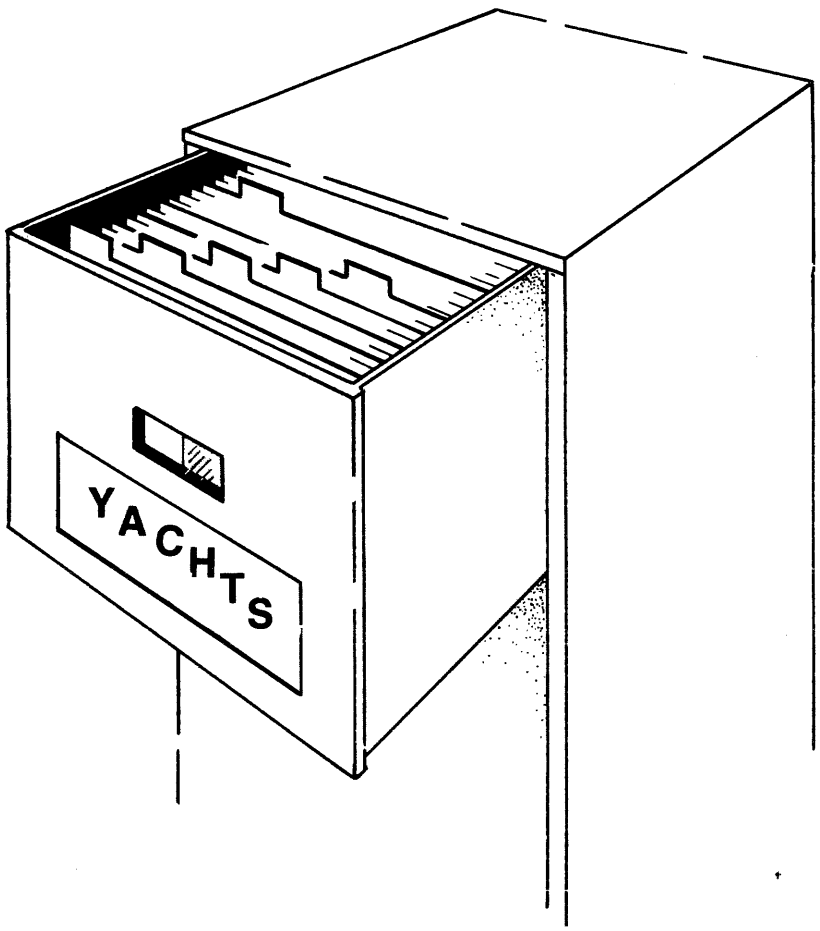
If you have read the *DATATRIEVE Primer*, many of the terms in this chapter will be familiar to you. This chapter is intended to supplement that information so you can perform operations more complex than those explained in the *DATATRIEVE Primer*. If you are an experienced DATATRIEVE user, you may want to go directly to the summary at the end of this chapter.

As you read this chapter, you can refer to the glossary at the back of this book to refresh your memory of some terms.

2.1 Records, Files, and Fields

The basis for any system that maintains data is a record. Just as a manual system for maintaining data consists of records, stored perhaps in a file folder and locked in a file drawer, DATATRIEVE maintains records that are stored in a file.

A file is any logically related group of data. For example, all the sample yacht data used in the examples in this manual are grouped into one file. The data in the file is stored in a series of records. A record consists of related data treated as a unit. For instance, all information for an individual yacht makes up that yacht's record.

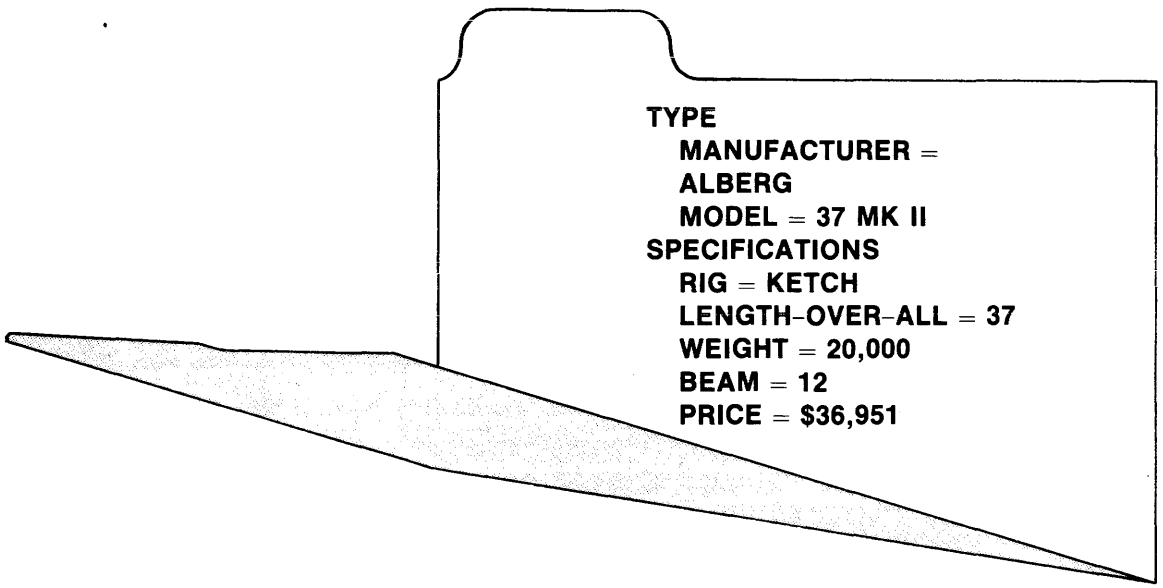


A record can contain any type of data: numbers (such as a price, a salary, or a part number), letters (such as the name of a person, company, or state), or any other combination of characters. Each item of data in a record is stored in a field. A record might contain only one field (that is, only one item of data); usually, though, a record contains many fields.

Each record for a yacht, for example, contains data on the following items:

- Manufacturer (or the builder of the yacht)
- Model
- Rig type (such as ketch or sloop)
- Length
- Weight
- Beam
- Price

Each of these data items is stored in its own field in a record. If you stored this information in a manual system, it might look like the following:



This simple record is, in fact, similar to the way DATATRIEVE stores data. Each field of data has a name (such as MANUFACTURER and MODEL). Some fields are grouped together because they are generally used together and are often referred to by a single name (such as TYPE). For instance, the type of yacht is determined by its builder's name and its model. Therefore, MANUFACTURER and MODEL are grouped under the name TYPE. The specifications for a yacht include its rig type, length, weight, beam, and price. The record lists the fields for this data under the SPECIFICATIONS field.

2.2 Domains

The sample yachts inventory contains over 100 records, each with the same categories of information. Figure 2–1 shows a portion of the sample data. Each line in that figure represents one record in the file. Every record contains the same fields.

Figure 2–1: Portion of Sample Yachts File

```
DTR> PRINT YACHTS
```

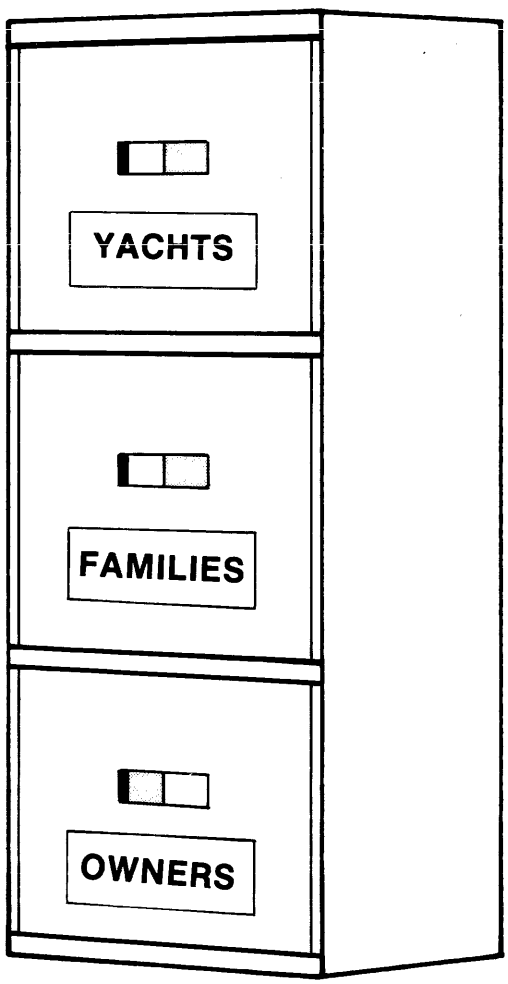
MANUFACTURER	MODEL	RIG	LENGTH OVER ALL	WEIGHT	BEAM	PRICE
ALBERG	37 MK II	KETCH	37	20,000	12	$36,000
ALBIN	79	SLOOP	26	4,200	10	$17,900
ALBIN	BALLAD	SLOOP	30	7,276	10	$27,500
ALBIN	VEGA	SLOOP	27	5,070	08	$18,600
AMERICAN	26	SLOOP	26	4,000	08	$9,890
AMERICAN	26-MS	MS	26	5,500	08	$18,890
BAYFIELD	30/32	SLOOP	32	9,500	10	$32,870
BLOCK I.	40	SLOOP	39	18,500	12	$29,050
BOMBAY	CLIPPER	SLOOP	31	9,400	11	$23,950
BUCCANEER	270	SLOOP	27	5,000	08	$11,500
BUCCANEER	320	SLOOP	32	12,500	10	$21,250
CABOT	36	SLOOP	36	15,000	12	$24,500
CAL	2-27	SLOOP	27	6,700	09	$13,710
CAL	2-34	SLOOP	33	9,500	10	$17,350
CAL	29	SLOOP	29	8,000	09	$15,400
CAL	3-30	SLOOP	30	10,500	10	$18,650
CAL	35	SLOOP	35	15,000	11	$24,500
CAPE DORY	25	SLOOP	25	4,000	07	$8,990
CAPE DORY	28	SLOOP	28	9,000	09	$21,990
CAPE DORY	TYPHOON	SLOOP	19	1,900	06	$4,290
CAPITAL	NEWPORT	SLOOP	28	7,000	09	$14,100
CARIBBEAN	35	SLOOP	35	18,000	11	$37,850
CHALLENGER	32	SLOOP	32	12,800	11	$31,830
CHALLENGER	35	SLOOP	35	14,800	12	$39,210
CHALLENGER	41	KETCH	41	26,700	13	$51,220
CHRIS-CRAF	CARIBBEAN	SLOOP	35	18,000	11	$37,850
CLIPPER	CM 30	SLOOP	30	3,800	08	$9,500
CLIPPER	CM 32	SLOOP	32	4,500	08	$12,950
COLUMBIA	35	SLOOP	35	11,350	10	$19,755
COLUMBIA	41	SLOOP	41	20,700	11	$48,490
COLUMBIA	PAYNE 9.6	SLOOP	32	10,200	10	$18,260
DOUGLAS	32	SLOOP	32	10,500	09	$18,650
DOWN EAST	32	SLOOP	32	15,000	11	$24,500
DOWN EAST	38	SLOOP	38	19,500	12	$30,350
DUFOUR	25	SLOOP	25	2,700	08	$8,510
EASTWARD	HO	MS	24	7,000	09	$15,900
ENCHILADA	20	SLOOP	20	2,300	07	$7,990
ENDEAVOUR	32	SLOOP	32	11,700	10	$20,210
ERICSON	23/ SPECIA	SLOOP	23	3,100	08	$9,030
ERICSON	CRUISING/3	SLOOP	36	16,000	12	$25,800
FISHER	30	KETCH	30	14,500	09	$23,850
FISHER	37	KETCH	37	30,000	12	$44,000
FJORD	MS 33	MS	33	14,000	11	$23,200

Of course, the data in any field may be different from one record to the next, but the type of information in a field is always the same: The MANUFACTURER field always contains the name of the builder; the PRICE field always contains a dollar amount.

The fields that a record contains and the type of information a field can hold is specified in the record's definition. Basically, a record definition lists all the fields in a record and indicates the type of data (such as numbers, letters, or other characters) that the field can contain.

All records with yachts data have the same record definition. As a group, the records form a DATATRIEVE domain. A domain is simply a complete set of records that have the same record definition. Thus, the sample DATATRIEVE data contains three domains:

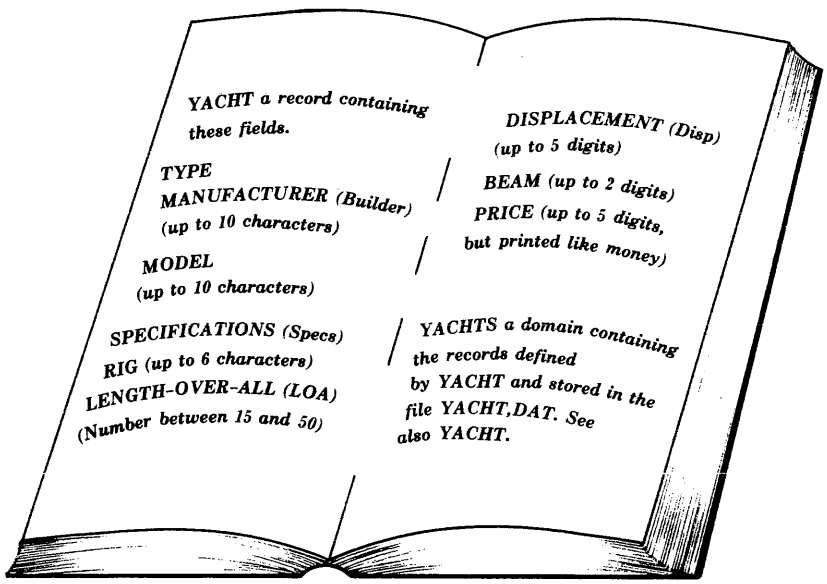
- YACHTS, which is the set of all records containing yacht data
- OWNERS, which is the set of all records containing yacht owner data
- FAMILIES, which is the set of all records containing family data



2.3 The Data Dictionary

Every domain has a name that uniquely identifies it. For instance, the domain containing all records of yachts is called YACHTS. To keep track of all domains and other pertinent information, DATATRIEVE maintains a data dictionary. Like a dictionary you use to look up the definition of a word, the data dictionary contains definitions. The definitions in the data dictionary, however, are not of words but of records, domains, procedures, and description tables. Records, domains, procedures, and description tables are called dictionary objects because their definitions are kept in the data dictionary.

We have seen that the definition of a record is similar to a list of field names and the type of data each field contains. This record definition is stored in the data dictionary under a record name. A domain definition, stored under its domain name, consists of its associated record definition and the name of the file containing its data. (The data is not stored in the data dictionary. Only the definition of its records is stored there. Data resides in a file outside of DATATRIEVE.)



When you invoke DATATRIEVE, you are automatically connected to a data dictionary. It is only through the data dictionary that you can gain access to records and their data. But, you may not be able to access every dictionary object.

Password tables, which are like locks on a file drawer, may prevent you from accessing some information. The person who creates a dictionary object determines who is allowed to access that dictionary object, and how they can use it. For example, you may be able to read the records in a domain but not change them. Password tables are also stored in the data dictionary.

2.4 The READY Command

If you invoke DATATRIEVE to work with records, generally the first thing you do is ready a domain. You ready a domain by issuing the READY command. READY signals your intention to use a domain and the records it contains. When you ready the domain you also indicate whether you will be reading records, modifying data in records, deleting records, or adding new records to the file.

2.5 Collections

A collection is a group of records from a domain, brought together for a specific purpose. A collection can contain all the records in a domain or just a few. For example, if you want to print the record of any yacht manufactured by Cape Dory, you could establish a collection of those records, then print them. Working with a small collection of records is more convenient and efficient than working with an entire domain.

The FIND statement establishes a collection. For instance, the following statement establishes a collection of all records for all yachts built by Cape Dory:

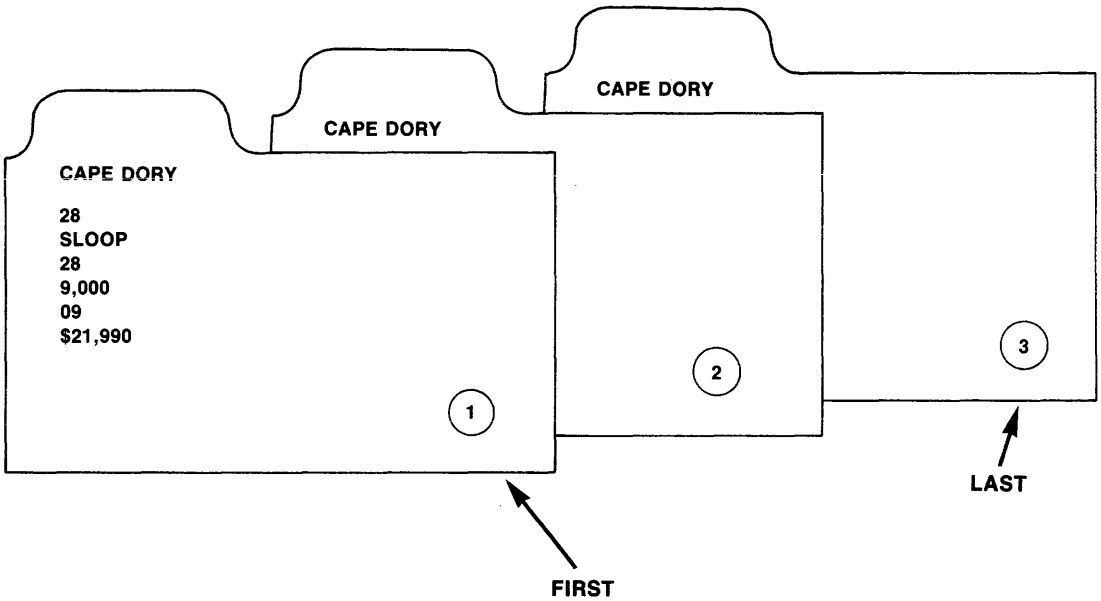
```
DTR> READY YACHTS READ
DTR> FIND YACHTS WITH MANUFACTURER="CAPE DORY"
[3 records found]
```

DATATRIEVE indicates that the collection contains three records.

Processing records generally includes sorting records, printing complete records (or just some fields in the records),and printing the whole collection or just selected records in the collection. To keep track of your place in the collection as you process it, DATATRIEVE provides a collection cursor. The collection cursor is a pointer that points to a single record in a collection. When you establish a collection with the FIND statement, the collection cursor is null. This means that the collection cursor does not point to any of the records. To make the collection cursor point to the record of your choice, use the SELECT statement.

As you would expect, the FIRST record of a collection is the first one encountered when the collection was established. The record that the collection cursor points to is the selected record. (There can be only one selected record at any one time.) The record following the selected record is the NEXT record in the collection. The LAST record is the one located at the end of the collection.

The following figure shows the collection of three records for Cape Dory. The circled numbers in the bottom right corner of each record indicate the record's position in the collection. These digits (like the words FIRST, NEXT, and LAST) allow you to refer to the specific record you want in the collection.



2.6 Record Streams

Like a collection, a record stream is a group of records from the same domain. A record stream is used only once, however. You create a record stream with a record selection expression. Only the command or statement containing the record selection expression uses the record stream. You can sort the record stream and specify conditions that the records must meet.

2.7 Summary

A record consists of related data treated as a whole. Each category of data in a record is stored in a field. The record definition lists the fields in the record and the type of data each field can contain. Related records are stored in a data file, which resides outside of DATATRIEVE.

A domain is a complete set of records with the same record definition. The domain definition lists the name of its associated record definition and the name of the data file containing the actual records.

Record and domain definitions (as well as procedure and description table definitions) are catalogued in the data dictionary. The dictionary also contains password tables that restrict the use of definitions to authorized users only.

Chapter 3
Sample Session

This chapter shows how to use DATATRIEVE to keep a telephone directory for a company. You start a DATATRIEVE session by invoking DATATRIEVE. On all operating systems except VMS, you invoke DATATRIEVE by typing the command DTR. On VMS type the command MCR DTR. DATATRIEVE responds with an identification message and a DTR> prompt.

```
>DTR
Datatrieve-11, DEC Query and Report System
Version: V02.00, 2-JUN-80
Type HELP for help
DTR>
```

If you cannot invoke DATATRIEVE with the command DTR (or MCR DTR on VAX/VMS) see your system manager. After invoking DATATRIEVE, define a domain for your telephone directory. Next, define the record and the data file that the domain will use.

```
DTR> DEFINE DOMAIN PHONES USING PHONE-REC ON PHONE.DAT;
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>           02 NAME PIC X(20).
DFN>           02 NUMBER PIC 9(7) EDIT-STRING IS XXX-XXXX.
DFN>           02 LOCATION PIC X(9).
DFN>           02 DEPARTMENT PIC XX.
DFN> ;
[Record PHONE-REC is 38 bytes long]
DTR> DEFINE FILE FOR PHONES KEY=NAME;
```

You may have noticed that the prompt changed to DFN> in the preceding example. DATATRIEVE issues the DFN> prompt whenever you are defining a record, domain, table, or procedure. When you finish a record definition DATATRIEVE tells you how long the record is and returns to DATATRIEVE command level.

The domain uses an indexed file, with the field NAME as the primary key. This means that the field NAME cannot have the same value in two records, and the value of NAME cannot be changed without erasing the record and storing a new one. The department is stored as a two-letter code. Now, create a description table to interpret the code.

```
DTR> DEFINE TABLE DEPT-TABLE
DFN> "CE":"Commercial Engineering",
DFN> "PE":"Plant Engineering",
DFN> "CS":"Customer Support",
DFN> "RD":"Research and Development",
DFN> "SD":"Sales Department",
DFN> ELSE "UNKNOWN DEPARTMENT"
DFN> END-TABLE
DTR>
```

To test the table, use it to print the description associated with a department code. You must put quotation marks around the code so that DATATRIEVE will recognize the code as a character string, and not try to interpret it as a field name. Also, to print out the entire description, you must use an edit-string. DATATRIEVE uses a default edit-string of X(10) for strings stored in tables, which means only the first 10 characters are printed unless you include an edit-string in the PRINT statement. (See Section 11.7 for more information on edit-strings.) The following examples show what happens if you leave out the quotation marks around the code or do not use an edit-string.

```
DTR> PRINT CE VIA DEPT-TABLE
Field "CE" is not defined or used out of context

DTR> PRINT "CE" VIA DEPT-TABLE
Commercial
```

The next example shows how to print the entire description.

```
DTR> PRINT "CE" VIA DEPT-TABLE USING X(25)
Commercial Engineering
```

To store information in the domain, use the STORE statement. You can prevent storing a nonexistent department code by using a VERIFY clause. Make the entire operation into a procedure which you can use any time.

```
DTR> READY PHONES WRITE
DTR> DEFINE PROCEDURE STORE-PHONE
DFN> STORE PHONES VERIFY USING
DFN> IF DEPARTMENT NOT IN DEPT-TABLE THEN
DFN> ABORT "BAD DEPARTMENT CODE"
DFN> END-PROCEDURE
DTR>
```

Because the procedure STORE–PHONE contains only one statement you can use it inside a REPEAT statement. This allows you to store as many records as you want with a single statement. After you type in the REPEAT statement DATATRIEVE prompts you for the data to be stored in each record.

The following example shows how DATATRIEVE reprompts for data if your original response was not in an acceptable format.

```
DTR> REPEAT 20 :STORE-PHONE
Enter NAME: Savage, Joseph
Enter NUMBER: 854-2744
Illegal placement of minus sign in string "854-2744"
Re-enter NUMBER: 8542744
Enter LOCATION: MB1-H5
Enter DEPARTMENT: RD
Enter NAME: Schneider, Henry
Enter NUMBER: 8542342
Enter LOCATION: MB1-H7
Enter DEPARTMENT: RD
Enter NAME: Krueger, William
Enter NUMBER: 8546919
Enter LOCATION: MB2-J5
Enter DEPARTMENT: CE
Enter NAME: Moore, Nancy
Enter NUMBER: 8545654
Enter LOCATION: MB1-J9
ENTER DEPARTMENT: CD
ABORT: BAD DEPARTMENT CODE
Execution terminated by "ABORT" Statement
```

At the end of the preceding example, DATATRIEVE stopped storing records because the VERIFY clause detected a bad department code. To check that DATATRIEVE did not store the record with the bad department code, use the PRINT command to see what records are in the domain PHONES.

```
DTR> PRINT ALL PHONES

        NAME          NUMBER  LOCATION DEPARTMENT

Krueger, William    854-6919 MB2-J5       CE
Savage, Joseph      854-2744 MB1-H5       RD
Schneider, Henry    854-2342 MB1-H7       RD
```

Instead of storing more records now, you may want to use the following DATATRIEVE statements on the records you have.

```
DTR> FOR PHONES PRINT NAME, NUMBER

        NAME          NUMBER

Krueger, William     854-6919
Savage, Joseph       854-2744
Schneider, Henry     854-2342

DTR> FIND PHONES
[3 records found]
DTR> SORT BY NUMBER
DTR> PRINT ALL

        NAME          NUMBER  LOCATION DEPARTMENT

Schneider, Henry    854-2342 MB1-H7       RD
Savage, Joseph      854-2744 MB1-H5       RD
Krueger, William    854-6919 MB2-J5       CE
```

(continued on next page)

```
DTR> PRINT MAX NUMBER

 NUMBER

854-6919

DTR> FOR CURRENT PRINT THEN MODIFY NUMBER

        NAME          NUMBER  LOCATION DEPARTMENT

Schneider, Henry     854-2342 MB1-H7       RD
Enter NUMBER: 8542341
Savage, Joseph       854-2744 MB1-H5       RD
Enter NUMBER: 8542744
Krueger, William     854-6919 MB2-J5       CE
Enter NUMBER: ^Z
Execution terminated by operator
DTR>
```

In the preceding example the CTRL/Z character caused DATATRIEVE to return to DATATRIEVE command level without modifying the last record. Whenever you are storing or modifying records and type CTRL/Z in response to an ENTER prompt DATATRIEVE will return you to command level. The next example shows how DATATRIEVE responds if you refer to a field without creating a "context."

```
DTR> FIND PHONES WITH NAME EQ "Schneider, Henry"
[1 Record found]
DTR> PRINT NUMBER
Field "NUMBER" is not defined or used out of context
```

Even though only one record is in the collection, there is no "selected" record. To create a context for the PRINT statement, you may either select the record or use the keyword ALL to refer to the whole collection.

```
DTR> PRINT ALL NUMBER


 NUMBER

854-2341
```

The following procedure produces a report on employees' locations and telephone numbers, with employees grouped by department.

```
DTR> DEFINE PROCEDURE REPORT-BY-DEPT
DFN> REPORT PHONES SORTED BY DEPARTMENT
DFN> SET REPORT-NAME = "COMPANY PHONE DIRECTORY"
DFN> AT TOP OF DEPARTMENT PRINT DEPARTMENT VIA DEPT-TABLE USING X(25)
DFN> PRINT NAME, NUMBER, LOCATION
DFN> END-REPORT
DFN> END-PROCEDURE
DTR>
DTR> :REPORT-BY-DEPT
```

(continued on next page)

```
                    COMPANY PHONE DIRECTORY

                                                          1-Apr-80
                                                            Page 1

      DEPARTMENT                NAME          NUMBER  LOCATION

Commercial Engineering
                        Krueger, William    854-6919    MB2-J5
Research and Development
                        Savage, Joseph      854-2744    MB1-H5
                        Schneider, Henry    854-2341    MB1-H7
```

To end a DATATRIEVE session, use the EXIT command. DATATRIEVE returns you to your operating system.

```
DTR> EXIT

>
```

Chapter 4
Introduction to Commands and Statements

Using DATATRIEVE interactively, you issue a series of commands and statements that direct DATATRIEVE's operation.

A command consists of a command name (such as READY or SHOW), and may include additional elements. A command can be issued only at the DATATRIEVE command level. Command level means that DATATRIEVE is not in the middle of a command or statement. The command level prompt is DTR> ; however, DATATRIEVE also issues this prompt when it is waiting for you to finish a command or statement. You cannot combine a command with another command or statement to form a compound command.

A statement consists of a statement name and may include additional elements. Unlike commands, however, statements can be nested within other statements to perform more complex operations. Statements can be repeated in a loop, and you can carry out statements conditionally by putting them within an IF-THEN-ELSE statement.

Statements are of two types: simple and compound. A simple statement is a single statement used alone. A compound statement is two statements combined with the word THEN or several statements placed within a BEGIN-END block. A compound statement can be used anywhere a simple statement is allowed. The format of a compound statement is:

```
statement-1 THEN statement-2
```

or

```
BEGIN
statement-1
statement-2
     .
     .
statement-n
END
```

DATATRIEVE executes the statements in the order in which they appear.

4.1 Command and Statement Elements

Every DATATRIEVE command and statement consists of one or more of the following elements:

- Command or statement name
- Other keywords
- Names, which identify an item containing a value
- Terminator, which signals the end of a command or statement
- Continuation character, which allows a command or statement to be continued on the next input line
- Comment, which allows you to enter text with a command or statement
- Expressions, which specify values or records

Every element of a DATATRIEVE command or statement must be constructed from characters in the DATATRIEVE character set.

The remainder of this chapter discusses the DATATRIEVE character set and all command and statement elements except expressions. Chapter 6 is devoted to a discussion of DATATRIEVE expressions.

4.2 Character Set

The DATATRIEVE character set consists of the upper- and lowercase letters A through Z, the digits 0 through 9, and special characters. Table 4–1 lists the characters in the DATATRIEVE set.

Table 4–1: DATATRIEVE Character Set

Uppercase Letters:
A through Z

Lowercase Letters:
a through z

Digits:
0 through 9

Special Characters:

.	(period)	"	(quotation mark)
,	(comma)	|	(vertical line)
;	(semicolon)	=	(equal sign)
:	(colon)	<	(less-than sign)
(	(left parenthesis)	>	(greater-than sign)
)	(right parenthesis)	$	(dollar sign)
[	(left bracket)	!	(exclamation point)
]	(right bracket)	_	(underscore)
+	(plus sign)	%	(percent sign)
-	(hyphen or minus sign)	?	(question mark)
*	(asterisk)	@	(at sign)
/	(slash)		(space)
RET	(Carriage return)	TAB	(tab)

NOTE

DATATRIEVE treats all letters that you enter as upper case unless you are storing data or enclose the letters in quotation marks.

4.3 Keywords

A keyword is a command or statement element that can be used only as indicated in the format of a command or statement. (Most keywords appear in commands or statements; however, some are used in record definitions and description tables.) In this manual, keywords are written in all uppercase letters in command and statement formats.

The use of keywords is restricted only to where shown in these formats; therefore, do not use a keyword as the name of a domain, record, field, description table, collection, procedure, variable, or view.

Appendix C contains a list of all DATATRIEVE keywords.

4.4 Names

A name is a character string used to identify any of the following items.

domain	collection
record	procedure
field	variable
description table	view

A name can consist of from 1 through 30 characters. The characters can be a letter, digit, hyphen(-), or underscore(_). The following restrictions apply to the construction of names:

1. A name must begin with a letter.
2. A name must end with a letter or digit.
3. A name cannot be a keyword.
4. A name must be 30 characters or less.

The following are all valid DATATRIEVE names:

TOTAL-SALARY

YACHTS

PRICE-PER-POUND

YEAR-TO-DATE-EARNINGS-FOR-1980

The following are all invalid DATATRIEVE names:

TOTAL
(Duplicates a keyword)

1980_EARNINGS
(Does not begin with a letter)

PRICE-PER-POUND($/LB)
(Contains illegal characters)

YEAR-TO-DATE_EARNINGS_FOR_FY_1980
(Contains too many characters)

4.5 Termination and Continuation Characters

A terminator signals the end of a command or statement. The formal terminator in DATATRIEVE is the semicolon (;). However, most commands and statements (except the DEFINE and DELETE commands) can be terminated by a carriage return (without a semicolon). You can enter multiple commands on a single input line by separating the commands with a semicolon. In this case, DATATRIEVE does not begin processing the commands on that line until you enter a carriage return.

If you enter a carriage return before a statement is logically complete, DATATRIEVE displays the following message on your terminal.

```
[LOOKING FOR element]
```

DATATRIEVE prompts for additional elements, if any, until the statement is logically complete. You can continue statements from one line to the next by entering carriage returns. However, DATATRIEVE does not prompt for an optional element if the line entered is already a complete statement. You can prevent DATATRIEVE from printing these statement prompts with the following command:

```
DTR> SET NO PROMPT
```

The continuation character hyphen (-) also lets you continue commands and statements on more than one input line. The hyphen lets you put optional elements on the next line, and DATATRIEVE does not issue any statement prompts between the two lines. You cannot continue a statement or command on more than two successive lines using a hyphen.

(The total number of characters for one input line may not exceed 132 characters.)

You must use a hyphen if you want to continue a literal expression or a name.

4.6 Comments

You can include a comment with a line of input by preceding the comment with an exclamation point (!) and terminating it with a carriage return. The comment can include any characters.

The primary use of comments is in indirect command files. During an interactive DATATRIEVE session, DATATRIEVE does not store comments, so they are of limited use.

Chapter 5
Commands and Statements

This chapter describes all DATATRIEVE commands and statements. Table 5–1 lists these commands and statements in alphabetical order. Table 5–2 lists several frequently performed functions and indicates the commands and statements you can issue to perform those functions.

The remainder of this chapter describes each command and statement, in alphabetical order. Before you issue a command or statement, you should read its description completely. Pay special attention to the following parts of the description:

- Format of the command or statement, including the spelling of keywords. As a rule, command and statement names and other keywords cannot be abbreviated. The sequence of command and statement elements is also critical. You must follow the sequence shown in the format. If you omit an optional element, just leave its relative position in the command or statement empty and proceed to the remaining elements in a left-to-right order. If necessary, review the documentation conventions listed in the front of this manual.
- Restrictions, which tell you the type of privilege you must have to issue the command or statement. A restriction could also indicate requirements for the correct operation of the command or statement.
- Usage Notes, which give you some common uses of the command or statement and its elements and indicates whether you can use other commands or statements with this command or statement.
- Examples, which show the command or statement syntax as well as some typical command and statement sequences.

Table 5-1: Alphabetical Summary of Commands and Statements

Section	Command/Statement (C = Command, S = Statement)	Function
5.1	ABORT (S)	Ends statement, procedure, or command file execution
5.2	ADT (C)	Invokes Application Design Tool (ADT)
5.3	ASSIGNMENT (S)	Establishes the value of a field or variable
5.4	BEGIN-END (S)	Groups statements into a compound statement
5.5	DECLARE (S)	Defines a variable
5.6	DEFINE DICTIONARY (C)	Creates a private data dictionary
5.7	DEFINE DOMAIN (C)	Creates a domain definition
5.8	DEFINE FILE (C)	Creates a data file for a domain
5.9	DEFINE PROCEDURE (C)	Creates a procedure definition
5.10	DEFINE RECORD (C)	Creates a record definition
5.11	DEFINE TABLE (C)	Creates a description table definition
5.12	DEFINEP (C)	Adds an entry to a password table
5.13	DELETE (C)	Removes a definition from a data dictionary
5.14	DELETEP (C)	Removes an entry from a password table
5.15	EDIT (C)	Invokes the DATATRIEVE editor
5.16	ERASE (S)	Deletes record(s) from a data file
5.17	EXIT (CTRL/Z) (C)	Ends a DATATRIEVE session
5.18	EXTRACT (C)	Copies a definition to an indirect command file
5.19	FIND (S)	Retrieves records and establishes current collection
5.20	FINISH (C)	Releases domain(s) and associated collections
5.21	FOR (S)	Applies statement(s) to selected records
5.22	HELP (C)	Provides online assistance with commands/statements
5.23	IF-THEN-ELSE (S)	Executes either of 2 statements, depending on evaluation of a Boolean expression
5.24	MODIFY (S)	Changes field(s) contents in specified record(s)
5.25	PRINT (S)	Prints specified field(s) of specified record(s)
5.26	READY (C)	Readies a domain for use
5.27	RELEASE (C)	Releases named collections, description tables, or global variables

(continued on next page)

Table 5–1: Alphabetical Summary of Commands and Statements (Cont.)

Section	Command/Statement (C = Command, S = Statement)	Function
5.28	REPEAT (S)	Executes a statement a specified number of times
5.29	REPORT (C)	Invokes the Report Writer
5.30	SELECT (S)	Selects a record from a collection
5.31	SET (C)	Establishes the current data dictionary, sets the maximum number of columns per page, controls the scope of aborts, starts Guide Mode, and inhibits or permits statement prompting
5.32	SHOW (C)	Prints information about the data dictionary and its contents
5.33	SHOWP (C)	Prints a password table
5.34	SORT (S)	Sorts records in a collection
5.35	STORE (S)	Stores a record in a domain
5.36	SUM (S)	Prints the total of contents of specified field(s)
5.37	THEN (S)	Creates a compound statement

Table 5-2: Summary of Commands and Statements by Function

Topic/Function	Command/Statement	Section
Online Assistance		
Printing command/statement information	HELP	5.22
Starting Guide Mode	SET GUIDE	5.31
Enabling/disabling statement prompting	SET PROMPT	5.31
Terminating a Session	EXIT (CTRL/Z)	5.17
Data Dictionaries		
Creating a data dictionary	DEFINE DICTIONARY	5.6
Establishing current dictionary	SET DICTIONARY	5.31
Displaying dictionary contents	SHOW ALL	5.32
Displaying dictionary file specification	SHOW DICTIONARY	5.32
Deleting a definition	DELETE	5.13
Domains		
Readying a domain for use	READY	5.26
Creating a domain definition	DEFINE DOMAIN	5.7
	ADT	5.2
Creating a data file for a domain	DEFINE FILE	5.8
Deleting a domain definition	DELETE	5.13
Copying a domain definition to an indirect command file	EXTRACT	5.18 5.20
Releasing control of a domain	FINISH	5.32
Displaying domain names or definitions	SHOW	
Records		
Creating a record definition	DEFINE RECORD	5.10
Deleting a record definition	ADT	5.2
Deleting a record from a file	DELETE	5.13
Copying a record definition to an indirect command file	ERASE	5.16
Changing field contents	EXTRACT	5.18
Printing records/field values	MODIFY	5.24
	PRINT	5.25
	REPORT	5.29
Adding records	SUM	5.36
Selecting records	STORE	5.35
Sorting records	SELECT	5.30
Displaying record names and definitions	SORT	5.34
	SHOW	5.32
Procedures		
Creating a procedure	DEFINE PROCEDURE	5.9
Deleting a procedure	DELETE	5.13
Modifying a procedure	EDIT	5.15
Copying a procedure to an indirect command file	EXTRACT	5.18
Displaying procedure names and definitions	SHOW	5.32

(continued on next page)

Table 5-2: Summary of Commands and Statements by Function (Cont.)

Topic/Function	Command/Statement	Section
Description Tables		
Creating a description table	DEFINE TABLE	5.11
Deleting a description table	DELETE	5.13
Modifying a description table	EDIT	5.15
Copying a description table to an indirect command file	EXTRACT	5.18
Releasing a description table	RELEASE	5.27
Displaying description table names and definitions	SHOW	5.32
Password Tables		
Creating a password table	DEFINE DOMAIN	5.7
for a domain	DEFINE PROCEDURE	5.9
for a procedure	DEFINE RECORD	5.10
for a record	DEFINE TABLE	5.11
for a description table	DEFINEP	5.12
Adding an entry to a password table	DELETEP	5.14
Deleting an entry from a password table	SHOWP	5.33
Displaying a password table		
Variables		
Defining a variable	DECLARE	5.5
Assigning a value to a variable	ASSIGNMENT	5.3.3
Releasing a variable	RELEASE	5.27
Combining/Repeating Statements		
Creating a compound statement	THEN	5.37
Creating a block of statements	BEGIN-END	5.4
Repeating a statement	REPEAT	5.28
Executing a statement conditionally	IF-THEN-ELSE	5.23
Applying statements to records	FOR	5.21

5.1 ABORT Statement

Function

Provides a means of stopping execution of either a single statement (during an interactive session) or an entire procedure or indirect command file.

Format

```
ABORT val-exp
```

Arguments

val-exp
: Is a value expression. If an abort occurs, DATATRIEVE prints a message containing this value expression.

Restrictions

None.

Results

1. If the abort conditions are met, DATATRIEVE prints the following message on the calling terminal:

   ```
   ABORT: val-exp
   Execution terminated by "ABORT" statement
   ```

2. If the abort occurs during an interactive session, DATATRIEVE stops executing the statement containing the ABORT statement and returns control to its command level. The ABORT statement does not end the interactive session.

3. If the abort occurs within a STORE or MODIFY statement the STORE or MODIFY is not carried out.

4. If the abort occurs in a procedure or indirect command file the results depend on whether SET ABORT is in effect. If SET ABORT is in effect DATATRIEVE returns to command level without executing the rest of the procedure or indirect command file. If SET NO ABORT is in effect DATATRIEVE executes any remaining statements and commands in the procedure or indirect command file. (At the start of a DATATRIEVE session SET NO ABORT is in effect. To change this setting use the SET ABORT command. See Section 5.31.)

Usage Notes

Use the ABORT statement within an IF-THEN-ELSE statement to abort if certain conditions are met.

Examples

1. Store a record in the YACHTS domain, but if the value of the BEAM field is 0, abort the operation and display a message.

```
DTR> STORE YACHTS VERIFY USING
[Looking for verification statement]
DTR> IF BEAM EQ 0 THEN ABORT "BAD VALUE FOR BEAM"
Enter MANUFACTURER: AMERICAN
Enter MODEL: 1980
Enter RIG: SLOOP
Enter LENGTH-OVER-ALL: 25
Enter BEAM: 0
Enter PRICE: 10000
ABORT: BAD VALUE FOR BEAM
Execution terminated by "ABORT" statement
DTR>
```

2. Define a procedure to write a report on the current collection. Abort the entire procedure if there is no current collection to report on.

```
DTR> DEFINE PROCEDURE YACHT-REPORT
DFN> SET ABORT
DFN> PRINT "THIS PROCEDURE REQUIRES A CURRENT COLLECTION"
DFN> IF *,"N to abort" CONTAINING "N" ABORT "TOO BAD"
DFN> REPORT
DFN> PRINT BOAT
DFN> AT BOTTOM OF REPORT PRINT COUNT, TOTAL PRICE
DFN> END-REPORT
DFN> END-PROCEDURE
DTR>
DTR> :YACHT-REPORT

THIS PROCEDURE REQUIRES A CURRENT COLLECTION
Enter N to abort: N
ABORT: TOO BAD
Execution terminated by "ABORT" statement
DTR>
```

5.2 ADT Command

Function

Invokes the Application Design Tool (ADT), which aids in the creation of a domain definition, its associated record definition, and its data file.

Format

ADT

Arguments

None.

Restrictions

ADT is an option when DATATRIEVE is installed. If DATATRIEVE is installed without ADT you cannot invoke ADT.

Results

DATATRIEVE invokes ADT, which asks you a series of questions about the domain and its field definitions. Refer to Chapter 11 for more information on ADT.

Example

Invoke the Application Design Tool.

```
DTR> ADT
Do you want help? (YES or NO) :
```

5.3 Assignment Statement

The assignment statement assigns a value to an elementary field, a group field, or a variable.

5.3.1 Assigning a Value to an Elementary Field

Function

Assigns a value to an elementary field in a MODIFY or STORE statement.

Format

```
fld-name = val-exp
```

Arguments

fld-name
Is the name of an elementary field.

=
Must be an equal sign, and *not* the relational operator EQ or EQUAL.

val-exp
Is the value to be assigned to the field. This argument can be any value expression.

Restrictions

1. This format of the assignment statement can be used only in a MODIFY...USING or STORE...USING statement. (See Section 5.25 or 5.36.)
2. To use the assignment statement inside a STORE...USING statement, you must ready the domain for WRITE or EXTEND access. To use this statement inside a MODIFY...USING statement, you must ready the domain for WRITE or MODIFY access. (See the READY command, Section 5.25.)
3. You cannot assign a value to a COMPUTED BY field.

Results

1. DATATRIEVE stores the value of the value expression in the specified field, performing any datatype conversions necessary.
2. If the argument val-exp is a prompting value expression (*.prompt-name) DATATRIEVE prompts for the value of the field. DATATRIEVE rejects your input and reprompts for the value if any of the following occurs:
   - Truncation error. You have entered more characters than the field definition allows.
   - Conversion error. You have entered a character that is inappropriate for the field, such as a letter for a numeric field.

- Sign error. You have entered a minus sign for an unsigned numeric field.
- VALID IF failure. You have entered an invalid value for the field. That is, the value results in a Boolean expression (specified in the record definition clause VALID IF bool-exp) that is "False." (See Section 11.16 for more information on record definitions.)

3. If the argument val-exp is not a prompting value expression and a truncation, conversion, or sign error occurs, DATATRIEVE accepts the value with a warning.
   - If you enter too many digits for a numeric field, DATATRIEVE truncates the high-order digits, stores the remaining digits in the field, and issues an error message.
   - If you enter too many characters for an alphanumeric field, DATATRIEVE truncates the rightmost characters, stores the remaining characters in the field, and issues an error message.
   - If a VALID IF failure occurs, DATATRIEVE does not execute the assignment statement. The STORE or MODIFY statement containing the assignment statement is not executed.

Usage Notes

1. When using this format of the assignment statement, you may want DATATRIEVE to check your values for validity and give you the opportunity to recover from an error caused by invalid input. The prompting value expression, shown in the assignment statement below, provides the only means for you to recover from a truncation, conversion, or sign error, or a VALID IF failure. This facility is especially useful when you are creating records with the STORE...USING statement.

```
fld-name = *.prompt-name
```

2. If you are unsure of the name of a field or its valid contents, use the SHOW FIELDS command (Section 5.33). SHOW FIELDS displays a brief description of each field in a readied domain.

Examples

1. Ready the YACHTS domain for WRITE access, then store a record in the domain for the manufacturer CHALLENGER.

```
DTR> READY YACHTS WRITE
DTR> STORE YACHTS USING MANUFACTURER="CHALLENGER"
DTR> PRINT YACHTS WITH MANUFACTURER EQ "CHALLENGER"

                                    LENGTH
                                     OVER
MANUFACTURER      MODEL     RIG      ALL       WEIGHT    BEAM PRICE

 CHALLENGER                                        0     00
 CHALLENGER       32       SLOOP    32         12,800    11 $31,835
 CHALLENGER       35       SLOOP    35         14,800    12 $39,215
 CHALLENGER       41       KETCH    41         26,700    13 $51,228
```

2. Ready the YACHTS domain for MODIFY access, then change the value of the PRICE field to a new value specified by the user.

```
DTR> READY YACHTS MODIFY
DTR> FIND YACHTS
[113 records found]
DTR> SELECT 1
DTR> PRINT PRICE THEN MODIFY USING PRICE=*,"NEW PRICE"

 PRICE

$36,951
Enter NEW PRICE: 39000
```

5.3.2 Assigning to a Group Field

Function

Assigns values to the elementary fields in a group field in a MODIFY or STORE statement.

Format

group-fld-name-1 = group-fld-name-2

Arguments

group-fld-name-1
Is the name of a group field.

=
Must be an equal sign, and not the Boolean operator EQ or EQUAL.

group-fld-name-2
Is the name of a group field.

Restrictions

1. This format of the assignment statement can be used only in a MODIFY...USING or STORE...USING statement. (See Section 5.25 or 5.36.)
2. To use the assignment statement in a MODIFY...USING statement, you must ready the domain containing group-fld-name-1 for MODIFY or WRITE access. To use it in a STORE...USING statement, you must ready the domain for WRITE or EXTEND access. (See the READY command, Section 5.27.)
3. Both group-fld-name-1 and group-fld-name-2 must have at least one elementary field with the same defined name or query name.

Results

DATATRIEVE changes the values of all fields in group-fld-name-1 to the values of identically named fields in group-fld-name-2. All other elementary fields in group-fld-1 are set to zero or blank in a STORE statement, or remain unchanged in a MODIFY statement.

Examples

1. Store a record in the YACHTS domain with the same values in the group field SPECIFICATIONS as the selected record.

```
DTR> READY YACHTS (SHHHH) WRITE
DTR> FIND YACHTS
[113 records found]
DTR> SELECT 1
DTR> STORE YACHTS USING BEGIN
[Looking for statement]
DTR> BUILDER=*.BUILDER
[Looking for statement or "end"]
DTR> SPECS=SPECS
[Looking for statement or "end"]
DTR> END
Enter BUILDER:
```

2. Store records into the new domain PRICED-YACHTS. Store only those yachts which have a price.

```
DTR> READY PRICED-YACHTS WRITE
DTR> FOR YACHTS WITH PRICE NE 0
[Looking for statement]
DTR> STORE PRICED-YACHTS USING BOAT=BOAT
DTR>
```

5.3.3 Assigning a Value to a Variable

Function

Assigns a value to a variable.

Format

variable-name = val-exp

Arguments

variable-name
: Is the name of a variable that has been defined with a DECLARE statement.

=
: Must be an equal sign, and *not* the relational operator EQ or EQUAL.

val-exp
: Is a value expression, the value of which is to be assigned to variable-name.

Restrictions

1. This format of the assignment statement can be used anywhere a statement is allowed.
2. The argument variable-name must be defined with the DECLARE statement (see Section 5.5).

Results

1. DATATRIEVE assigns the specified value to the variable.
2. If the argument val-exp is a prompting value expression (*.prompt-name) DATATRIEVE prompts for the value of the field. DATATRIEVE rejects your input and reprompts for the value if any of the following occurs:
   - Truncation error. You have entered more characters than the field definition allows.
   - Conversion error. You have entered a character that is inappropriate for the field, such as a letter for a numeric field.
   - Sign error. You have entered a minus sign for an unsigned numeric field.
   - VALID IF failure. You have entered an invalid value for the field. That is, the value results in a Boolean expression (specified in the record definition clause VALID IF bool-exp) that is "False." (See Section 11.16 for more information on record definitions.)
3. If the argument val-exp is not a prompting value expression and a truncation, conversion, or sign error occurs, DATATRIEVE accepts the value with a warning.
   - If you enter too many digits for a numeric field, DATATRIEVE truncates the high-order digits, stores the remaining digits in the field, and issues an error message.
   - If you enter too many characters for an alphanumeric field, DATATRIEVE truncates the rightmost characters, stores the remaining characters in the field, and issues an error message.
   - If a VALID IF failure occurs DATATRIEVE does not execute the assignment statement.

Usage Notes

When using this format of the assignment statement, you may want DATATRIEVE to check your values for validity and give you the opportunity to recover from an error caused by invalid input. The prompting value expression shown in the assignment statement below provides the only means for you to recover from a truncation, conversion, or sign error or VALID IF failure.

```
variable-name = *.prompt-name
```

Examples

1. Declare the global variable NEWPRICE. Assign a value to it.

```
DTR> DECLARE NEWPRICE PIC 99999 EDIT-STRING IS $$$,$$$.
DTR> NEWPRICE = $25000
```

2. Declare two global variables, then assign the value of OLDPRICE to NEWPRICE.

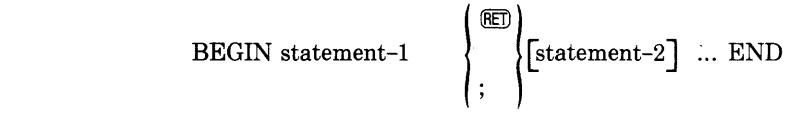
```
DTR> DECLARE NEWPRICE PIC 99999.
DTR> DECLARE OLDPRICE PIC 99999.
                .
                .
                .
DTR> NEWPRICE = OLDPRICE
```

5.4 BEGIN–END Statement

Function

Groups statements into a single compound statement, called a BEGIN–END block.

Format



Arguments

statement
Is a DATATRIEVE statement. If you enter more than one statement use carriage returns or semicolons to separate them.

END
Ends the BEGIN–END block.

Restrictions

1. You can use a BEGIN–END block anywhere you can use a statement.
2. You must observe any restriction on the statements included in the BEGIN–END block that is listed in their descriptions.
3. You cannot use a FIND or SELECT statement in a BEGIN–END block.

Results

1. DATATRIEVE executes the statements in the BEGIN–END block in the order in which you enter them.
2. If the BEGIN–END block includes a DECLARE statement, the variable defined by the DECLARE is a local variable and cannot be referenced outside the BEGIN–END block. The variable is automatically released when the END statement is executed. See Section 6.1.3 for more information on variables.

Examples

1. Store five records in the domain PHONES, each having LOCATION MB1-H2 and DEPARTMENT CE.

```
DTR> READY PHONES WRITE
DTR> REPEAT 5 STORE PHONES USING
[Looking for assignment statement(s)]
DTR> BEGIN
[Looking for statement]
DTR> NAME= *,NAME
[Looking for statement or "end"]
DTR> NUMBER= *,NUMBER
[Looking for statement or "end"]
DTR> LOCATION= "MB1-H2"
[Looking for statement or "end"]
DTR> DEPARTMENT= "CE"
[Looking for statement or "end"]
DTR> END
Enter NAME:
```

2. This example shows how a BEGIN-END block is treated as a single statement.

```
DTR> DEFINE PROCEDURE LOOP-EXAMPLE
DFN> PRINT "SHOW HOW A BEGIN-END WORKS WITH REPEAT"
DFN> PRINT "AND MORE THAN ONE STATEMENT"
DFN> END-PROCEDURE
DTR>
DTR> REPEAT 2 :LOOP-EXAMPLE
SHOW HOW A BEGIN-END WORKS WITH REPEAT
SHOW HOW A BEGIN-END WORKS WITH REPEAT

AND MORE THAN ONE STATEMENT

DTR>
DTR> REPEAT 2 BEGIN :LOOP-EXAMPLE END
SHOW HOW A BEGIN-END WORKS WITH REPEAT
AND MORE THAN ONE STATEMENT
SHOW HOW A BEGIN-END WORKS WITH REPEAT
AND MORE THAN ONE STATEMENT
```

5.5 DECLARE Statement

Function

Defines a global or local variable.

Format

```
DECLARE variable-name variable-defn.
```

Arguments

variable-name
: Is the name of the variable being defined. The name must conform to the rules for names listed in Section 4.5.

variable-defn
: Is the definition of the variable. The definition consists of field definition clauses. It must include at least a COMPUTED BY, PICTURE (PIC), or USAGE clause to define the type and size of the variable. (Refer to Section 11.6 for more information on field definition clauses.) If you include more than one clause, separate them with spaces.

. (period)
: Ends the DECLARE statement.

Restrictions

None.

Results

The DECLARE statement defines either a global variable or a local variable. A variable defined within a BEGIN–END block (see Section 5.4) is a local variable. You can reference a local variable only within the BEGIN–END block containing the DECLARE statement. DATATRIEVE releases local variables when it encounters the END statement of the BEGIN–END block.

If you are not inside a BEGIN–END block when you issue the DECLARE statement, DATATRIEVE creates a global variable. A global variable exists until you end a DATATRIEVE session (with the EXIT command or CTRL/Z) or until you release the variable with the RELEASE command.

Usage Notes

The DECLARE statement defines a variable; it does not assign a value to the variable. To assign a value to the variable, use the following format of the assignment statement:

variable-name = val-exp

See Section 5.3.2 for more information on assigning a value to a variable.

Examples

1. Declare the global variable NEWBEAM as a two-digit numeric field.

```
DTR> DECLARE NEWBEAM PIC 99.
```

2. Declare the variable NOW as a date. Assign today's date to NOW.

```
DTR> DECLARE NOW USAGE IS DATE.
DTR> NOW= "TODAY"
DTR> PRINT NOW(-)
22-May-80
```

5.6 DEFINE DICTIONARY Command

Function

Creates a file for DATATRIEVE to use as a data dictionary and connects you to the new dictionary.

Format

DEFINE DICTIONARY file-spec

Arguments

file-spec
Is the file specification for the data dictionary, in the format:

dev:[UFD]filename.ext;ver

You must specify at least one field. If you omit fields in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	QUERY
.ext	.DIC
;ver	1 or 1 higher than current version number (non–RSTS/E systems only)

Restrictions

You must have write access to the specified directory.

Results

1. DATATRIEVE creates an empty indexed file for use as a data dictionary.
2. If you do not enter file-spec on the same input line as DEFINE DICTIONARY, DATATRIEVE prompts with DFN>. After you enter the file specification, control returns to the DATATRIEVE command level.
3. The new data dictionary becomes the current data dictionary.

Usage Notes

1. You can verify the name of the new data dictionary with the following format of the SHOW command:

   SHOW DICTIONARY

   SHOW DICTIONARY prints the complete file specification of the current data dictionary.

2. DATATRIEVE does not automatically enter definitions in a private data dictionary. You can add definitions and associated password tables to the dictionary with the DEFINE DOMAIN, DEFINE PROCEDURE, DEFINE RECORD, and DEFINE TABLE commands.

Examples

1. Define a file named NEWDIC on the system device for use as a data dictionary. The file will have the same UFD as your default UFD and an extension of DIC.

```
DTR> DEFINE DICTIONARY NEWDIC
```

2. Define the file SY:[42,43]DATA.DIC for use as a data dictionary.

```
DTR> DEFINE DICTIONARY [42,43]DATA
```

5.7 DEFINE DOMAIN Command

The DEFINE DOMAIN command stores a domain definition in a data dictionary. The definition can be for an RMS domain or a view.

5.7.1 Defining an RMS Domain

Function

Stores a domain definition in the current data dictionary for an RMS domain and creates a password table for the domain.

Format

DEFINE DOMAIN domain-name USING record-name [(password) | (*)] ON file-spec ;

Arguments

domain-name
: Is the name of the domain being defined. It cannot duplicate a DATATRIEVE keyword or any other name in the current data dictionary.

record-name
: Is the name of the record definition to be associated with the domain. This record definition must be defined before you ready the domain.

password
: Is the password DATATRIEVE will use to check that you have E (execute) privilege for the record definition. Use (*) to have DATATRIEVE prompt you for the password.

file-spec
: Is the file specification of a file containing the data for the domain. This file does not have to exist until you try to ready the domain. The file specification has the following format:

```
dev:[UFD]filename.ext;ver
```

You must specify at least a file name. All other fields are optional.

; (semicolon)
: Ends the domain definition.

Restrictions

1. You cannot use a DEFINE DOMAIN command in a procedure.
2. You cannot invoke a procedure in a domain definition.

Results

1. If you do not end the DEFINE DOMAIN command with a semicolon, DATATRIEVE prompts with DFN>, indicating it is expecting a continuation of the domain definition. DATATRIEVE continues to prompt with DFN> until you enter a semicolon followed by a carriage return or until it detects an error. If a syntax error occurs while you are entering the domain definition, DATATRIEVE returns to command level without creating a domain definition.
2. DATATRIEVE enters the domain definition in the current data dictionary and creates a password table for the domain. DATATRIEVE enters a UIC/PPN in the password table with full access privileges (RWMEC). The actual UIC/PPN stored in the password table is installation-dependent. Refer to Chapter 14 for more information.

Usage Notes

1. The record or data file associated with the domain need not be defined when you issue a DEFINE DOMAIN command.
2. You cannot modify a domain definition with the DEFINE DOMAIN command. To change an existing domain definition, use the DATATRIEVE editor, explained in Chapter 8. You can also use the EXTRACT command to copy the domain definition to an indirect command file for editing. See Section 5.18 for more information on EXTRACT.

Examples

1. Define the domain PHONES. Use PHONE-REC as the associated record and store the data in the file PHONE.DAT.

```
DTR> DEFINE DOMAIN PHONES USING PHONE-REC ON PHONE.DAT;
```

2. Define the domain YACHTS. Use YACHT as the associated record and store the data in the file [202,202]YACHT.DAT.

```
DTR> DEFINE DOMAIN YACHTS USING YACHT ON [202,202]YACHT.DAT;
```

5.7.2 Defining a Domain as a View

Function

Stores a domain definition for a view in the current data dictionary and creates a password table for the domain.

Format

```
DEFINE DOMAIN view-name OF domain-name [,domain-name-2] ... USING

01      fld-name-1    OCCURS FOR rse-1 .

lev-no-m fld-name-n { OCCURS FOR rse-n  } .
                    { FROM domain-name-n }

                    .
                    .
                    .

;
```

Arguments

view-name
: Is the name of the view being defined. It cannot duplicate a DATATRIEVE keyword or any other name in the current data dictionary.

domain-name
: Is the name of a domain containing the records to be included in the view. You cannot use the name of a view. Use commas to separate domain names.

lev-no
: Is the level number for a field in the view.

fld-name
: Is the name of a field in the view. If fld-name is followed by a FROM domain-name clause, fld-name must be the name of a field in a domain-name specified in the OF clause.

OCCURS FOR rse
: Indicates that the associated field is to be included in the view for only those records that are identified by the record selection expression (RSE). The RSE must contain a reference to one of the domains specified in the OF clause.

FROM domain-name
: Indicates that the associated field is identical to the field of the same name in domain-name. The argument domain-name must be the same as the domain used in the last OCCURS FOR clause.

. (period)
: Ends a field definition.

; (semicolon)
: Ends the domain definition.

Restrictions

1. You cannot use a DEFINE DOMAIN command in a procedure.
2. You cannot invoke a procedure in a domain definition.

Results

1. If you do not end the DEFINE DOMAIN command with a semicolon, DATATRIEVE prompts with DFN>, indicating it is expecting a continuation of the domain definition. DATATRIEVE continues to prompt with DFN> until you enter a semicolon followed by a carriage return or until it detects an error. If a syntax error occurs while you are entering the domain definition, DATATRIEVE returns to command level without creating a domain definition.
2. DATATRIEVE enters the domain definition in the current data dictionary and creates a password table for the domain. DATATRIEVE enters a UIC/PPN in the password table with full access privileges (RWMEC). The actual UIC/PPN stored in the password table is installation-dependent. Refer to Chapter 14 for more information.

Usage Notes

Refer to Chapter 15 for more information on views.

Example

Define a view of big yachts.

```
DTR>  DEFINE DOMAIN BIGGEST-YACHTS OF YACHTS USING
DFN>  01 BIGGEST-YACHT  OCCURS FOR YACHTS WITH LOA GT 40.
DFN>  03 BUILDER        FROM YACHTS.
DFN>  03 MODEL          FROM YACHTS.
DFN>  03 PRICE          FROM YACHTS.
DFN>  ;
DTR>
```

See Chapter 15 for more examples.

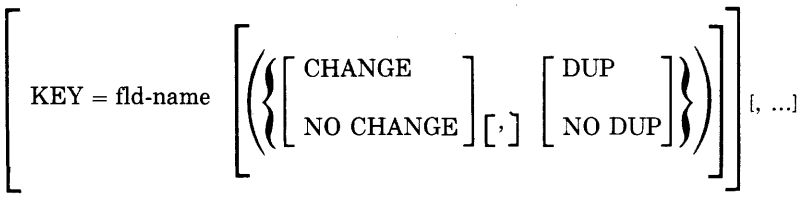
5.8 DEFINE FILE Command

Function

Creates a sequential or indexed sequential data file for an RMS domain defined in the current data dictionary. The data file has the same file specification as the domain definition.

Format

DEFINE FILE [FOR] domain-name [ALLOCATION = n] [SUPERSEDE,][MAX,]



Arguments

domain-name

Is the name of an RMS domain for which a file is to be created. The domain and its associated record must be in the current data dictionary when you issue this command.

ALLOCATION = n

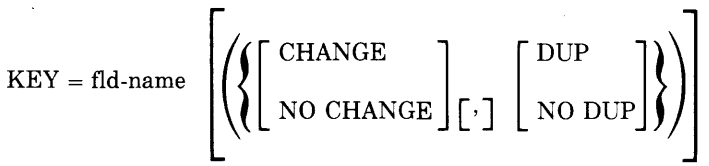
Specifies the number of disk blocks to be allocated for the file. The argument n must be an unsigned, nonzero integer, and not a value expression. If omitted, an allocation of 0 (zero) disk blocks is used.

SUPERSEDE

Indicates that any file with the same file specification as the domain's file specification should be deleted and replaced with a new file.

MAX

Indicates that DATATRIEVE should create a fixed-length record file for a domain with a record definition containing an OCCURS...DEPENDING clause. The length of a record in the file is the maximum possible size.



Indicates that the file is an indexed sequential file and specifies the field(s) to be used as key field(s). You can specify more than one key field; the first key you specify is the primary key; all others are alternate keys. If you specify more than one KEY clause, then the clauses must be separated by a comma (,). If you omit this clause, the file is sequential.

fld-name
Is the name of the field to be used as a key field.

(CHANGE)
(NO CHANGE)
Indicates if the associated key field can be modified. A primary key cannot be modified. See Tables 5–3 and 5–4 for defaults and allowed combinations.

(DUP)
(NO DUP)
Indicates if the associated key field can have a duplicate value; that is, if two or more records can have the same value for this field. See Tables 5–3 and 5–4 for defaults and allowed combinations.

Table 5–3: Default Values for KEY Fields

Default Values		
Option	**Primary Key**	**Alternate Key(s)**
CHANGE NO CHANGE	NO CHANGE	CHANGE
DUP NO DUP	NO DUP	DUP

Table 5–4: Allowed Combinations for KEY Fields

Combinations				
Key Type	**CHANGE + DUP**	**CHANGE + NO DUP**	**NO CHANGE + DUP**	**NO CHANGE + NO DUP**
Primary	Not Allowed	Not Allowed	Allowed	Allowed
Alternate	Allowed	Not Allowed	Allowed	Allowed

Restrictions

1. You must have control access privilege to the domain and execute access privilege to the record.

2. The domain and record must both be in the current data dictionary before you issue this command.

3. The domain must be an RMS domain, and not a view.

Results

1. DATATRIEVE prompts with DFN>, indicating it is expecting a definition. DATATRIEVE continues to prompt with DFN> until you enter a semicolon (;) followed by a carriage return or until it detects an error. If a syntax error occurs while you are entering the file definition, DATATRIEVE aborts the DEFINE FILE command and returns control to the DATATRIEVE command level. No file is created.

2. DATATRIEVE creates a sequential file or indexed sequential file. The file specification is the same as the file specification in the domain definition in the current data dictionary. If the domain definition omits a field in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	No default. Must be specified.
.ext	.DAT
;ver	1 or 1 higher than current version number (non–RSTS/E systems only)

3. The file is either fixed- or variable-length, depending on the record definition referenced in the domain definition, and on whether you include the MAX clause.

4. Under all operating systems but RSTS/E, if the file specification in the domain definition includes a version number and you specified the SUPERSEDE argument, the existing file is deleted and replaced with the new file. The new file has the same file specification (including version number) as the deleted file.

   RSTS/E does not support multiple versions of the same file. If there is already a file with the same file specification you must include the SUPERSEDE argument to define a new file. If you include the SUPERSEDE clause, DATATRIEVE deletes any file with the same file specification and replaces it with a new file for the domain.

Examples

1. Define an indexed file for the domain PHONES. Use the field NAME as the primary key.

```
DTR> DEFINE FILE FOR PHONES KEY= NAME;
```

2. Define a sequential file for the domain FAMILIES.

```
DTR> DEFINE FILE FOR FAMILIES;
```

3. Define a new indexed file for the domain YACHTS. Use the group field TYPE as the primary key, and allow duplicate values for this key. This command replaces the previous data file for YACHTS.

```
DTR> DEFINE FILE FOR YACHTS SUPERSEDE KEY=TYPE (DUP);
```

5.9 DEFINE PROCEDURE Command

Function

Enters a procedure in the current data dictionary and creates a password table for the procedure.

Format

```
DEFINE PROCEDURE proc-name
        .
        .
        .
END-PROCEDURE
```

Arguments

proc-name
Is the name of the procedure being defined. It cannot duplicate a DATATRIEVE keyword or any other name in the current data dictionary.

END-PROCEDURE
Ends the procedure definition.

Restrictions

The procedure cannot contain the following commands:

DEFINE DOMAIN

DEFINE PROCEDURE

DEFINE RECORD

DEFINE TABLE

Results

1. When you issue this command, DATATRIEVE prompts with DFN>, indicating that it is expecting a procedure definition; that is, the commands, statements, and clauses included in the procedure. DATATRIEVE treats all your input as input to the procedure and continues to prompt with DFN> until you end the procedure definition with END-PROCEDURE.
2. DATATRIEVE enters the procedure in the current data dictionary and creates a password table for the procedure. DATATRIEVE enters a UIC/PPN in the password table with full access privileges (RWMEC). The actual UIC/PPN stored on the password table is installation-dependent. Refer to Chapter 14 for more information.
3. DATATRIEVE does not store any comments specified within a procedure.

Usage Notes

1. For more information on procedures, see Chapter 13.
2. This command creates a procedure, but does not modify an existing procedure. To modify a procedure once it is stored in the data dictionary, use the DATATRIEVE editor, explained in Chapter 8. You can also use the EXTRACT command to copy the procedure definition to an indirect command file for editing. See Section 5.18 for more information on EXTRACT.

Examples

1. Define a procedure to procedure to prevent DATATRIEVE prompting you for missing statement elements, connect you to your private data dictionary, and then display the dictionary to which you are connected.

```
DTR> DEFINE PROCEDURE SET-UP
DFN> SET NO PROMPT
DFN> SET DICTIONARY MYOWN.DIC
DFN> SHOW DICTIONARY
DFN> END-PROCEDURE
DTR> :SET-UP
The current dictionary is SY:[202,202]MYOWN.DIC
DTR>
```

2. Define a procedure to write a report on all yachts with a given builder.

```
DTR> DEFINE PROCEDURE BUILDER-REPORT
DFN> READY YACHTS
DFN> REPORT YACHTS WITH BUILDER *.BUILDER ON *.DEVICE
DFN> SET REPORT-NAME = "REPORT OF YACHTS BY ONE BUILDER"
DFN> AT TOP OF REPORT PRINT REPORT-HEADER, BUILDER, SKIP2
DFN> PRINT MODEL, LOA, PRICE
DFN> AT BOTTOM OF REPORT PRINT "AVERAGE PRICE", AVERAGE PRICE
DFN> END-REPORT
DFN> END-PROCEDURE
DTR> :BUILDER-REPORT
Enter DEVICE: TI:
Enter BUILDER: AMERICAN

                    REPORT OF YACHTS BY ONE BUILDER
                                                          22-May-80
                                                             Page 1


                                AMERICAN

     26                              26          $9,895
     26-MS                           26         $18,895
     AVERAGE PRICE                              $14,395


DTR>
```

3. Define a procedure to simplify looking up phone numbers.

```
DTR> DEFINE PROCEDURE GET-NUMBER
DFN> READY PHONES
DFN> FOR PHONES WITH NAME CONTAINING *,"WHOSE NUMBER DO YOU NEED?"
DFN> PRINT NAME (-), NUMBER (-)
DFN> FINISH PHONES
DFN> END-PROCEDURE
DTR>
DTR>
DTR> :GET-NUMBER
Enter WHOSE NUMBER DO YOU NEED?: HENRY
Schneider, Henry      854-2341

DTR>
```

5.10 DEFINE RECORD Command

Function

Enters a record definition in the current data dictionary and creates a password table for the record.

Format

```
DEFINE RECORD record-name USING fld-def-1. [fld-def-2]...
;
```

Arguments

record-name
: Is the name of the record being defined. It cannot duplicate a DATATRIEVE keyword or any other name in the current data dictionary.

fld-def
: Is a field definition. A record definition consists of at least one field definition. Each field definition must be ended with a period (.). (Refer to Chapter 12 for a full description of record definitions.)

; (semicolon)
: Ends the record definition. You must enter the semicolon on a line by itself.

Restrictions

1. You cannot use a DEFINE RECORD command in a procedure.
2. You cannot invoke a procedure in a record definition.

Results

1. After you issue this command, DATATRIEVE prompts with DFN>, indicating it is expecting a record definition. DATATRIEVE continues to prompt with DFN> until you end the record definition with a line consisting of only a semicolon or until it detects an error. If a syntax error occurs while you are entering the record definition, DATATRIEVE returns to command level without creating a record definition.
2. When you terminate the record definition, DATATRIEVE prints the following message on your terminal, indicating the length (in bytes) of the new record.

```
[Record record-name is n bytes long]
```

3. DATATRIEVE enters the record definition in the current data dictionary and creates a password table for the record. DATATRIEVE enters a UIC/PPN in the password table with full access privileges (RWMEC). The actual UIC/PPN stored in the password table is installation-dependent. Refer to Chapter 14 for more information.

Usage Notes

This command creates a record, but does not modify an existing record. To modify a record once it is stored in the data dictionary, use the DATATRIEVE editor, explained in Chapter 8. You can also use the EXTRACT command to copy the record definition to an indirect command file forediting. See Section 5.18 for more information on EXTRACT.

If you change a record definition, you may not be able to use old data files. If the new record definition is not the same length, you will have to define a new data file.

Examples

1. Define the record PHONE-REC.

```
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>     02 NAME PIC X(20).
DFN>     02 NUMBER PIC 9(7) EDIT-STRING IS XXX-XXXX.
DFN>     02 LOCATION PIC X(9).
DFN>     02 DEPARTMENT PIC XX.
DFN> ;
[Record PHONE-REC is 38 bytes long.]
DTR>
```

2. Define the record FAMILY.

```
DTR> DEFINE RECORD FAMILY USING
DFN> 01 FAMILY.
DFN>  03 PARENTS.
DFN>   06 FATHER PIC X(10).
DFN>   06 MOTHER PIC X(10).
DFN>  03 NUMBER-KIDS PIC 99 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 X(10) QUERY-NAME IS KID.
DFN>      09 AGE PIC 99 EDIT-STRING IS Z9.
DFN> ;
```

5.11 DEFINE TABLE Command

Function

Enters a description table in the current data dictionary and creates a password table for the description table.

Format

```
DEFINE TABLE table-name
{"code-1" }: {"descrip-1" },
{ code-1  }  { descrip-1  }
[{"code-2" }: {"descrip-2" },]
{ code-2  }  { descrip-2  }
      .
      .
[ ELSE       {"descrip-n" }]
             { descrip-n  }
END-TABLE
```

Arguments

table-name
Is the name of the description table to be defined. It cannot duplicate a DATATRIEVE keyword or any other name in the current data dictionary.

{"code-1" }: {"descrip-1" }
{ code-1 } { descrip-1 }

Is a code-and-description pair. You must separate the pair with a colon, and you must put a comma after each pair but the last. If you include an ELSE, you must put a comma after the last pair. If the code or description conforms to the rules for names given in Section 4.4, you do not have to enclose it in quotation marks. However, DATATRIEVE will convert any lowercase letters in the code or description to uppercase.

If the code or description does not conform to these rules or if you want to preserve lowercase letters, you must enclose it in quotation marks and the rules for character string literals (Section 6.1.1) apply. (That is, the code or description cannot exceed 250 characters and cannot contain a carriage return, line feed, or control character.)

ELSE {"descrip-n" }
     { descrip-n }

Is the description to be used if you specify a code that is not defined in the description table. The rules for specifying this description are the same as for the description in the code-and-description pairs.

END-TABLE
Terminates the description table definition.

Restrictions

1. You cannot use a DEFINE TABLE command in a procedure.
2. You cannot invoke a procedure in a description table definition.

Results

1. After you issue this command, DATATRIEVE prompts with DFN>, indicating it expects a description table definition; that is, the code-and-description pairs. DATATRIEVE continues to prompt with DFN> until you terminate the description table definition with a line consisting of the keyword END-TABLE.
2. DATATRIEVE enters the description table in the current data dictionary and creates a password table for the table. DATATRIEVE enters a UIC/PPN in the password table with full access privilege (RWMEC). The actual UIC/PPN stored in the password table is installation-dependent. Refer to Chapter 14 for more information.

Usage Notes

1. For more information on description tables, see Chapter 13.
2. This command creates a description table but does not modify an existing table. To modify a description table once it is stored in the data dictionary, use the DATATRIEVE editor, explained in Chapter 8. Or, you can use the EXTRACT command to copy the description table definition to an indirect command file for editing. See Section 5.18 for more information on EXTRACT.

Examples

1. Define a table of department codes.

```
DTR> DEFINE TABLE DEPT-TABLE
DFN> "CE":"Commercial Engineering",
DFN> "PE": "Plant Engineering",
DFN> "CS": "Customer Support",
DFN> "RD": "Research and Development",
DFN> "SD": "Sales Department",
DFN> ELSE "UNKNOWN DEPARTMENT"
DFN> END-TABLE
DTR>
```

2. Define a table with a description of each possible rig.

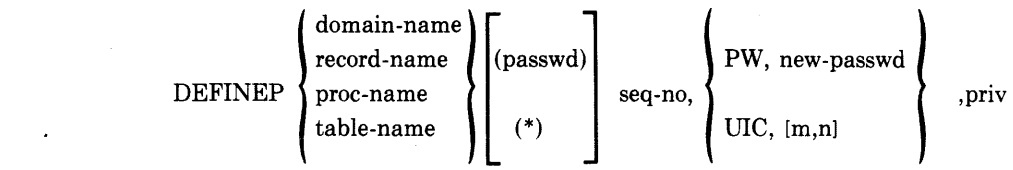
```
DTR> DEFINE TABLE RIG-TABLE
DFN> "SLOOP": "ONE MAST",
DFN> "KETCH": "TWO MASTS, BIG ONE IN FRONT",
DFN> "YAWL": "SIMILAR TO KETCH",
DFN> "MS": "SAILS AND A BIG MOTOR",
DFN> ELSE "SOMETHING ELSE"
DFN> END-TABLE
DTR>
```

5.12 DEFINEP Command

Function

Adds an entry to the password table for a domain, record, procedure, or description table.

Format



Arguments

domain-name
record-name
proc-name
table-name

Is the name of the domain, record, procedure, or description table whose password table you want to modify.

(passwd)
(*)

Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege.

seq-no

Is the sequence number of the entry to be added to the password table. This argument must be an unsigned integer and must be followed by a comma (,).

PW, new-password
UIC, [m,n]

Specifies the lock type and key for the entry. If the lock type is PW, specify a one- to ten-character password. If it is UIC, specify a UIC/PPN, enclosed in square brackets. You must put a comma after the lock type and another comma after the key.

priv

Is a letter or string of letters indicating the type of access privilege to be granted. You can also use a space enclosed in quotation marks to indicate that the user is not granted any access privileges.

Letter	Privilege
R	Read
E	Execute or Extend
M	Modify
W	Write
C	Control
" "	No privilege

You can specify more than one privilege by including a string of letters (such as RW for read and write access).

Restrictions

You must have control access privilege to the dictionary object.

Results

DATATRIEVE creates an entry in the password table. Depending on the sequence number you supply, DATATRIEVE may change the sequence number of other entries in the password table.

- If the sequence number already exists in the password table, DATATRIEVE adds the entry immediately before the existing entry with the same number. DATATRIEVE then increments the sequence number of all entries after the new entry by one.

- If the sequence number you supply is greater than the last sequence number plus one, DATATRIEVE ignores your sequence number and adds the entry to the end of the table. Its sequence number becomes the next sequential number in the table.

Usage Notes

1. To avoid errors when making an addition, print a copy of the current password table with the SHOWP command before issuing the DEFINEP command. See Section 5.34 for a description of the SHOWP command. Because of the way DATATRIEVE processes sequence numbers, a new entry can affect the numbering of other table entries.

2. For more information on password tables see Chapter 14.

Examples

1. Add an entry to the password table of PHONES show that everyone can show the domain definition. Look at the password table to see that the new entry is there.

```
DTR> SHOWP PHONES
       1,UIC, [202,*], "RWMEC"
DTR> DEFINEP PHONES 2,UIC,[*,*],"R"
DTR> SHOWP PHONES
       1,UIC, [202,*], "RWMEC"
       2,UIC, [*,*,] "R"
DTR>
```

2. Add an entry to the password table of PHONES to prohibit UIC/PPN [204,202] from accessing PHONES.

```
DTR> DEFINEP PHONES 1,UIC,[204,202]," "
DTR> SHOWP PHONES
       1,UIC, [204,202], " "
       2,UIC, [202,*], "RWMEC"
       3,UIC, [*,*], "R"
DTR>
```

5.13 DELETE Command

Function

Deletes the definition of a domain, procedure, record, or description table (and its associated password table) from the current data dictionary.

Format

```
DELETE { domain-name  }
       { record-name  }
       { proc-name    }  [ (passwd) ];
       { table-name   }  [   (*)    ]
```

Arguments

domain-name
record-name
proc-name
table-name
: Is the name of the domain, record, procedure, or description table to be deleted.

(passwd)
(*)
: Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege.

; (semicolon)
: Ends the DELETE command.

Restrictions

1. You must have C (control) access privilege to the domain, record, procedure, or description table.
2. The object must be defined in the current data dictionary.

Results

DATATRIEVE deletes only the definition of a domain or record from the current data dictionary. The data or records in the corresponding file are not affected by this command. If you specify a procedure or description table name in the DELETE command, DATATRIEVE deletes the procedure or table from the data dictionary. DATATRIEVE also deletes the password table associated with the dictionary object from the data dictionary.

Examples

1. Delete the domain PHONES and the record PHONE-REC.

```
DTR> DELETE PHONES;
DTR> DELETE PHONE-REC;
DTR>
```

2. Delete the table RIG-TABLE.

```
DTR> DELETE RIG-TABLE;
DTR>
```

5.14 DELETEP Command

Function

Deletes an entry from a password table associated with a domain, record, procedure, or description table, defined in the current data dictionary.

Format

```
                 ⎧ domain-name ⎫ ⎡          ⎤
DELETEP          ⎪ record-name ⎪ ⎢ (passwd) ⎥        seq-no
                 ⎪ proc-name   ⎪ ⎢          ⎥
                 ⎩ table-name  ⎭ ⎣ (*)      ⎦
```

Arguments

domain-name
record-name
proc-name
table-name

Is the name of the domain, record, procedure, or description table whose password table is to be modified.

(passwd)
(*)

Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege.

seq-no

Is the sequence number of the entry to be deleted. This argument must be an unsigned integer.

Restrictions

1. You must have C (control) access privilege to the domain, record, procedure, or description table.
2. You can delete only one password table entry each time you issue DELETEP. To delete more than one entry, you must issue multiple DELETEP commands.
3. There must be an entry with the sequence number you specify.
4. If there is only one entry in the password table, you cannot delete that entry.

Results

DATATRIEVE deletes the password table entry, as long as there is at least one other entry in the table.

Usage Notes

1. For more information on password tables, see Chapter 14.
2. To ensure that you delete the correct entry, print a copy of the password table using the SHOWP command (Section 5.33) before issuing DELETEP.
3. DATATRIEVE renumbers the remaining entries so that the sequence numbers begin at one and increase in steps of one.
4. If only one entry has C (control) access, be careful not to delete it.

Examples

1. Look at the password table for PHONES, and delete the first entry.

```
DTR> SHOWP PHONES
       1,UIC, [204,202], " "
       2,UIC, [202,*], "RWMEC"
       3,UIC, [*,*] "R"
DTR> DELETEP PHONES 1
DTR> SHOWP PHONES
       1,UIC, [202,*], "RWMEC"
       2,UIC, [*,*] "R"
```

2. Delete the entry which allows everyone R (read) access to PHONES.

```
DTR> DELETEP PHONES 2
DTR> SHOWP PHONES
       1,UIC, [202,*], "RWMEC"
DTR>
```

3. Attempt to delete the last entry in the password table.

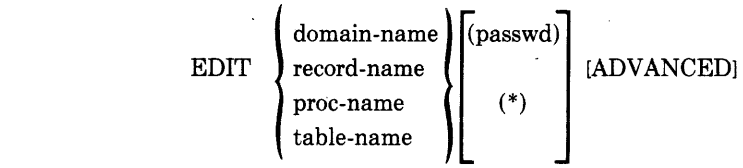
```
DTR> SHOWP PHONES
       1,UIC, [202,*], "RWMEC"
DTR> DELETEP PHONES 1
Attempt to delete last privilege entry
DTR> SHOWP PHONES
       1,UIC, [202,*], "RWMEC"
DTR>
```

5.15 EDIT Command

Function

Invokes the DATATRIEVE editor to edit a procedure or description table defined in the current data dictionary.

Format



Arguments

domain-name
record-name
proc-name
table-name

Is the name of the domain, record, procedure, or description table to be edited.

(passwd)
(*)

Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege.

ADVANCED

Must be included in the EDIT command if you are editing a domain or record definition.

Restrictions

You must have C (control) access privilege to the dictionary object.

Results

DATATRIEVE invokes its editor, which prompts for an editor command with QED>. To terminate the editor, type EXIT (or CTRL/Z), which updates the dictionary object, or QUIT, which leaves the dictionary object unchanged.

Usage Notes

1. Refer to Chapter 8 for a complete description of the DATATRIEVE editor.
2. Be when careful editing record definitions. You may not be able to ready the associated domain without defining a new file.
3. The DATATRIEVE editor does not check syntax. If you make syntax errors, you can end up with invalid definitions.

Examples

1. Edit the table DEPT-TABLE.

```
DTR> EDIT DEPT-TABLE
QED>
```

2. Edit the record PHONE-REC.

```
DTR> EDIT PHONE-REC ADVANCED
QED>
```

5.16 ERASE Statement

Function

Deletes one or more records from a data file.

Format

ERASE [ALL [OF rse]]

Arguments

ALL
Causes DATATRIEVE to delete every record in the current collection.

ALL OF rse
Causes DATATRIEVE to delete every record identified by the record selection expression.

Restrictions

1. The domain containing the record must be readied for WRITE access. (See the READY command, Section 5.26.)
2. The domain containing the record must use an indexed file. You cannot delete a record from a domain that uses a sequential file. You cannot delete a record from a view.

Results

1. If you use the argument ALL OF rse, each record in the record stream is deleted from the data file.
2. If you use the argument ALL, each record in the current collection is deleted from the data file.
3. If you nest the ERASE statement in a FOR statement and do not supply an argument, DATATRIEVE deletes each record specified by the FOR statement from the data file.
4. If you do not nest the ERASE statement in a FOR statement and do not supply an argument, DATATRIEVE deletes the selected record from the data file. If there is no selected record DATATRIEVE issues an error message.

Usage Notes

Before deleting a record with this command, you can check the current collection and selected record with the SHOW CURRENT command (Section 5.32) or the PRINT statement (Section 5.25).

Examples

1. Erase all the yachts built by Albin.

```
DTR> FIND YACHTS WITH BUILDER EQ "ALBIN"
[3 records found]
DTR> ERASE ALL
```

2. Define a procedure to erase selected yachts.

```
DTR> DEFINE PROCEDURE SELL-BOAT
DFN> FIND YACHTS WITH BUILDER EQ *.BUILDER AND
DFN> MODEL EQ *.MODEL
DFN> PRINT ALL
DFN> IF *."SELL?" EQ "Y" THEN ERASE ALL
DFN> END-PROCEDURE
DTR>
```

5.17 EXIT (CTRL/Z) Command

Function

Terminates a DATATRIEVE session.

Format

{EXIT / CTRL/Z}

Arguments

None.

Restrictions

None.

Results

EXIT stops a DATATRIEVE session and returns control to the operating system. DATATRIEVE automatically releases all readied domains and all established collections.

CTRL/Z acts as an EXIT command and stops a DATATRIEVE session if you enter it in response to the following prompts: DTR>, DFN>, and RW>. CTRL/Z does not stop a DATATRIEVE session if you are in ADT or the DATATRIEVE editor. CRTL/Z does not stop a session if you enter it when DATATRIEVE prompts you to enter a value.

Example

End a DATATRIEVE session.

```
DTR> EXIT
```

5.18 EXTRACT Command

Function

Copies a domain, record, procedure, or description table definition from the current data dictionary to an indirect command file.

Format

```
                                  { domain-name  }  [          ]
                                  { record-name  }  [ (passwd) ]  [, ...]
EXTRACT ON file-spec              { proc-name    }  [ (*)      ]
                                  { table-name   }  [          ]
```

Arguments

file-spec

Is the file specification of the file to contain the definition(s), in the following format:

dev:[UFD]filename.ext;ver

You must specify at least a file name. All other fields are optional. If you omit a field in the file specification, DATATRIEVE uses the following defaults.

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	No default. Must be specified.
.ext	.CMD
;ver	1 or 1 higher than current version number (non-RSTS/E systems only)

domain-name
record-name
proc-name
table-name

Is the name of the domain, record, procedure, or description table whose definition is to be copied. If you specify more than one name, separate the names with a comma.

Restrictions

You must have read access privilege to the domain, record, procedure, or description table.

Results

1. DATATRIEVE creates a file with the file specification you enter. For each dictionary object you name, DATATRIEVE writes a DELETE command and a DEFINE command into the file. The DEFINE command contains a copy of the definition in the current data dictionary.

2. DATATRIEVE does not delete the definition(s) from the data dictionary when you execute this command.

3. When you invoke the file as an indirect command file, DATATRIEVE deletes the definitions from the dictionary and replaces them with the definitions in the file.

Usage Notes

1. EXTRACT allows you to modify or restructure a data dictionary entry without affecting the data dictionary itself. You can first copy definitions to be modified, edit the definitions in the indirect command file using your operating system's editor, then invoke the command file to replace the old definitions.

2. EXTRACT is also useful for creating a backup copy of data dictionary entries and copying definitions to another data dictionary.

3. To invoke the indirect command file, type the following at the DATATRIEVE command level:

```
DTR> @file-spec
```

4. The EXTRACT command does not copy the password table of a dictionary object. After you invoke the indirect command file, the password table will have only one entry. To change this password table, use the DEFINEP and DELETEP commands.

Examples

1. Extract the definitions of the domain YACHTS and the record YACHT so that you can copy them into your private data dictionary.

```
DTR> EXTRACT ON YACHTS.CMD YACHTS, YACHT
DTR>
```

2. Copy all the definitions for the company phone directory to keep as a backup.

```
DTR> EXTRACT ON PHONE.CMD PHONES, PHONE-REC, DEPT-TABLE, STORE-PHONE
```

5.19 FIND Statement

Function

Establishes a collection of records from a domain, view, collection, or list. The collection established with FIND becomes the current collection, replacing any existing current collection.

Format

```
FIND rse
```

Arguments

rse
Is a record selection expression (RSE) specifying the records to be included in the collection.

Restrictions

1. The domain containing the records to be selected must be readied for READ, WRITE, or MODIFY access, but not EXTEND access. (See the READY command, Section 5.27.)
2. You cannot use a FIND and SELECT statement in the same compound statement.
3. You cannot use a FIND statement in a BEGIN–END block or a FOR statement.

Results

1. DATATRIEVE establishes a current collection. If you specify a context variable in the RSE, the collection has two names: CURRENT and the context variable.
2. DATATRIEVE displays the following message on your terminal, indicating the number of records in the collection:

   ```
   [n records found]
   ```

   If the FIND statement is inside a procedure, DATATRIEVE does not display the number of records found unless the FIND statement is the last statement of the procedure.
3. DATATRIEVE collects the records in the order in which they are found. Do not assume that there is any order to the collection unless you include the SORTED BY clause in the record selection expression, or unless the domain uses an indexed file.
4. There is no selected record in the collection. To select a record, use the SELECT statement.

Examples

1. Form a collection of yachts longer than 30 feet. Give the collection the name BIG-ONES.

```
DTR> FIND BIG-ONES IN YACHTS WITH LOA GT 30
[57 records found]
DTR>
```

2. Form a collection of the ten most expensive yachts.

```
DTR> FIND FIRST 10 YACHTS SORTED BY DESC PRICE
[10 records found]
DTR>
```

5.20 FINISH Command

Function

Relinquishes control of a domain and releases any collections associated with that domain.

Format

```
FINISH  domain-name-1 [,domain-name-2 ...]
```

Arguments

domain-name
: Is the name of a readied domain to be released. If omitted, DATATRIEVE ends your control of all readied domains. If more than one domain name is specified, they must be separated by a comma (,).

Restrictions

None.

Results

DATATRIEVE ends your control of the domain(s) and any collections you established with the domain(s) during this session.

Usage Notes

1. The FINISH command frees resources used by readied domains.
2. Readying a domain affects how other users can access it. Finishing a domain releases this control. If you ready a domain for exclusive use, no one else can ready that domain until you finish it.
3. It is not necessary to issue a FINISH command before an EXIT command. EXIT automatically releases all readied domains and collections.

Examples

1. Release control of the domain YACHTS.

```
DTR> FINISH YACHTS
```

2. Free more workspace by finishing all readied domains. Then ready YACHTS so that you can work with it.

```
DTR> FINISH
DTR> SHOW READY
No ready domains
DTR> READY YACHTS
DTR>
```

5.21 FOR Statement

Function

Executes a statement or group of statements once for each record in a group of records.

Format

FOR rse statement

Arguments

rse
: Is a record selection expression (RSE) specifying the records that the statement(s) are to act on.

statement
: Is a simple or compound statement (except a FIND or SELECT) to be applied to the records specified by the record selection expression.

Restrictions

1. The domain associated with the record selection expression must be readied for READ, WRITE, or MODIFY access.
2. A FOR statement cannot include a FIND or SELECT statement.

Results

For each record specified in the record selection expression, DATATRIEVE executes the statement once, using that record as the context record for the statement. If the statement is a BEGIN–END block, DATATRIEVE executes the statements of the BEGIN–END block in order for each record specified.

Examples

1. Increase the age of each child in each record of the domain FAMILIES by one year.

```
DTR> PRINT FIRST 1 FAMILIES

                          NUMBER     KID
  FATHER      MOTHER       KIDS      NAME     AGE

JIM           ANN           2        URSULA   7
                                     RALPH    3

DTR> FOR FAMILIES FOR KIDS MODIFY USING AGE = AGE + 1
DTR> PRINT FIRST 1 FAMILIES

                          NUMBER     KID
  FATHER      MOTHER       KIDS      NAME     AGE

JIM           ANN           2        URSULA   8
                                     RALPH    4
```

2. Assign a value to the field PRICE for all yachts that do not have a value in this field.

```
DTR> FIND YACHTS WITH PRICE = 0
[63 records found]
DTR> FOR CURRENT MODIFY USING PRICE = DISP * 1.3 + 5000
```

5.22 HELP Command

Function

Provides on-line assistance in the use of DATATRIEVE commands, statements, and language elements.

Format

HELP [[ADVANCED] [topic]...] ...

Arguments

ADVANCED
Provides additional information about the specified topic.

topic
Is a DATATRIEVE command, statement, or statement element. Use commas to separate topics.

NOTE

If no topic is specified, DATATRIEVE displays information on the type of help available and how to get it.

Restrictions

None.

Results

1. DATATRIEVE displays the requested help on your terminal.
2. If you specify HELP ADVANCED, DATATRIEVE displays the advanced help message for each topic listed.

Examples

1. Ask for a list of the help that is available.

```
DTR> HELP HELP
```

2. Ask for help in using the FIND command.

```
DTR> HELP FIND
```

3. Ask for the advanced help message for the PRINT statement.

```
DTR> HELP ADVANCED PRINT
```

5.23 IF–THEN–ELSE Statement

Function

Executes statements conditionally.

Format

IF bool-exp [THEN] statement-1 [ELSE statement-2]

Arguments

bool-exp
Is a Boolean expression.

statement-1
Is a simple or compound statement to be executed if bool-exp evaluates to "True."

ELSE statement-2
Specifies that statement-2 is to be executed if bool-exp evaluates to "False."

Restrictions

None.

Results

DATATRIEVE executes statement-1 if the Boolean expression is "True". If you specify ELSE statement-2 and the Boolean expression is "False" DATATRIEVE executes statement-2.

Example

```
Print each yacht built by Pearson, and modify the price if you want to.

DTR> FOR YACHTS WITH BUILDER EQ "PEARSON"
[Looking for statement]
DTR> BEGIN
[Looking for statement]
DTR> PRINT
[Looking for statement or "end"]
DTR> IF *."CHANGE EQ Y" EQ "Y" MODIFY PRICE ELSE PRINT "NOT CHANGED"
[Looking for statement or "end"]
DTR> END

                                LENGTH
                                 OVER
MANUFACTURER    MODEL     RIG    ALL   WEIGHT BEAM  PRICE

 PEARSON        10M       SLOOP    33  12,441  11   $21,173

Enter CHANGE EQ Y:
```

5.24 MODIFY Statement

Function

Changes the value of all fields or selected fields in any or all records in a collection.

Format

```
MODIFY [ALL] [ USING statement-1                    ] [VERIFY [USING] statement-2] [OF rse]
             [ fld-name-1 [,fld-name-2 ...]         ]
```

Arguments

ALL
Specifies that all records in the current collection are to be modified.

USING statement-1
Specifies a statement to be used to modify the field values in the record(s). Typically, statement-1 is an assignment statement or a BEGIN-END block containing a series of assignment statements.

fld-name
Specifies the name of the field to be modified. Use commas to separate field names. DATATRIEVE prompts for the value of each field you specify. VERIFY [USING] statement-2 causes DATATRIEVE to execute statement-2 before modifying the record. If DATATRIEVE executes an ABORT statement in statement-2, then the record is not changed. Typically, statement-2 contains an IF-THEN-ELSE statement and an ABORT statement. The keyword USING is optional and is included only for similarity to English language syntax.

OF rse
Is a record selection expression identifying the records to be modified.

Restrictions

1. The domain containing the records to be modified must be readied for WRITE or MODIFY access. (See the READY command, Section 5.26.)
2. In an RMS indexed sequential file, you cannot modify a primary key or any alternate key with the NO CHANGE option.
3. You cannot modify a COMPUTED BY field.

Results

The results of this command are highly dependent on the arguments you specify.

1. If you include a list of field names, DATATRIEVE prompts for each elementary field contained in them. If you do not include a list of field names or a USING clause, DATATRIEVE prompts for each elementary field in the record. DATATRIEVE prompts for field values with the following message:

```
ENTER fld-name:
```

2. If you specify a USING clause, DATATRIEVE uses statement–1 to modify the field values and does not prompt for any field values.

3. If you specify the argument ALL, DATATRIEVE modifies each record in the current collection similarly.. Unless you specify the USING clause, DATATRIEVE prompts for field values only once and uses the same field value in modifying each record. If you specify the USING clause, you can make similar changes in each record without storing identical field values in each record. (See Example 1.)

4. If you specify a record selection expression, DATATRIEVE modifies each record in the record stream similarly. Unless you specify the USING clause, DATATRIEVE prompts for field values only once and uses the same field value in modifying each record. If you specify the USING clause, you can make similar changes in each record without storing identical field values in each record. (See Example 2.)

5. If you nest MODIFY within a FOR statement, DATATRIEVE modifies the context record resulting from the FOR statement. When you want DATATRIEVE to prompt for field values for each record that it modifies, nest the MODIFY statement within a FOR statement.

6. If you do not nest the MODIFY statement within a FOR statement, and you do not specify the argument ALL or OF rse, DATATRIEVE modifies the selected record. If there is no selected record, DATATRIEVE issues an error message.

7. If you specify a VERIFY clause, DATATRIEVE executes statement–2 before changing the record. If statement–2 contains an ABORT statement and the abort conditions are met, DATATRIEVE aborts the MODIFY statement without changing the record. If the MODIFY statement is in a procedure or command file and SET NO ABORT is in effect, DATATRIEVE executes the next statement. If SET ABORT is in effect, DATATRIEVE returns to command level without executing the rest of the procedure or command file.

Usage Notes

1. If you specify the USING statement-1 clause, statement-1 will generally be an assignment statement or a BEGIN-END block containing assignment statements. The formats of the assignment statement are:

   fld-name = val-exp

   group-fld-name-1 = group-fld-name-2

   You should review the description of the assignment statement for elementary fields (Section 5.3.1) or group fields (Section 5.3.2) before using this format of the MODIFY statement.

   If you put a BEGIN-END block in the USING clause, you can include statements other than assignment statements. To see what you are modifying, include a PRINT statement in the BEGIN-END block. Section 5.4 describes the BEGIN-END block.

2. If DATATRIEVE prompts for a field value, you can perform any of the following functions in response to the prompt.

   - *Leave the value unchanged* by entering a tab character followed by a carriage return.
   - *Change the value* of the field by typing a new value, followed by a carriage return. The new value should conform to the data description for the field, as defined in the record definition.
   - *Change the value to blank* (or 0 if the field is numeric) by entering a space followed by a carriage return.
   - *Terminate the prompting cycle* by entering a CTRL/Z. In this case, the record being modified is unchanged, but previously modified records are changed. CTRL/Z has the same effect as an ABORT statement in this case.

   If you enter just a carriage return in response to a prompt, DATATRIEVE reprompts for the field value.

3. The MODIFY statement can be very dangerous if you make a mistake. If DATATRIEVE prompts you for an unexpected field, enter CTRL/Z to prevent changing records that you do not wish to change. It is safest to print each record before you modify it.

Examples

1. Form a collection of yachts with a specific builder, and increase the price of each yacht in the collection.

```
DTR> FIND YACHTS WITH BUILDER EQ "GRAMPIAN"
[5 records found]
DTR> PRINT ALL PRICE

 PRICE

$29,675
$11,495
$14,475
$17,775
$29,675

DTR> MODIFY ALL USING PRICE = PRICE + 700
DTR> PRINT ALL PRICE

 PRICE

$30,375
$12,195
$15,175
$18,475
$30,375

DTR>
```

2. Form a collection of yachts. Modify all the elementary fields within the group field SPECIFICATIONS for the first record.

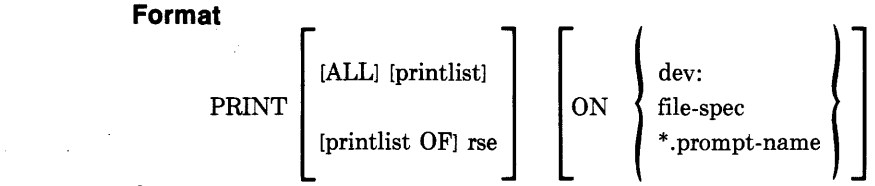
```
DTR> FIND YACHTS
[113 records found]
DTR> SELECT 1
DTR> MODIFY SPECS
Enter RIG:
```

5.25 PRINT Statement

Function

Provides output of one or more field values in a record or a group of records.

Format



Arguments

ALL
Specifies that the records in the current collection are to be used in the output.

print-list
Is a list of value expressions and formatting information. Print-list elements are described in Table 5–5. Use commas to separate print-list elements. If you specify a print-list and a record selection expression, you must separate them with the keyword OF.

rse
Is a record selection expression that specifies the records to be used in the output.

dev:
file-spec
*.prompt-name
Is a prompting value expression or the name of the file-spec file or device to which DATATRIEVE sends the output. By specifying a device name in the PRINT command, you can tell DATATRIEVE to send the output to any system device to which you have access, such as a line printer, a tape drive, or your terminal. On RSTS/E systems, if you specify the line printer for output and it is busy, nothing will be printed. To send your output to a tape drive, you must mount your tape and assign the tape drive to your process before specifying the tape drive in the PRINT command. If you specify a file, the file specification must be in the following format:

```
dev:[UFD]filename.ext;ver
```

You must specify at least a file name. All other fields are optional. If you omit a field in the file specification, DATATRIEVE uses the following defaults.

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	No default. Must be specified.
.ext	.LST
;ver	1 or 1 higher than current version number (non-RSTS/E systems only)

If you specify a prompting value expression (*.prompt-name), DATATRIEVE will prompt for the output file or device when this command is executed. If you omit this argument, the output is printed on your terminal.

Restrictions

The domain must be readied for READ, WRITE, or MODIFY access. You cannot use a PRINT statement if the domain is readied for EXTEND access. (See the READY command, Section 5.26.)

Results

DATATRIEVE writes the specified data on the file or device which you specify. The format and content of the output depends on the optional arguments you choose.

1. If you include a record selection expression, DATATRIEVE prints data from each record in the record stream.
2. If you specify the argument ALL, DATATRIEVE prints data from each record in the current collection.
3. If you nest PRINT within a FOR statement, DATATRIEVE prints data from the context record resulting from the FOR statement.
4. If you do not nest the PRINT statement inside a FOR statement and do not include a record selection expression or the argument ALL, DATATRIEVE prints data from the selected record. If there is no selected record and you do not include any arguments, DATATRIEVE prints the current collection.

5. The print-list argument allows you to specify the following kinds of information:

   - The *data* to be included in the output. Data can be the contents of a field, the value of a variable, or any other value expression.

   - The *format* of the data in the output. You can specify an edit string to override any edit string in the field or variable definition or to format a value expression.

   - The *spacing* (both horizontal and vertical) for the output. You can insert tabs or spaces between columns or skip lines between lines of output.

   - *Column headers* for each column of data in the output. You can also indicate that no header is to be printed above a column.

   If you omit the print-list argument, DATATRIEVE automatically formats the output for you. DATATRIEVE uses the following defaults when you omit the print list.

   - The *data* included in the output is the contents of all fields in the records.

   - The *format* of the field contents is determined by the record definition.

   - The horizontal *spacing* is based on the length of the header and the field contents. The output begins in column 1. The output is single-spaced with a single blank line following the header line.

   - *Column headers* for fields are the query headers, or the field names if there is no query header. If the field name contains a hyphen, DATATRIEVE ignores the hyphen and places each part of the field name on a separate line. The header is centered above the column of data. If the query header contains only a hyphen, DATATRIEVE does not print any header.

Examples

1. Write to a file the builder and length over all of each yacht.

```
DTR> PRINT BUILDER, LOA OF YACHTS ON REPORT.LST
DTR>
```

2. Display the yachts that are 30 or 31 feet long on your terminal.

```
DTR> FIND YACHTS WITH LOA BETWEEN 31 AND 33
[24 records found]
DTR> PRINT ALL BUILDER, RIG VIA RIG-TABLE (-) USING X(30)

MANUFACTURER

 BAYFIELD    ONE MAST
 BOMBAY      ONE MAST
 BUCCANEER   ONE MAST
 CC          ONE MAST
 CAL         ONE MAST
 CHALLENGER  ONE MAST
 COLUMBIA    ONE MAST
 DOUGLAS     ONE MAST
 DOWN EAST   ONE MAST
 ENDEAVOUR   ONE MAST
 FJORD       SOMETHING ELSE
 GRAMPIAN    TWO MASTS, BIG ONE IN FRONT
 MARIEHOLD   ONE MAST
 METALMAST   ONE MAST
 MOODY       ONE MAST
 NICHOLSON   ONE MAST
 O'DAY       ONE MAST
 ONTARIO     ONE MAST
 PEARSON     ONE MAST
 RANGER      ONE MAST
 RHODES      ONE MAST
 RYDER       ONE MAST
 WESTSAIL    ONE MAST
 WRIGHT      ONE MAST

DTR> RELEASE DEPT-TABLE
```

Table 5–5: Print-List Elements

Element	Function
fld-name [modifier]	Specifies the name of the field whose contents are to be output. The optional modifier (see Table 5–6) describes the format of the output and/or column header for the field. If the field name is the name of a group field, DATATRIEVE prints all fields subordinate to the group field. If you omit this element, all fields are output.
val-exp [modifier]	Specifies a value expression to be evaluated and output. The optional modifier (see Table 5–6) describes the format of the output and/or column header for the value expression.
SPACE [n]	Inserts n horizontal spaces before the next entry in the print list. If you omit n, DATATRIEVE inserts one space before the next entry.
TAB [n]	Inserts n tab characters before the next entry in the print list. (Tabs are set at every 8 spaces.) If you omit n, DATATRIEVE inserts one tab before the next entry.
COL n	Advances across the line to column n before outputting the next entry in the print list. If n is less than the current column number, DATATRIEVE skips a line.
SKIP [n]	Skips n blank lines to start a new line before outputting the next entry. If you omit n, DATATRIEVE skips one line. If you omit this element, DATATRIEVE outputs entries on consecutive lines. All output begins at column 1 unless another print-list element modifies the position.
NEW-PAGE	Begins a new print page, without column headers. Printing starts at column 1 unless another print-list element modifies the position.
ALL [print-list OF] rse	Nests a print list within a print list. If you omit the print-list OF argument, all records specified by the RSE are printed.

Table 5-6: Print-List Modifiers

("header-1" [/"header-2"] ...) (-) Indicates the column header to be printed above the immediately preceding field name or value expression. You must place this modifier before the USING edit-string modifier. This modifier must be enclosed in parentheses. If you specify a character string literal ("header"), the character string is printed above the column. If you specify more than one character string literal ("header-1"\"header-2", etc.), the literals are printed on successive lines, centered above the column of data. If you specify a hyphen (-), no header is printed above the column of data. If you specify a hyphen for a group field, no column headers are printed for any fields contained in the group field. If you omit this modifier, DATATRIEVE uses either the query header or the field name as the column header for a field. The column header for value expressions formed with statistical functions (MAX, MIN, AVERAGE, and TOTAL) and a field name (such as MAX PRICE) is the field name (such as PRICE). For all other value expressions, DATATRIEVE does not use a column header.
USING edit-string Imposes the characteristics of the specified edit string on the immediately preceding field or value expression. The edit string must conform to the rules for edit strings in the EDIT-STRING clause of a record definition. (See Section 11.8.) This modifier has no effect if you use it with a group field. If you omit this modifier for a field, DATATRIEVE uses the edit string specified for the field in the record definition or the PICTURE (or PIC) clause if no edit string is given.

5.26 READY Command

Function

Readies a domain for use.

Format

```
                     [ (passwd) ] [ SHARED    ] [ READ   ]
READY domain-name    [          ] [ PROTECTED ] [ MODIFY ]
                     [   (*)    ] [ EXCLUSIVE ] [ WRITE  ]
                                                [ EXTEND ]
```

Arguments

domain-name
Is the name of a domain in the current data dictionary to be readied for use.

(passwd)
(*)
Is the password required to access the domain or an asterisk enclosed in parentheses (*). If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password but does not print it on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have appropriate access privilege.

SHARED
PROTECTED
EXCLUSIVE
Indicates how you allow other users to access this domain. (See Table 5–7 for an explanation of allow types.) If you omit this argument, your access is Protected, and other users can read, but not write or modify, records in the domain.

READ
MODIFY
WRITE
EXTEND
Is the access mode, indicating the type of access you require. (See Table 5–8 for access modes and the access privilege required for each mode.) If you omit this argument, your access mode is READ and you can retrieve records, but you cannot modify or store them.

Restrictions

1. You must have E (execute) privilege to the associated record definition.
2. You must have an access privilege to the domain that permits the access mode you specify. See Table 5–8 for a list of privileges you must have for each access mode.
3. You can ready only one domain each time you issue the READY command. However, you can ready more than one domain in a session by issuing multiple READY commands.

4. You must have operating-system-level access to the data file.
5. If another user has readied the domain for exclusive use, you cannot ready the domain. If another user has readied the domain for protected use then you can ready the domain for READ access only.
6. The file specified in the domain definition must exist.

Results

1. DATATRIEVE readies the specified domain. DATATRIEVE uses the file specification in the domain definition. If the domain definition omits a field in the file specification, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	No default. Must be specified.
.ext	.DAT
;ver	Highest version number (non-RSTS/E systems only)

2. If you issue more than one READY command for a domain without an intervening FINISH command, DATATRIEVE preserves your collections.

Table 5-7: Domain Allow Types

Allow Type	Meaning
SHARED	Any other user can have concurrent access to the records in this domain, for any purpose.
PROTECTED	Any other user can read, but not write or modify, records in the domain. (Default)
EXCLUSIVE	No other user can access this domain at the same time, for any purpose.

Table 5-8: Domain Access Modes

Access Mode	Meaning	Privilege Required
READ	You can retrieve records only. (Default)	Read, write, or modify
MODIFY	You can retrieve and modify records.	Modify or write
WRITE	You can retrieve, modify, store, and erase records.	Write
EXTEND	You can store records only.	Extend, write, or modify

Usage Notes

1. To verify that a domain is ready, use the following format of the SHOW command:

```
SHOW READY
```

   SHOW READY prints the name, file type, access type, and access mode of all readied domains. (See Section 5.32 for more information on SHOW READY.)

2. The access mode you specify determines the commands and statements you can use on this domain. If you will be issuing any command or statement listed in Table 5–9, be sure to request the appropriate access mode.

Table 5–9: Access Mode Required by Commands/Statements

Command/Statement	Access Mode Required
ERASE	WRITE
FIND	MODIFY, READ, or WRITE
MODIFY	MODIFY or WRITE
PRINT	MODIFY, READ, or WRITE
STORE	WRITE or EXTEND

3. When specifying the access allowed to other users, be as nonrestrictive as possible. For example, if you specify EXCLUSIVE access, no other user can access the domain.

4. When specifying the type of access you require, be as restrictive as possible. For example, if you will only be retrieving records, specify READ access, which is more efficient than MODIFY or WRITE access.

5. You can issue more than one READY command for the same domain in a single session. This facility allows you to change the allow type or access mode you require. However, when you re-ready a domain without finishing it, DATATRIEVE ignores the optional password argument and uses the original access information.

   For example, during a session you might modify records, then output records with the PRINT command. In this case, you can ready the domain

for WRITE access, modify the records with the MODIFY statement, and then ready the domain for READ access before issuing the PRINT statement.

```
DTR>  READY YACHTS (PASSWD) EXCLUSIVE WRITE
DTR>  MODIFY ...
        .
        .
        .
DTR>  READY YACHTS PROTECTED READ
DTR>  PRINT ...
```

6. A domain remains ready until you release it with the FINISH command or terminate the session with EXIT or CTRL/Z. If you have no further use for a domain, but do not want to terminate a session, issue the following FINISH command for the domain:

```
FINISH domain-name
```

FINISH releases your control of a domain and its collections, allowing other users access to the domain and freeing up the resources utilized by the domain. (See Section 5.20 for more information on FINISH.)

Examples

1. Ready the domain YACHTS for WRITE access.

```
DTR> READY YACHTS WRITE
```

2. Ready the domain PHONES for EXTEND access.

```
DTR> READY PHONES (*) EXTEND
Enter password for PHONES:
DTR>
```

5.27 RELEASE Command

Function

Releases a collection, a description table, or a global variable, freeing its workspace for other use.

Format

```
RELEASE { collectn-name  }
        { table-name     } ...
        { variable-name  }
```

Arguments

collectn-name
table-name
variable-name

Is the name of a collection, a description table, or a global variable to be released. To release variable-name more than one item, separate them with commas.

Restrictions

None.

Results

1. DATATRIEVE releases the collection, description table, or global variable and frees its workspace. The records and domain from which a collection originated are unaffected by this command.
2. If you specify more than one item to be released, DATATRIEVE releases the workspace of each item in the order specified in the command. If the command fails before the last item is processed, DATATRIEVE prints an error message indicating which item failed to be released. All items before that item in the command were released.

Usage Notes

1. You cannot reference a global variable once it is released. If necessary, you can redefine the variable with the DECLARE statement. (See Section 5.5.)
2. Use of the RELEASE command can be redundant, as in the following instances:
   - A FIND statement that successfully establishes a collection releases an existing collection of the same name.
   - Ending a session (with EXIT or CTRL/Z) releases all global variables and all collections established during a session.
   - A FINISH command releases all collections associated with the domain(s) specified in the command.

Examples

1. Release the collection BIG-ONES.

```
DTR> SHOW COLLECTIONS
Collections:
        BIG-ONES (also CURRENT)
DTR> RELEASE BIG-ONES
DTR> SHOW COLLECTIONS
No established collections
DTR>
```

2. Release the table DEPT-TABLE.

```
DTR> RELEASE DEPT-TABLE
```

5.28 REPEAT Statement

Function

Causes DATATRIEVE to execute a statement a specified number of times.

Format

REPEAT val-exp statement

Arguments

val-exp
: Is a value expression indicating the number of times to execute the statement. This argument must evaluate to a positive, nonzero integer.

statement
: Is any simple or compound statement (except a FIND or SELECT statement).

Restrictions

1. Do not include a FIND or SELECT statement in a REPEAT statement.
2. You must observe any restriction on the statement included in a REPEAT statement that is listed in its description.

Results

DATATRIEVE executes the specified statement the indicated number of times. DATATRIEVE then executes the command or statement following the REPEAT statement.

Examples

1. Print "TEST REPEAT" three times.

```
DTR> REPEAT 3 PRINT "TEST REPEAT"
TEST REPEAT
TEST REPEAT
TEST REPEAT

DTR>
```

2. Store five new records in the YACHTS domain.

```
DTR> REPEAT 5 STORE YACHTS
```

5.29 REPORT Command

Function

Invokes the Report Writer.

Format

```
REPORT [rse]  ON { dev:
                 { file-spec
                 { *.prompt-name }
```

Arguments

rse

Is a record selection expression specifying the data for your report. You can make reports on the following kinds of data:

- Readied domains
- Collections
- Sublists

See Chapter 6 for detailed information on record selection expressions. See Chapter 15 for information on sublists. If you omit this argument, the Report Writer uses all records in the current collection for your report.

dev:
file-spec
*.prompt-name

Is the name of the file or device to contain the report or a prompting value expression. By specifying a device name in the REPORT command, you can tell the Report Writer to send the report to any system device to which you have access, such as a line printer, a tape drive, or your terminal. On RSTS/E systems, if you specify the line printer for output and it is busy, the Report Writer may not produce the report. If no report is produced for this reason, you may have to consult with your system manager to work around the problem. To send your report to a tape drive, you must mount your tape and assign the tape drive to your process before specifying the tape drive in the REPORT command. If you specify a file, the file specification must be in the following format:

```
dev:[UFD]filename.ext;ver
```

You must specify at least a file name. All other fields are optional. If you omit a field, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	No default. Must be specified.
.ext	.LST
;ver	1 or 1 higher than current version number (non-RSTS/E systems only)

If you specify a prompting value expression (*.prompt-name), DATATRIEVE will prompt for the output file or device when this command is executed.

If you omit this argument, the report is printed on your input terminal.

Restrictions

None.

Results

When this command is executed, DATATRIEVE prompts with RW>, indicating that the Report Writer is executing and is expecting a report definition. The Report Writer continues to prompt with RW> until you either enter a line consisting of END-REPORT or enter a syntax error. After you enter END-REPORT, DATATRIEVE writes your report to the device or file you specify. If you make a syntax error, DATATRIEVE returns to command level without writing any report.

Usage Notes

Refer to Chapter 7 for a full description of the Report Writer.

Example

Use the Report Writer to display data from the domain FAMILIES.

```
DTR> REPORT FAMILIES
RW>  SET REPORT-NAME = "REPORT ON FAMILIES"
RW>  PRINT FATHER, MOTHER, KIDS, SKIP
RW>  AT BOTTOM OF REPORT PRINT TOTAL NUMBER-KIDS (-) , COUNT (-)
RW>  END-REPORT

                    REPORT ON FAMILIES
                                                        26-May-80
                                                           Page 1


                                 KID
  FATHER          MOTHER         NAME      AGE

JIM               ANN            URSULA    8
                                 RALPH     4

JIM               LOUISE         ANNE      32
                                 JIM       30
                                 ELLEN     27
                                 DAVID     25
                                 ROBERT    17

JOHN              JULIE          ANN       30
                                 JEAN      27

                    :
                    :

                                                35        14
```

5.30 SELECT Statement

Function

Selects a record from the CURRENT collection or any named collection.

Format

```
SELECT { FIRST | NEXT | LAST | val-exp } [collectn-name]
```

Arguments

FIRST
LAST
NEXT
val-exp
: Indicates which record in the collection is to be selected. If you specify a value expression, the expression must evaluate to a positive integer that does not exceed the number of records in the collection. If this argument is omitted, DATATRIEVE selects either the first record (if this is the first time SELECT is issued for the collection) or the next record in the collection.

collectn-name
: Is the name of the collection containing the records to be selected. If omitted, the record is selected from the current collection.

Restrictions

1. You must establish the named collection or a current collection before issuing this statement.
2. The collection cannot be empty.
3. You cannot use a SELECT and FIND statement in the same compound statement.
4. You cannot use a SELECT statement in a BEGIN–END block or in a FOR or REPEAT statement.

Results

1. SELECT establishes the current record. The actual record selected is dependent on the arguments you supply, the state of the collection, and the position of the collection cursor when this command is issued.
2. If you specify SELECT NEXT and the collection cursor points to the last record, DATATRIEVE prints an error message. The collection cursor is unchanged.
3. Selecting a record does not make the collection current if it was not already current.

Usage Notes

1. If you want to know the record number of the record selected, use the SHOW command (Section 5.32) in the following format:

```
SHOW collectn-name
```

   This command prints the domain name of the collection, the number of records in the collection, and the record number of the selected record.

2. To ascertain the name and attributes of the current collection, you can use the SHOW CURRENT command (Section 5.32).

3. Fields of the selected record can be referenced like global variables.

4. If you want to perform the same set of statements on each record in the current collection you *cannot* do the following:

```
REPEAT 50
  BEGIN
    SELECT NEXT
       :
       :
  END
```

   The proper way to do this is:

```
FOR CURRENT
  BEGIN
     :
     :
  END
```

Examples

1. Select the last record in the current collection.

```
DTR> SELECT LAST
DTR>
```

2. Select the fifth record in the collection BIG-ONES.

```
DTR> SELECT 5 BIG-ONES
DTR>
```

5.31 SET Command

Function

Establishes the current data dictionary, sets the maximum number of columns per page, inhibits or permits aborts, starts Guide Mode, and inhibits or permits statement prompting.

Format

```
SET { DICTIONARY [file-spec]  }
    { COLUMNS-PAGE=n          }
    { ABORT                   }
    { NO ABORT                } [, ...]
    { GUIDE                   }
    { PROMPT                  }
    { NO PROMPT               }
```

Arguments

DICTIONARY [file-spec]

Causes DATATRIEVE to connect you to a specific data dictionary. If you do not include a file-spec, DATATRIEVE connects you to the data dictionary that was your current data dictionary when you started the session. If you include a file-spec, DATATRIEVE connects you to the data dictionary with that file specification. Use the following format for a file-spec:

dev:[UFD]filename.ext;ver

If you omit a field in the file specification, DATATRIEVE uses the following defaults.

Field	Default
dev:	SY: (the system device)
[UFD]	Your default UFD
filename	QUERY
.ext	.DIC
;ver	Highest version number (non-RSTS/E systems only)

COLUMNS-PAGE=n

Establishes the number of columns per page for DATATRIEVE output and the default page width for the Report Writer. The argument n must be an unsigned integer that is less than 256. The default number of columns per page is established at installation time.

ABORT
NO ABORT

Determines the effect of an ABORT statement or CTRL/Z inside a procedure or indirect command file. If SET ABORT is in effect and DATATRIEVE executes an ABORT statement or you enter a CTRL/Z in response to a prompt, DATATRIEVE aborts the entire procedure or command file. If SET NO ABORT is in effect, DATATRIEVE continues with the next statement of the procedure or command file. At the start of a DATATRIEVE session, SET NO ABORT is in effect.

GUIDE

Starts GUIDE Mode. To terminate GUIDE Mode, type L. (Refer to the *DATATRIEVE Primer* for more information.)

PROMPT
NO PROMPT

Determines whether DATATRIEVE prompts for missing statement elements with the following message:

```
[Looking for element]
```

At the start of a DATATRIEVE session, SET PROMPT is in effect.

Restrictions

None.

Results

If you specify SET DICTIONARY file-spec and the command fails because the specified file does not exist, cannot be accessed, or is not a valid dictionary, the current data dictionary is unchanged. That is, the dictionary in effect before the SET command was issued is still in effect.

Examples

1. Set the number of columns per page to 80.

```
DTR> SET COLUMNS_PAGE =80
DTR>
```

2. Connect to your private data dictionary.

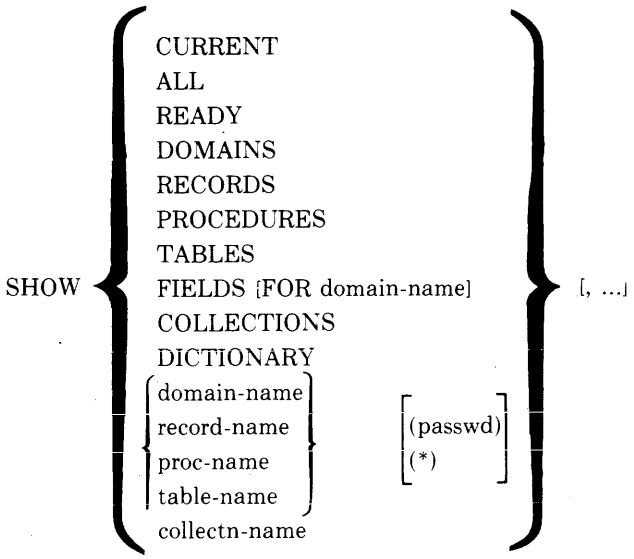
```
DTR> SET DICTIONARY [202,202]MYOWN.DIC
```

5.32 SHOW Command

Function

Prints information about the data dictionary and its contents.

Format



Arguments

CURRENT
Prints the domain name, the number of records in, and the number of the selected records in the current collection.

ALL
Prints the names of all domains, records, description tables, collections, procedures, views, and readied domains in the current data dictionary.

READY
Prints the names, file types, access types, and access modes of all readied domains in the current data dictionary.

DOMAINS
Prints the names of all domains defined in the current data dictionary.

RECORDS
Prints the names of all records defined in the current data dictionary.

PROCEDURES
Prints the names of all procedures defined in the current data dictionary.

TABLES
Prints the names of all description tables defined in the current data dictionary and all description tables in the DATATRIEVE workspace.

FIELDS [FOR domain-name]

Prints the names of all fields in the specified domain or in all readied domains. This also prints the names of all global variables. (If a domain name is specified, it must be the name of a readied domain.)

COLLECTIONS

Prints the names of all collections.

DICTIONARY

Prints the complete file specification of the current data dictionary.

domain-name
record-name
proc-name
table-name

Prints the dictionary definition of the named domain, record, procedure, or description table.

(passwd)
(*)

Is an asterisk enclosed in parentheses (*) or the password necessary to gain R (read) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have R (read) access privilege.

collectn-name

Prints the domain name, the number of records in, and the number of the selected record in the specified collection.

Restrictions

You must have read access to a domain, record, procedure, or description table to print its definition.

Results

DATATRIEVE prints the requested information on your terminal, in the order requested.

Examples

1. Display the names of all the entries in the current data dictionary.

```
DTR> SHOW ALL
```

2. Display the definition of the record YACHT.

```
DTR> SHOW YACHT
```

3. Display the names of all global variables, and the field names of all readied domains.

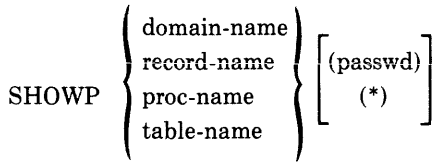
```
DTR> SHOW FIELDS
```

5.33 SHOWP Command

Function

Prints a password table on your terminal.

Format



Arguments

domain-name
record-name
proc-name
table-name

Is the name of the domain, record, procedure, or description table whose password table is to be printed.

(passwd)
(*)

Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts for the password, but does not print your response on the terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege.

Restrictions

You must have C (control) access privilege to the domain, record, procedure, or description table.

Results

DATATRIEVE prints the entire password table for the specified domain, record, procedure, or description table.

Examples

1. Display the password table for the YACHTS domain.

```
DTR> SHOWP YACHTS
```

2. Use a password to gain C (control) access to the record YACHT and display its password table.

```
DTR> SHOWP YACHT (*)
Enter password for YACHT:
```

5.34 SORT Statement

Function

Sorts records in a collection according to the contents of one or more fields.

Format

SORT [collectn-name][BY] sort-key-1 [,sort-key-2]...

Arguments

collectn-name
Is the name of the collection to be sorted. If you omit the collection name, records in the current collection are sorted.

BY
Is included with the command to preserve English-language-like syntax. It serves no other purpose.

sort-key
Is a sort key indicating both the order of the sort and the field to be used for the sort. Each sort key must be in the following format:

```
ASC[ENDING]
DESC[ENDING]   fld-name
INCREASING
DECREASING
```

If more than one sort key is specified, they must be separated by a comma (,).

Restrictions

1. You must have established the collection to be sorted before issuing this statement.
2. You cannot sort COMPUTED BY fields.

Results

1. DATATRIEVE sorts the collection based on the contents of the field(s) you specify. Appendix F describes the order by which alphanumeric fields are sorted.
2. If you specify ASC[ENDING] (or INCREASING), DATATRIEVE places the lowest value first and the highest value last in the collection.

   If you specify DESC[ENDING] (or DECREASING), DATATRIEVE places the highest value first and the lowest value last in the collection.
3. You can specify any number of sort keys. If you specify more than one, DATATRIEVE treats the first field name as the major sort key and each successive field name as increasingly minor keys.

4. If you omit a sorting direction (ASCENDING, INCREASING, etc.) for the first sort key, DATATRIEVE sorts the field in ascending order.

   If you omit a sorting direction on the second or subsequent sort keys, DATATRIEVE uses the preceding sort direction (implied or specified) for the field.

5. The collection cursor is null at the completion of this command.

Usage Notes

If you will be sorting large amounts of data or will be specifying several sort keys, it is a good practice to free as much workspace as possible before issuing the SORT statement. You can free workspace in the following ways.

- Relinquish any domains you can with the FINISH command (Section 5.20).
- Relinquish any collections, description tables, or global variables you can with the RELEASE command (Section 5.27).
- Change the access mode of all readied domains from WRITE or MODIFY to READ, which uses less buffer space. (See the READY command, Section 5.26.)

Examples

1. Form a collection of families. Sort the collection by decreasing number of children.

```
DTR> READY FAMILIES
DTR> FIND FAMILIES
[14 records found]
DTR> SORT BY DECREASING NUMBER-KIDS
DTR> PRINT ALL

                          NUMBER    KID
  FATHER      MOTHER       KIDS     NAME     AGE

BASIL         MERIDETH       6      BEAU     29
                                    BROOKS   27
                                    ROBIN    25
                                    JAY      23
                                    WREN     18

                                    JILL     21
JIM           LOUISE         5      ANNE     32
                                    JIM      30

                                 :
                                 :
ROB           DIDI           0
```

2. Form a collection of yachts. Sort the collection using the builder as the primary key and the length over all as the second key.

```
DTR> FIND YACHTS
[113 records found]
DTR>SORT BY BUILDER, LOA
```

5.35 STORE Statement

Function

Creates a record in a domain and stores values in all fields or selected fields in the record.

Format

STORE domain-name [[USING] statement-1] [VERIFY [USING] statement-2]

Arguments

domain-name
Is the name of the domain to contain the new record. The domain must be readied for WRITE or EXTEND access.

USING
Is included in the command to preserve English-language-like syntax. It serves no other purpose. If specified, the keyword USING must be on the same input line as the keyword STORE.

statement-1
Is a statement to be used to store field values in the record. Typically, statement-1 is an assignment statement or a BEGIN-END block containing a series of assignment statements. If omitted, DATATRIEVE prompts for all elementary field values in the record.

VERIFY [USING]
statement-2
Indicates that no value is to be stored in a field unless the condition specified by statement-2 is met. Typically, statement-2 is an IF-THEN-ELSE statement or a BEGIN-END block containing a series of IF-THEN-ELSE statements.

Restrictions

1. The domain to contain the new record must be readied for WRITE or EXTEND access. (See the READY command, Section 5.26.)
2. You cannot store a new record in an RMS relative file or a view.

Results

1. If you omit the USING statement-1 clause, DATATRIEVE prompts for a value for the first elementary field in the record with the following message.

```
ENTER fld-name:
```

After you enter a value, DATATRIEVE prompts for the next field value in the record until you have entered a value for every field in the record.

To leave a field empty, enter a single space followed by a carriage return. If you enter only carriage return in response to a prompt, DATATRIEVE repeats the prompt.

If you enter data that is inappropriate for a field, DATATRIEVE reprompts for valid data.

2. If you include a USING clause, DATATRIEVE does not reprompt for data.

   If you enter too many digits for a numeric field, DATATRIEVE truncates the high-order digits, stores the remaining digits in the field, and issues an error message.

   If you enter too many characters for an alphanumeric field, DATATRIEVE truncates the rightmost characters, stores the remaining characters in the field, and issues an error message.

3. If you specify a VERIFY USING clause, no data is stored until statement-2 is successfully executed.

Usage Notes

1. If you include the USING statement-1 clause, statement-1 will generally be an assignment statement or a BEGIN-END block containing a series of assignment statements. The formats of the assignment statement are:

   fld-name = val-exp

   group-fld-name-1 = group-fld-name-2

   You should review the description of the assignment statement for elementary fields (Section 5.3.1) or group fields (Section 5.3.2) before using this format of the STORE statement.

   If you put a BEGIN-END block in the USING clause, you can include statements other than assignment statements. To see what you are storing, include a PRINT statement in the BEGIN-END block. Section 5.4 describes the BEGIN-END block.

2. To store more than one record in a domain, you can use the following format of the REPEAT statement:

   REPEAT n STORE domain-name

   The argument n represents the number of records to be stored in the domain.

Examples

1. Store a record in the FAMILIES domain.

```
DTR> READY FAMILIES WRITE
DTR> STORE FAMILIES
Enter FATHER:
```

2. Store a record in the YACHTS domain.

```
DTR> READY YACHTS WRITE
DTR> REPEAT 3 STORE YACHTS USING
[Looking for assignment statement(s)]
DTR> BEGIN
[Looking for statement]
DTR> BUILDER="AMERICAN"
[Looking for statement or "end"]
DTR> MODEL = *.MODEL
[Looking for statement or "end"]
DTR> END
Enter MODEL: 1980
Enter MODEL: 1980-A
Enter MODEL: 1980-B
DTR> PRINT YACHTS WITH BUILDER EQ "AMERICAN"

                              LENGTH
                               OVER
MANUFACTURER  MODEL    RIG      ALL  WEIGHT  BEAM  PRICE
 AMERICAN     1980                       0    00
 AMERICAN     1980-A                     0    00
 AMERICAN     1980-B                     0    00
 AMERICAN     26       SLOOP   26    4,000    08    $9,895
 AMERICAN     26-MS    MS      26    5,500    08   $18,895

DTR>
```

5.36 SUM Statement

Function

Provides a summary report of the total of the contents of any number of numeric fields in the current collection. The output can be sorted by the contents of one or more fields and can be directed to any device or file.

Format

SUM print-list BY sort-key ON { dev: / file-spec / *.prompt-name }

Arguments

print-list
Is a list of numeric fields or other value expressions for DATATRIEVE to use. The format of the print-list is:

val-exp-1 [modifier-1] ,val-exp-2 [modifier-2] . . .

Modifiers are the same as the print-list modifiers described in Section 5.25 and summarized in Table 5-6.

sort-key
Determines the order of the sort and the fields for DATATRIEVE to sort by. See Section 5.34 for the format and effect of sort keys.

dev:
file-spec
*.prompt-name
Is the name of the file or device to contain the summary report or a prompting value expression. If you specify a file, the file specification must be in the following format:

dev:[UFD]filename.ext;ver

You must specify at least a file name. All other fields are optional. If you omit a field, DATATRIEVE uses the following defaults:

Field	Default
dev:	SY: (the system device)
UFD	Your default UFD
filename	No default. Must be specified
.ext	.LST
;ver	1 or 1 higher than current version number (non-RSTS/E systems only)

If you specify a prompting value expression (*.prompt-name), DATATRIEVE prompts for the output file or device when this command is executed. If you omit this argument, the output is printed on your terminal.

Results

1. DATATRIEVE sorts the CURRENT collection according to the fields you specify and outputs a summary report on the specified (or default) device. The summary report includes the total of the contents of the fields specified by the print list.

2. The order and format of the data in the report depends on the arguments you select. For more information on the sort order, refer to the description of the SORT command, Section 5.34. For more information on the format of the data, refer to the description of print-list modifiers in Section 5.25 and Table 5–6.

Usage Notes

You can use the SUM statement to get a count of the number of records in the collection by field name with the following format:

SUM 1 BY sort-list

Examples

1. Form a collection of yachts. Use the SUM statement to summarize the prices of yachts in the collection.

```
DTR> FIND YACHTS
[113 records found]
DTR> SELECT 1
DTR> MODIFY SPECS
Enter RIG:

DTR>

DTR>

DTR> FIND YACHTS
[113 records found]
DTR> SUM PRICE BY BUILDER

MANUFACTURER    PRICE

 ALBERG           36951
 ALBIN            64000
 AMERICAN         28790
 BAYFIELD         32875
 BLOCK I.             0
 BOMBAY           23950
 BUCCANEER            0
 CC                   0
 CABOT                0
 CAL                  0
                  :
                  :
                  :

                 1272933
```

2. Use the SUM statement to display the number of yachts built by each builder.

```
DTR> SUM 1 BY BUILDER

MANUFACTURER

 ALBERG            1
 ALBIN             3
 AMERICAN          2
 BAYFIELD          1
 BLOCK I.          1
 BOMBAY            1
 BUCCANEER         2
 CC                1
 CABOT             1
 CAL               5
 CAPE DORY         3
 CAPITAL           1
          :
          :
          :
          113
```

5.37 THEN Statement

Function

Joins two statements into a compound statement.

Format

statement–1 THEN statement–2

Arguments

statement–1
Is any DATATRIEVE statement.

statement–2
Is any DATATRIEVE statement.

Restrictions

You cannot include FIND and SELECT in the same compound statement.

Results

DATATRIEVE executes statement–1 and statement–2. You can use a compound statement anywhere you can use a single statement.

Example

Make a compound statement by using THEN to join a PRINT statement and a MODIFY statement. Nest the compound statement within a FOR statement.

```
DTR> FOR YACHTS WITH BUILDER EQ "ALBIN" PRINT THEN MODIFY
```

Chapter 6
Expressions

An expression is a statement element that represents a value or that specifies the record (or records) that statements are to operate on. DATATRIEVE provides three types of expressions:

- Value expressions, which represent values
- Boolean expressions, which are either "True" or "False"
- Record selection expressions, which specify a record stream

6.1 Value Expressions

A value expression is a means of specifying a value, which DATATRIEVE computes. DATATRIEVE provides several methods for specifying value expressions. The methods for specifying value expressions are explained in Sections 6.1.1 through 6.1.8.

6.1.1 Literals

The simplest way to specify a value is with a literal. A literal is a character string enclosed in quotation marks or a number. A character string literal can contain up to 132 characters. You can continue a character string literal on more than one line by using the continuation character hyphen. To specify character strings longer than 132 characters use concatenated expressions (Section 6.1.8). You can include a quotation mark in a character string literal by directly preceding it with a quotation mark. Examples of character string literals are:

"Invalid value for this field"

"MAXIMUM PRICE IS $1400. PLEASE REENTER PRICE."

"Capt. Jack says, " "Time to reorder." " "

A numeric literal is a string of digits that DATATRIEVE interprets as a decimal number. A numeric literal can contain up to 18 digits and may contain a decimal point. A numeric literal cannot begin with a decimal point. Thus, for example, 0.5 is a valid numeric literal while .5 is not.

6.1.2 Field Names

You can use a field name as a value expression. The value that you specify is the value stored in a field of a record. If the record definition contains a query name for a field, the query name is used just like a field name.

Use the following format to qualify field names.

[context-variable.] [record-name.] [group-name.]... field-name

The context-variable qualifier allows you to distinguish between fields from different record streams. The record-name qualifier allows you to distinguish between fields in different domains which have the same field name. The group-name qualifier allows you to distinguish between fields that are contained in different group fields and have the same field name.

To refer to the fields within a record, you must create a context. Table 6–1 lists the different ways to create contexts, and how long each context lasts.

Table 6–1: Context Types

Statement or Clause	Context	Valid for
SELECT	Selected record	Any statement
FOR	Record in record stream	Statements nested within FOR statement
VERIFY	Record being stored or modified	Statements within VERIFY clause
OF	Record stream	Statement containing the OF clause
WITH	Domain, collection, or list that the rse refers to	Boolean expression in the WITH clause of an rse

If there is more than one valid context for a field name, DATATRIEVE uses the most recent valid context. If there is no valid context for a field name, DATATRIEVE issues an error message. The following examples show how field names are used as value expressions:

Use a selected record to show how you include qualifiers in field names. Use a query name to print the length.

```
DTR> FIND YACHTS
[114 records found]
DTR> SELECT 9
DTR> PRINT MODEL

MODEL

CLIPPER

DTR> PRINT TYPE.MODEL
```

(continued on next page)

```
MODEL

CLIPPER

DTR> PRINT BOAT,TYPE,MODEL

MODEL

CLIPPER

DTR> PRINT LOA

LENGTH
 OVER
 ALL

 31
```

Increase the price of each yacht by $500.

```
DTR> FOR YACHTS MODIFY USING PRICE = PRICE + 500
DTR>
```

In the previous example the field name PRICE is used twice. On the left side of the equal sign it specifies the field being modified, on the right side of the equal sign it is a value expression, representing the value originally stored in the field.

6.1.3 Variables

DATATRIEVE allows you to declare both global and local variables. Global variables are all variables that are not declared within BEGIN–END blocks. You can use a global variable as a value expression in any statement.

Local variables are all variables declared within BEGIN–END blocks. You can only use a local variable within the BEGIN–END block where you declare it. You can give a local variable the same name as a variable declared outside of that BEGIN–END block. DATATRIEVE resolves any ambiguous variable name by using the most recently declared variable with that name.

If a variable name is the same as a field name, DATATRIEVE uses the most recent valid context for a name. The context of a local variable is more recent than any context created outside the BEGIN–END block. The context of a global variable is less recent than all other possible contexts. To avoid trouble, do not use variable names that duplicate field names.

The following example shows how to use a variable as a value expression.

```
DTR> DECLARE PRICE-PER-POUND COMPUTED BY PRICE / DISP
DTR> EDIT-STRING IS 9.99,
DTR> FOR FIRST5 YACHTS PRINT BOAT,PRICE-PER-POUND
```

(continued on next page)

```
                                    LENGTH                                  PRICE
                                     OVER                                    PER
MANUFACTURER   MODEL      RIG       ALL   WEIGHT   BEAM     PRICE          POUND

 ALBERG        37 MK II   KETCH     37    20,000    12      $36,951         1.84
 ALBIN         79         SLOOP     26     4,200    10      $17,900         4.26
 ALBIN         BALLAD     SLOOP     30     7,276    10      $27,500         3.77
 ALBIN         VEGA       SLOOP     27     5,070    08      $18,600         3.66
 AMERICAN      26         SLOOP     26     4,000    08       $9,895         2.47

DTR> PRINT MAX PRICE-PER-POUND OF YACHTS

PRICE
 PER
POUND

5.00

DTR>
```

6.1.4 Prompting Value Expressions

If you want DATATRIEVE to prompt you for a value, use a prompting value expression. This feature is especially useful in procedures, as it allows you to use different values each time you invoke your procedures. The basic format of a prompting value expression is:

*."prompt-name"

You specify a prompt name, preceded by an asterisk and period. If the prompt name you select conforms to the rules for names given in Section 4.4, you do not have to enclose it in quotation marks. If the prompt name does not conform to those rules, you must enclose it in quotation marks. If you place a prompting value expression in a REPEAT or FOR statement, you can have DATATRIEVE prompt you only once by using two asterisks. If you include only a single asterisk, DATATRIEVE prompts you for a value each time the statement is executed.

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

```
DTR> READY PHONES WRITE
DTR> REPEAT 5 STORE PHONES USING BEGIN
DTR> DEPARTMENT = **.DEPARTMENT
DTR> LOCATION = **.LOCATION
DTR> NAME = *.NAME
DTR> NUMBER = *.NUMBER
Enter DEPARTMENT: SD
Enter LOCATION: PK5
Enter NAME: Sales, John
Enter NUMBER: 8543477
Enter NAME: Hammond, Ken
Enter NUMBER: 8541010
Enter NAME:
```

You can use a prompting value expression anywhere that a value expression is required. You can also use it in place of a device name or file name in the REPORT command and the PRINT statement.

The following example shows how to use *.prompt in a REPORT command:

```
DTR> DEFINE PROCEDURE REPORT1
DFN> REPORT ON *,"Device or File name"
DFN>       :
DFN>       :
DFN> END-REPORT
DFN> END-PROCEDURE
DTR>
DTR> :REPORT1
Enter Device or File name: REPORT.DAT
DTR>
```

6.1.5 Values from a Table

You can use a value stored in a description table anywhere a value expression is required. The format for using a value from a description table is:

value-expression VIA table-name

If the value expression you specify is stored as a code in the table you specify, then the value of the entire expression is the corresponding description in the table. If the value expression you specify does not match any code in the table, then the value of the entire expression is the description stored in the ELSE clause. If the value expression is not found and there is no ELSE clause, DATATRIEVE issues an error message. See Chapter 13 for more information on using tables.

6.1.6 Statistical Functions

DATATRIEVE provides several statistical functions which can be used anywhere a value expression is required. The statistical functions are MAX, MIN, AVERAGE, TOTAL and COUNT. The format for using statistical functions is:

function val-exp [OF rse]

You must include a value expression for all statistical functions except COUNT. Do not include a value expression with COUNT. The value expression is usually a field name. Table 6–2 shows the results of each statistical function.

Table 6–2: Statistical Functions

Function	Value of Function
MAX	The largest value of the value expression
MIN	The smallest value of the value expression
AVERAGE	The average value of the value expression
TOTAL	The total value of the value expression
COUNT	The number of records in the record stream

If you include a record selection expression, DATATRIEVE uses all records in the resulting record stream to compute the value of the function. If you do not include a record selection expression, DATATRIEVE uses all records in the current collection to compute the value of the function.

The following examples show how to use the statistical functions:

```
DTR> FIND YACHTS WITH PRICE GT 0
[50 records found]
DTR> PRINT AVERAGE PRICE

 PRICE

$25,388

DTR> PRINT AVERAGE PRICE OF YACHTS

 PRICE

$11,233

DTR> PRINT YACHTS WITH PRICE EQ MIN PRICE

                                LENGTH
                                 OVER
MANUFACTURER     MODEL   RIG     ALL  WEIGHT  BEAM    PRICE

 VENTURE         21      SLOOP   21    1,500   07      $2,823

DTR> READY FAMILIES
DTR> PRINT TOTAL NUMBER-KIDS OF FAMILIES

NUMBER
 KIDS

  35

DTR> PRINT COUNT OF FAMILIES WITH NUMBER-KIDS EQ 2

 6

DTR> PRINT FAMILIES WITH
[Looking for Boolean expression]
DTR> NUMBER-KIDS EQ MAX NUMBER-KIDS OF FAMILIES

                          NUMBER      KID
  FATHER          MOTHER   KIDS       NAME   AGE

BASIL             MERIDETH    6       BEAU    28
                              BROOKS  26
                              ROBIN   24
                              JAY     22
                              WREN    17
                              JILL    20
```

6.1.7 Arithmetic Expressions

An arithmetic expression consists of value expressions and arithmetic operators. The value expressions must all represent numbers. You can use an

arithmetic expression anywhere a value expression is allowed. DATATRIEVE provides four arithmetic operators. Table 6–3 shows the arithmetic operators and the operation performed.

Table 6–3: Arithmetic Operators

Operator	Operation
+	addition
-	subtraction or negation
*	multiplication
/	division

Use spaces to separate arithmetic operators from value expressions. You can use parentheses to make DATATRIEVE perform some operations first. DATATRIEVE uses the normal rules of precedence in evaluating arithmetic expressions. That is, first DATATRIEVE evaluates any value expressions that are in parentheses, then DATATRIEVE performs multiplications and divisions from left to right, and finally DATATRIEVE performs additions and subtractions from left to right. The following examples show how DATATRIEVE evaluates arithmetic expressions:

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

DTR> PRINT 6 * (5 + 9)
  84

DTR> PRINT 5 + 4 * 3
  17

DTR> PRINT 12 - 6 * 2
  0

DTR> PRINT 5 + 10 / 2
  10
```

6.1.8 Concatenated Expressions

DATATRIEVE allows you to join value expressions to form concatenated expressions. Concatenated expressions can be used anywhere a character string literal is allowed. The two formats for concatenated expressions are:

value-expression | value-expression

value-expression | | value-expression

If you use | (one vertical line), DATATRIEVE joins the two value expressions to form a longer string, the first character of the second value expression

following the last character of the first value expression. If you use || (two vertical lines), DATATRIEVE removes all trailing spaces from the first value expression and then joins the second value expression to it. The following examples show how DATATRIEVE evaluates concatenated expressions:

```
DTR> PRINT "HELLO"|"THERE"
HELLOTHERE

DTR> PRINT "SINGLE BAR-   "|"LEAVES TRAILING SPACES"
SINGLE BAR-   LEAVES TRAILING SPACES

DTR> PRINT "DOUBLE BAR-   "||"DOES NOT"
DOUBLE BAR-DOES NOT


DTR> PRINT "GOOD"||" BYE"
GOOD BYE
```

The following example shows how concatenation expressions can be used to assign values to long variables. You can use similar assignment statements in a STORE...USING or MODIFY...USING statement to handle long field values.

```
DTR> DECLARE LONG-STRING PIC X(200) EDIT-STRING IS T(50),
DTR> LONG-STRING = *,LINE1|*,LINE2|*,LINE3
Enter LINE1: This line contains the first part of a very long string,
Enter LINE2:   This string is too long to use a character string literal
Enter LINE3:  to assign it, You need concatenation expressions,
DTR> PRINT LONG-STRING

                      LONG
                     STRING

This is the first part of a very long string,
This string is too long to use a character string
literal to assign it, You need concatenation
expressions,

DTR>
```

6.2 Boolean Expressions

Boolean expressions are expressions that DATATRIEVE evaluates to either "True" or "False". You use Boolean expressions in the following places:

- a WITH clause in a record selection expression (Section 6.3)
- an IF-THEN-ELSE statement (Section 5.23)
- a VALID IF clause in a record definition (Section 11.16)

All Boolean expressions contain relational operators and some contain Boolean operators.

6.2.1 Relational Operators

Relational operators compare value expressions, check whether a code is contained in a table, and check whether a record stream is empty. Table 6-4 shows the format for using each relational operator, and explains when the Boolean expression containing it is "True".

Table 6-4: Relational Operators

Boolean Expression Format	Evaluation of Expression
field name { = / EQ / EQUAL } val-exp-1 [,val-exp-2]...	"True" if value of the field is equal to any value expression in the list.
field name { NE / NOT EQUAL / NOT-EQUAL } val-exp-1	"True" if value of the field is not equal to val-exp-1.
field name { > / GT / GREATER-THAN } val-exp-1	"True" if value of the field is greater than val-exp-1.
field name { GE / GREATER-EQUAL } val-exp-1	"True" if value of the field is greater than or equal to val-exp-1.
field name { < / LT / LESS-THAN } val-exp-1	"True" if value of the field is less than val-exp-1.
field name { LE / LESS-EQUAL } val-exp-1	"True" if value of the field is less than or equal to val-exp-1.
field name { BT / BETWEEN } val-exp-1 [AND] val-exp-2	"True" if value of the field is between val-exp-1 and val-exp-2. Val-exp-1 must be less than val-exp-2.
field name { NOT BT / NOT BETWEEN } val-exp-1 [AND] val-exp-2	"True" if value of the field is not between val-exp-1 and val-exp-2. Val-exp-1 must be less than val-exp-2.
field name CONTAINING val-exp-1	"True" if value of the field contains val-exp-1.
field name NOT CONTAINING val-exp-1	"True" if value of the field does not contain val-exp-1.
field name IN table-name	"True" if value of the field is a code in the table.
field name NOT IN table-name	"True" if value of the field is not a code in the table.
ANY rse	"True" if the record stream is not empty.

Most of the relational operators compare field values to value expressions. You can use a prompting value expression instead of a field name. You can also use a variable name. You cannot use any other value expressions.

DATATRIEVE considers lowercase letters to have a greater value than uppercase letters. Within each case letters are sorted alphabetically, thus "ALBIN" is less than "AMERICAN".

The relational operator CONTAINING is case insensitive. DATATRIEVE treats all the characters as uppercase when you use CONTAINING.

The following examples show how to use relational operators which compare field values to value expressions:

```
DTR> FIND YACHTS WITH BEAM EQ 9,10,14
[50 records found]
DTR> PRINT CURRENT WITH RIG NE "SLOOP"

                                  LENGTH
                                   OVER
MANUFACTURER     MODEL    RIG      ALL   WEIGHT   BEAM     PRICE

 EASTWARD        HO       MS       24     7,000    09      $15,900
 FISHER          30       KETCH    30    14,500    09
 GRAMPIAN        34       KETCH    33    12,000    10      $29,675

DTR> PRINT YACHTS WITH LOA BT 30 AND 31

                                  LENGTH
                                   OVER
MANUFACTURER     MODEL    RIG      ALL   WEIGHT   BEAM     PRICE

 ALBIN           BALLAD   SLOOP    30     7,276    10      $27,500
 BOMBAY          CLIPPER  SLOOP    31     9,400    11      $23,950
 CC              CORVETTE SLOOP    31     8,650    09
                          :
                          :
 SOLNA CORP      SCAMPI   SLOOP    30     6,600    10

DTR> PRINT COUNT OF YACHTS WITH BUILDER CONTAINING "a"
  78

DTR> FIND YACHTS WITH LOA < 20
[2 records found]
```

The relational operator IN compares a the contents of a field with the codes in a description table. This is very useful in automatic data validation. The following example shows you can change the record definition PHONE-REC to take advantage of this:

```
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>     02 NAME PIC X(20).
DFN>     02 NUMBER PIC 9(7) EDIT-STRING IS XXX-XXXX.
DFN>     02 LOCATION PIC X(9).
DFN>     02 DEPARTMENT PIC XX VALID IF
DFN>             DEPARTMENT IN DEPT-TABLE.
DFN> ;
```

The relational operator ANY checks whether a record stream is empty. This is very useful if you want to work with records in a hierarchy. The record selection expression following any typically uses a sublist in the hierarchy. The following examples show how to use ANY. For more information on hierarchies and lists, see Chapter 15.

```
DTR> PRINT FAMILIES WITH ANY KIDS WITH AGE = 20

                          NUMBER       KID
  FATHER      MOTHER      KIDS         NAME     AGE

BASIL         MERIDETH      6          BEAU      28
                                       BROOKS    26
                                       ROBIN     24
                                       JAY       22
                                       WREN      17
                                       JILL      20
JEROME        RUTH          4          ERIC      32
                                       CISSY     24
                                       NANCY     22
                                       MICHAEL   20

DTR> PRINT FAMILIES WITH
[Looking for Boolean expression]
DTR> ANY KIDS WITH KID-NAME CONTAINING "RALP"

                          NUMBER       KID
  FATHER      MOTHER      KIDS         NAME     AGE

JIM           ANN           2          URSULA     7
                                       RALPH      3
```

6.2.2 Boolean Operators

There are three Boolean operators: AND, OR, and NOT. AND and OR allow you to join two or more Boolean expressions together to form a single Boolean expression. NOT allows you to reverse the value of a Boolean expression.

If you link Boolean expressions with AND, the resulting Boolean expression is "True" only if all the Booleans linked with AND are "True".

If you link Boolean expressions with OR, the resulting Boolean expression is "True" if any of the Booleans linked with OR are "True".

If you precede a Boolean expression with NOT, the resulting Boolean expression is "True" if the Boolean expression following NOT is "False".

The following examples show the use of Boolean operators:

```
DTR> PRINT YACHTS WITH BUILDER = "PEARSON" AND LOA = 30

                                  LENGTH
                                   OVER
MANUFACTURER     MODEL    RIG      ALL   WEIGHT   BEAM     PRICE

 PEARSON          30      SLOOP     31    8,320    09
```

(continued on next page)

```
DTR> FIND YACHTS WITH BUILDER = "PEARSON" OR LOA EQ 30
[21 records found]
DTR> READY FAMILIES
DTR> PRINT FAMILIES WITH NOT ANY KIDS

                          NUMBER      KID
  FATHER      MOTHER       KIDS       NAME      AGE

ROB           DIDI          0
```

You can also group Boolean expression using parentheses. DATATRIEVE evaluates Boolean expressions in parentheses before evaluating other Booleans. If a Boolean expression contains Boolean operators as well as parentheses, DATATRIEVE evaluates the Boolean expression in the following order:

First: Expressions enclosed in parentheses

Then: Expressions preceded by NOT

Then: Expressions combined with AND

Finally: Expressions combined with OR

The following examples illustrate the use of parentheses and the evaluation of a compound Boolean expression:

bool-1 AND bool-2 AND bool-3	"True" if all three Boolean expressions are "True."
bool-1 AND (bool-2 OR bool-3)	"True" if bool-1 is "True" and either bool-2 or bool-3 is "True."
(bool-1 AND bool-2) OR bool-3	"True" if bool-1 and bool-2 are "True" or if bool-3 is "True."
(bool-1 AND bool-2) OR (bool-3 AND bool-4)	"True" if both bool-1 and bool-2 are "True" or if both bool-3 and bool-4 are "True."
NOT (bool-1 OR bool-2) AND bool-3	"True" if both bool-1 and bool-2 are "False" and bool-3 is "True".

The following example shows several Boolean operators in the same statement.

```
DTR> PRINT YACHTS WITH
[Looking for Boolean expression]
DTR> (MODEL = "BALLAD" AND BUILDER = "ALBIN") OR
[Looking for Boolean expression]
DTR> (BUILDER = "TANZER" AND MODEL = 28)
```

(continued on next page)

```
                                    LENGTH
                                     OVER
MANUFACTURER     MODEL     RIG       ALL   WEIGHT   BEAM     PRICE
 TANZER          28        SLOOP      28    6,800     10      $17,500
 ALBIN           BALLAD    SLOOP      30    7,276     10      $27,500
```

6.3 Record Selection Expressions

A record selection expression (or rse) allows you to create a record stream. The record stream contains only those records that meet your restrictions. The records can come from a domain, a collection, or a list. You can specify whether the record stream contains all records that meet your restrictions or just a few. You can also sort the records in ascending or descending order according to the contents of one or more fields. The full format of a record selection expression is:

```
[ ALL      ]                 { list-name     }
[ FIRST n  ] [context-var IN] { domain-name   } [WITH Boolean-expression]
                              { collectn-name }

[SORTED BY sort-key-1 [,sort-key-2] ...]
```

The rse contains one required element and four optional elements. The following sections describe each element.

6.3.1 Specifying the Source of the Records

Only one element is required in a record selection expression:

list-name

domain-name

collectn-name

This element tells DATATRIEVE what domain, collection, or list to use in creating the record stream.

You must ready a domain (with the READY command) before using it in an rse.

To use a collection in an rse, you must first establish the collection (with the FIND statement).

To use a list in an rse, you must make the record containing the list the context record (with the FOR statement, the SELECT statement, or another rse). Lists and record selection expressions using lists are explained in Chapter 15.

The following example has three record selection expressions: one using the domain FAMILIES, one using the collection CURRENT, and one using the list KIDS:

```
DTR> FIND FAMILIES WITH NUMBER-KIDS > 0
[13 records found]
DTR> FOR CURRENT FOR KIDS MODIFY USING AGE= AGE + 1
```

6.3.2 Specifying the Number of Records

The ALL or FIRST n element specifies how many records are in the record stream. If you include the element FIRST n, the record stream has n records. The argument n must be a value expression that represents a positive integer. If you also specify a sort order in the rse, DATATRIEVE sorts the records which meet your restrictions first. Then the first n records become the record stream.

For example:

```
DTR> PRINT FIRST 3 YACHTS

                                  LENGTH
                                   OVER
MANUFACTURER     MODEL    RIG      ALL   WEIGHT   BEAM     PRICE

 ALBERG        37 MK II  KETCH     37   20,000    12      $36,951
 ALBIN         79        SLOOP     26    4,200    10      $17,900
 ALBIN         NALLAD    SLOOP     30    7,276    10      $27,500

DTR> PRINT FIRST 3 YACHTS SORTED BY LOA

                                  LENGTH
                                   OVER
MANUFACTURER     MODEL    RIG      ALL   WEIGHT   BEAM     PRICE

 WINDPOWER     IMPULSE   SLOOP     16      650    07       $3,500
 CAPE DORY     TYPHOON   SLOOP     19    1,900    06       $4,295
 ENCHILADA     20        SLOOP     20    2,300    07
```

If you use the element ALL, or omit this element, the record stream consists of all records meeting your restrictions. If you use the element FIRST n and n is greater than the number of records meeting your restrictions, the record stream consist of all records meeting your restrictions. Thus, the statements in the following example all have the same result:

```
DTR> FIND ALL YACHTS
[113 records found]
DTR> FIND YACHTS
[113 records found]
DTR> FIND FIRST 200 YACHTS
[113 records found]
```

6.3.3 Naming a Record Stream

If you include a context variable in the rse, you give the record stream a name. This allows you to qualify field names more fully. Using context variables to qualify field names enables you to refer to fields from different record streams in the same statement.

The following examples show how you can use context variables.

```
DTR> FIND A IN YACHTS WITH ANY YACHTS WITH
[Looking for Boolean expression]
DTR> BUILDER= A.BUILDER AND MODEL NE A.MODEL
[76 records found]
DTR> PRINT A IN YACHTS WITH ANY B IN YACHTS WITH
[Looking for Boolean expression]
DTR> B.TYPE NE A.TYPE AND B.SPECS = A.SPECS

                                  LENGTH
                                   OVER
MANUFACTURER     MODEL    RIG      ALL   WEIGHT   BEAM     PRICE

 CARIBBEAN       35       SLOOP     35   18,000    11      $37,850
 CHRIS-CRAF      CARIBBEAN SLOOP    35   18,000    11      $37,850
 SCAMPI          30       SLOOP     30    6,600    10
 SOLNA CORP      SCAMPI   SLOOP     30    6,600    10

DTR> FOR A IN YACHTS
[Looking for statement]
DTR> FOR B IN YACHTS WITH BUILDER EQ A.BUILDER AND RIG GT A.RIG
[Looking for statement]
DTR> PRINT BUILDER, A.RIG, B.RIG

MANUFACTURER  RIG        RIG

 AMERICAN     MS         SLOOP
 CHALLENGER   KETCH      SLOOP
 CHALLENGER   KETCH      SLOOP
       :
       :
 PEARSON      KETCH      SLOOP
```

If you use a context variable in a FIND statement, the context variable becomes the collection name. (The collection actually has two names: CURRENT and the context variable.)

For example, the following statement establishes a collection containing the first four records in YACHTS and names the collection NEW-COLLECT:

```
DTR> FIND FIRST 4 NEW-COLLECT IN YACHTS
[4 records found]
```

6.3.4 Restricting the Records

The element WITH Boolean-expression restricts the record stream to those records for which the Boolean expression is true.

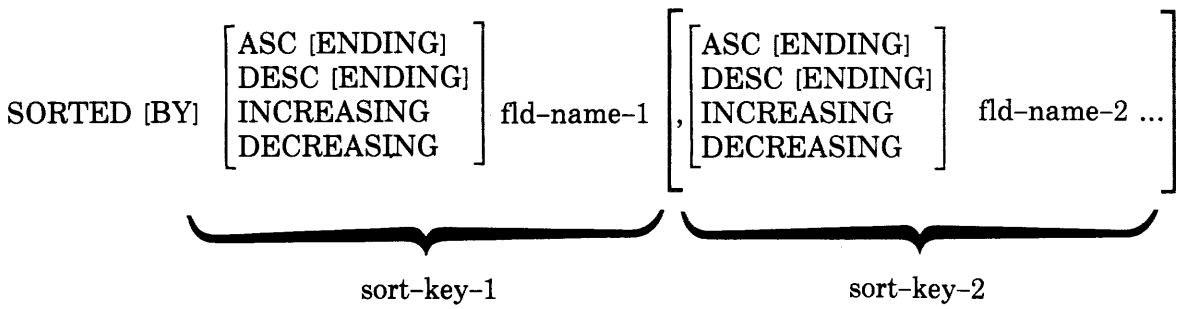
For example, the following statement prints only yachts built by Grampian:

```
DTR> PRINT YACHTS WITH BUILDER EQ "GRAMPIAN"

                                    LENGTH
                                     OVER
MANUFACTURER     MODEL    RIG       ALL   WEIGHT   BEAM     PRICE

 GRAMPIAN        2-34     SLOOP      34   11,800    10      $29,675
 GRAMPIAN        26       SLOOP      26    5,600    08      $11,495
 GRAMPIAN        28       SLOOP      28    6,900    10      $14,475
 GRAMPIAN        30       SLOOP      30    8,600    09      $17,775
 GRAMPIAN        34       KETCH      33   12,600    10      $29,675
```

Refer to Section 6.2 for other examples of Boolean expressions and for the rules for forming Booleans.

6.3.5 Sorting the Records

The SORTED BY element of the rse lets you sort the records based on the contents of one or more fields. Its full format is:



To sort records, you must specify the keyword SORTED. The keyword BY is optional; it is included only to preserve English-language-like syntax. You must specify at least one sort key, which indicates both the order of the sort and the field to be used in the sort. If you specify a sort order, it can be ASCENDING (or its synonyms ASC or INCREASING) or DESCENDING (or its synonyms DESC or DECREASING). The sort order is described in Appendix F.

If you omit the sort order for the first field, DATATRIEVE sorts the records in ascending order. If you omit the sort order on the second or subsequent field, DATATRIEVE uses the preceding sort order (implied or specified) for the field.

For example, the following statements are equivalent: They all establish a collection of all records in YACHTS and sort them in ascending order by builder name:

```
DTR> FIND YACHTS SORTED BY MANUFACTURER
[113 records found]

DTR> FIND YACHTS SORTED BY ASC MANUFACTURER
[113 records found]

DTR> FIND YACHTS SORTED BY INCREASING MANUFACTURER
[113 records found]
```

You can specify any number of sort keys. If you specify more than one, you must separate the sort keys with a comma. DATATRIEVE treats the first field name as the major sort key and successive field names as minor keys. For instance, the following statement establishes a collection of all records in YACHTS. The records are sorted in alphabetical order by builder name. If a builder has more than one record for a yacht, the records are listed in decreasing order of price:

```
DTR> FIND YACHTS SORTED BY BUILDER, DESC PRICE
[113 records found]
```

Chapter 7
Using the Report Writer

7.1 Introduction

You can produce simple reports with the PRINT and SUM statements, but, to produce complex reports, use the DATATRIEVE Report Writer.

Like the PRINT statement, the Report Writer selects and formats data; it also performs statistical functions, including that of the SUM statement. In addition, it can number the report pages, print the date at the top of each page, and give you control over several other formatting capabilities. For example, you can:

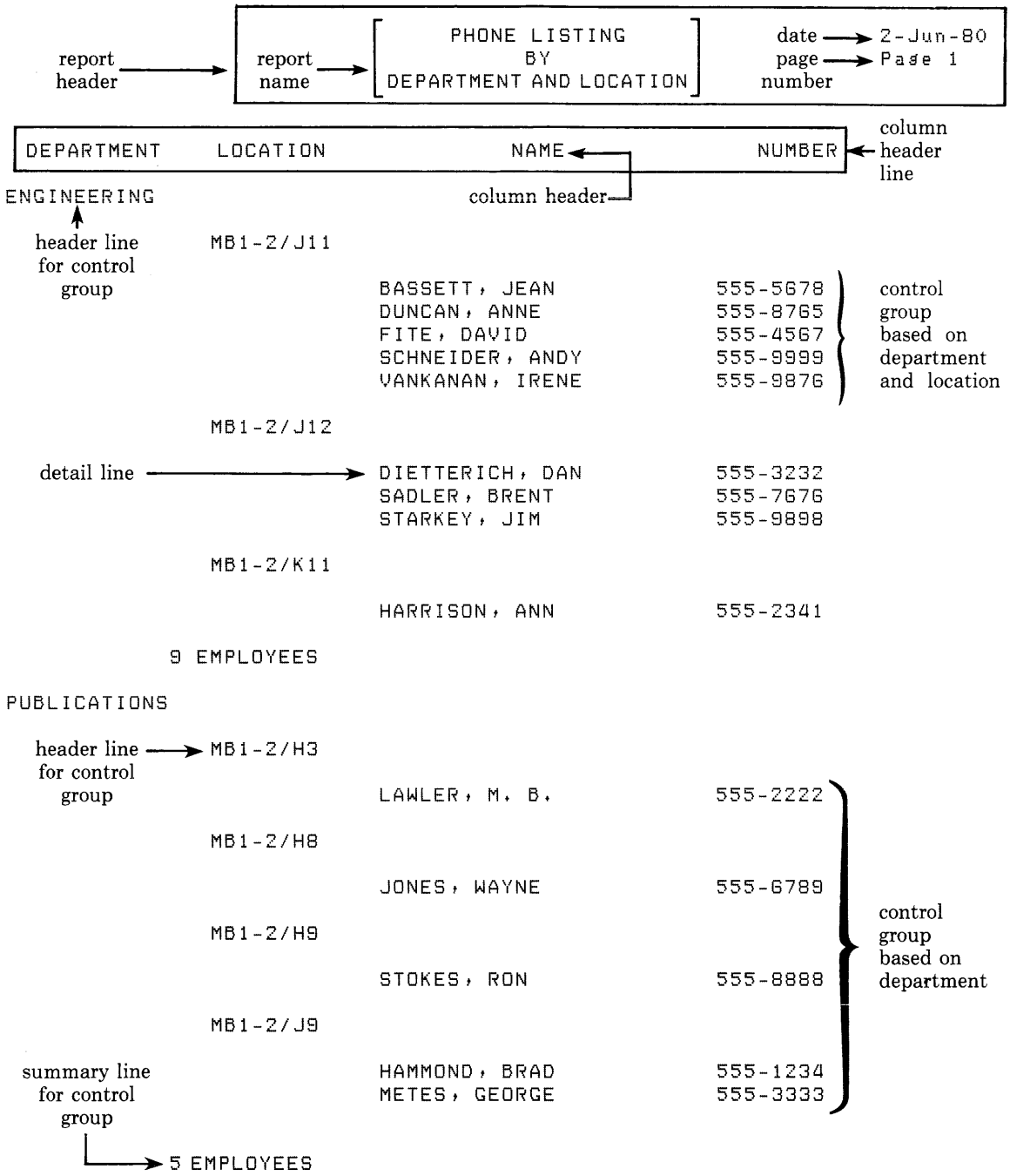
1. Print a heading for the report.
2. Set the page size of the report.
3. Print sorted data that is grouped by values in one field and is sorted within each group by the values in another field.
4. Print headings above groups of sorted data.
5. Print totals, averages, counts, minimums, and maximums below groups of sorted data.

This chapter describes how to use the Report Writer statements to create DATATRIEVE reports.

7.2 A Sample DATATRIEVE Report

A DATATRIEVE report looks like a typewritten report; Figure 7–1 is an example.

Figure 7-1: A DATATRIEVE Report



7.3 Creating a Report Specification

You create a DATATRIEVE report with a series of Report Writer statements. These control the format and select the content of a report. The order of statements in a report specification is important; although the order is not absolutely rigid, errors in the sequence of statements can either prevent the production of the report or result in a report with a jumbled format.

7.3.1 Choosing Direct or Indirect Creation of a Report Specification

You can create a report specification in two ways:

1. Directly, by invoking the Report Writer and entering statements in response to its prompts.
2. Indirectly, by putting Report Writer statements into a procedure.

The differences between these two are important. When you work directly (that is, interactively) with the Report Writer to create report specifications, responses to your input are immediate. The Report Writer creates the report as soon as you end the report specification. When you make syntax errors, a message indicates the problem, and the Report Writer returns you to the DATATRIEVE command level. These immediate responses are useful for creating short report specifications.

However, after the Report Writer creates a report or sends you an error message, it does not save the statements you have entered. You cannot run the same specification again, and you cannot correct syntax errors unless you retype the whole sequence of report statements. To avoid this repetition, many users put their report specifications into procedures.

Although writing procedures is discussed in Section 5.9 and Chapter 12, two points about procedures need to be made here:

1. The form and sequence of report statements in a procedure are identical to their form and sequence in a report specification that you create interactively.
2. Putting report specifications in procedures offers you several advantages, all of which prevent retyping the whole sequence of report statements:
   - You can rerun a procedure.
   - You can change report specifications to alter the format and content of reports.
   - You can change syntax errors by using the DATATRIEVE Editor (see Chapter 8) on the faulty report statements.
   - You can share procedures with other users.

For each report, you can decide whether to create the report specification interactively or to put it into a procedure. In either case, the sequence of report statements and their syntax will be the same.

7.3.2 Invoking the Report Writer

When you are at the DATATRIEVE command level, you can invoke the Report Writer with the REPORT command.

```
DTR> REPORT BIG-YACHTS ON YACHTS.LST
RW>
```

The RW> prompt shows that the Report Writer is ready to accept the statements of the report specification.

On the other hand, you can call a procedure that contains a report specification with the REPORT command as its first line.

- First, put a report specification into a procedure:

```
DTR> DEFINE PROCEDURE SAMPLE-REPORT
DFN> REPORT BIG-YACHTS ON YACHTS.LST
DFN>            .
DFN>            .
DFN>            .
DFN> END-REPORT
DFN> END-PROCEDURE
DTR>
```

- Then, invoke the Report Writer indirectly by calling the procedure:

```
DTR> :SAMPLE-REPORT
```

  If the report specification contains any prompts, the Report Writer prompts you for the needed information. Then, if you have directed the output of the report to your terminal, the formatted text is displayed as directed. If you have directed the output to a file or another device, the Report Writer returns you to the DATATRIEVE command level, and you see the DTR> prompt.

In summary, the DATATRIEVE REPORT command invokes the Report Writer. Remember, whether entered directly as a response to DTR> prompts or indirectly as part of a procedure, the REPORT command must be the first entry in the report specification.

7.3.3 The Order and Function of Report Statements — An Overview

The following list gives a brief overview of the REPORT command and each Report Writer statement; the list follows the normal order of statements in a report specification.

Note that AT and SET statements can be entered both before and after the PRINT statement. Each report statement is fully described in Section 7.4.

1. REPORT

   Always begin a report specification with a REPORT command. You enter it in response to the DTR> prompt at the DATATRIEVE command level or put it as the first entry of a report specification in a procedure. The REPORT command invokes the Report Writer, and it is part of the report specification. With REPORT you specify the data you want to report and the file or device for output. The default data for the report is your current collection, and the default output device is your terminal.

2. SET

   Use SET to specify the report header, which consists of the report name, the page number, and the date. Also use SET to specify the length and width of the report pages and to limit the maximum number of lines and pages a report. With SET you can suppress the date and the page number in the report header. Section 7.4.2 explains the defaults and options of the SET statement. The Report Writer supplies default values for page-size, date, and number of lines per page, and it sets no limits to the number of lines or pages in a report. However, you must always specify a report name. If you omit the SET REPORT-NAME statement, the Report Writer prompts you for one before creating the report. You can suppress the report name only by setting it equal to one or more spaces.

3. AT

   The AT statement contains most of the special features that set the Report Writer apart from the PRINT statement used at the DATATRIEVE command level. In addition to these special features, the AT statement contains a PRINT clause that resembles the PRINT statement used at the DATATRIEVE command level.

   The AT statement has two forms: AT TOP and AT BOTTOM. Use AT TOP statements to print either header lines or summary lines at the tops of reports, pages, and control groups. (See the next section for a description of control groups and their uses.) These lines can contain values and text, including special column headers.

   The AT BOTTOM statement resembles AT TOP, but you use it to print summary lines at the bottoms of reports, pages, and control groups. In reports you can use the statistical functions in AT BOTTOM statements. You can also specify special column headers for values you want to include in the summary lines. To create a report specification for a summary report, use an AT BOTTOM statement and omit the Report Writer PRINT statement.

   Both forms of the AT statement allow you to control the formats of headers and summary lines. You can include several AT statements in a report specification, or you can leave them out.

4. PRINT

   The Report Writer PRINT statement is similar to the PRINT statement used at the DATATRIEVE command level. Use PRINT to specify the format of the reported data. With the PRINT statement you specify two report parts: the column headers, which are printed above the data, and the detail lines, which contain the information you want to report and which may be values, text, or a mixture of both. You can include only one PRINT statement in a report specification, but you can leave out the PRINT statement if your specification includes an AT statement.

5. AT

   If your report specification contains a PRINT statement, you can enter AT statements both before and after the PRINT statement.

   NOTE

   Every report specification must include at least one PRINT statement or one AT statement. Each specification can contain only one PRINT statement, but you can include several AT statements in the same specification. If you use an AT BOTTOM statement in a report specification without a PRINT statement, you will produce a summary report, not a fully detailed one.

6. SET

   You can put SET statements anywhere in a report specification between the REPORT command and the END-REPORT statement.

7. END-REPORT

   To end a report specification, you must use END-REPORT as your last report statement. If your report specification contains no syntax errors, the Report Writer begins to create the report after you enter the END-REPORT statement. The Report Writer prompts you for any values it may need and then directs the report to the file or output device you have specified.

7.3.4 Using Control Groups

With the Report Writer you can divide collections of sorted data into control groups. A control group is a series of sorted data records that all have the same value in at least one field. For example:

The domain YACHTS contains eleven boats with LENGTH-OVER-ALL (LOA) of 36 or 37. Those boats have:

- Two values for BEAM: 11 and 12
- Two values for RIG: KETCH AND SLOOP

If you sort these eleven boats by BEAM, you form two control groups:

- One of six boats with BEAM = 11
- One of five boats with BEAM = 12

If you sort the first control group of six boats by the key field LOA, you form two more control groups:

- One of two boats with LOA = 36 (and BEAM = 11)
- One of four boats with LOA = 37 (and BEAM = 11)

This second control group (with BEAM = 11 and LOA = 37) can be further divided into two control groups:

- KETCHES (BEAM = 11, LOA = 37)
- SLOOPS (BEAM = 11, LOA = 37)

This example shows that you can nest control groups; that is, you can form control groups inside other control groups. You form these nested control groups by using one or more key fields to sort your data.

You can establish the sort order of your data with the REPORT command; however, you can sort the data with the DATATRIEVE SORT command or with a sort utility outside of DATATRIEVE. If the data has been sorted prior to entering the report specification, the Report Writer makes the breaks between control groups just as though you had established the sort order with the REPORT command. (See Section 7.4.4, Usage Note 3.)

Control groups give you a useful way of dividing data in a report. With the AT TOP and AT BOTTOM statements you can print text, values, and headers above and below each nested control group. The following example shows a report specification and report that divides 36 and 37 foot yachts into control groups based on BEAM, LOA, and RIG (the Report Writer prompt for line continuation have been omitted from this and succeeding examples):

```
DTR> READY YACHTS
DTR> FIND YACHTS WITH LOA BETWEEN 36 AND 37
DTR> REPORT CURRENT SORTED BY BEAM, LOA, RIG
RW>  SET REPORT-NAME = "YACHTS WITH LENGTH-OVER-ALL"/
RW>    "OF 36 AND 37 FEET"
RW>  AT TOP OF BEAM PRINT COL 1, "BEAM = ", COL 7, BEAM
RW>  AT TOP OF LOA PRINT COL 12, "LENGTH = ", COL 20, LOA, SKIP
RW>  PRINT BUILDER, RIG, DISP
RW>  AT BOTTOM OF RIG PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
RW>    LOA(" "), COL 17, "FOOT ", COL 22, RIG, COL 28,
RW>    "WITH BEAM OF ", COL 40, BEAM, COL 43, " = ", COL 46,
RW>    COUNT USING Z9, SKIP
RW>  AT BOTTOM OF REPORT PRINT SKIP, "LIGHTEST = ", MIN (DISP),
RW>    SKIP, "HEAVIEST = ", MAX (DISP), SKIP,
RW>    "AVERAGE WEIGHT OF ALL BOATS = ", AVERAGE (DISP)
RW>  SET DATE = "DD-MMM-YY"
RW>  SET COLUMNS-PAGE = 60
RW>  END-REPORT
```

(continued on next page)

YACHTS WITH LENGTH-OVER-ALL
OF 36 AND 37 FEET

DD-MMM-YY
Page 1

BEAM	LENGTH OVER ALL	MANUFACTURER	RIG	WEIGHT
BEAM = 11				
	LENGTH = 36			
		PEARSON	KETCH	17,700
NUMBER OF 36 FOOT KETCH WITH BEAM OF 11 = 1				
		ISLANDER	SLOOP	13,450
NUMBER OF 36 FOOT SLOOP WITH BEAM OF 11 = 1				
	LENGTH = 37			
		IRWIN	KETCH	20,000
		NORTHERN	KETCH	14,000
NUMBER OF 37 FOOT KETCH WITH BEAM OF 11 = 2				
		PEARSON	SLOOP	13,500
		ROBERTS	SLOOP	14,750
NUMBER OF 37 FOOT SLOOP WITH BEAM OF 11 = 2				
BEAM = 12				
	LENGTH = 36			
		I. TRADER	KETCH	18,600
NUMBER OF 36 FOOT KETCH WITH BEAM OF 12 = 1				
		CABOT	SLOOP	15,000
		ERICSON	SLOOP	16,000
NUMBER OF 36 FOOT SLOOP WITH BEAM OF 12 = 2				
	LENGTH = 37			
		ALBERG	KETCH	20,000
		FISHER	KETCH	30,000
NUMBER OF 37 FOOT KETCH WITH BEAM OF 12 = 2				
		LIGHTEST =		13,450
		HEAVIEST =		30,000
		AVERAGE WEIGHT OF ALL BOATS =		17,545

7.4 Report Writer Statements

This section describes the statements that make up a report specification.

7.4.1 REPORT Command

Function

The REPORT command invokes the Report Writer and is the first entry in a report specification. In the REPORT statement you specify:

1. The data you want to report.
2. The output device for the report.

After you enter the REPORT command, the RW> prompt shows that the Report Writer is ready to accept the statements of a report specification.

Format

```
REPORT [rse] [ ON { dev:           } ]
             [    { file-spec      } ]
             [    { *.prompt-name  } ]
```

Arguments

rse

Specifies the data for your report. To create a collection for your report, enter the appropriate record selection expression in the REPORT command. You can make reports on the following kinds of data:

- Readied domains
- Collections
- Lists

When you omit the rse, the Report Writer uses the data in your current collection for the report. If you omit the rse and you have no current collection, the Report Writer tells you there is no current collection only after you have entered the entire report specification.

For detailed information on the record selection expression, see Chapter 6. For information on lists, see Chapter 15.

dev:

Specifies the output device for your report. By specifying a device name in the REPORT command, you can tell the Report Writer to send the report to any system device to which you have access, such as a line printer, a tape drive, or your own terminal.

To specify an output device for a report:

- A line printer:

```
DTR> REPORT [rse] ON LP:
```

On RSTS/E systems, if you specify the line printer for output and it is busy, the Report Writer may not produce the report. Furthermore, the Report Writer does not save the report specification when the line printer is busy, and you must reenter the report specification. If no report is produced for this reason, you may have to consult with your system manager to work around the problem. To save yourself from retyping report specifications, put them into procedures.

- A tape drive:

```
DTR> REPORT [rse] ON mmn:file-spec
```

where:

mmn = tape drive number
file-spec = the specification of the destination file

Before specifying the tape drive as the output device for your report, you must mount your tape and assign the tape drive to your task or process.

- Your own terminal:

```
DTR> REPORT [rse] ON TI:
```

You can specify you own terminal as the output device by entering TI: as the device name. Because your terminal is the default output device, you do not need to specify it explicitly.

file-spec

Specifies a disk file for the output of the report. By supplying a file specification in the REPORT command, you tell the Report Writer to send the report to a disk file. Putting a report in a disk file allows you to revise the text, to print multiple copies of your report, or to direct your output to the printer at another time.

To specify a disk file for output of a report:

- In your own directory:

```
DTR> REPORT [rse] ON YACHTS.LST
```

- In another user's directory:

```
DTR> REPORT [rse] ON ddn:[ufd]YACHTS.LST
```

where:

ddn = disk drive number
ufd = user file directory

You must have access privilege to the destination directory.

To revise your report, use a text editor to make the necessary changes to your disk file.

To print multiple copies of your report, supply the REPORT command with a file specification for the output of the report:

```
DTR> REPORT ON YACHTS.LST.
```

After you exit from DATATRIEVE, print the file using your operating system routines and specify the number of copies you want.

*.prompt-name

Specifies the text of a prompt for a device name or file specification. If you supply *.prompt-name in the REPORT command, the Report Writer prompts you for an output file-spec or device name after you enter the END-REPORT statement:

```
DTR> REPORT [rse] ON *."FILE-SPEC OR DEVICE NAME"
RW>      .
RW>      .
RW>      .
RW>  END-REPORT
Enter FILE-SPEC OR DEVICE NAME:
```

When the Report Writer prompts you, it supplies the word "Enter". Make the text of your prompt-name a word or phrase that completes the prompt's request:

- Wrong:

```
DTR> REPORT ON *."PLEASE SUPPLY FILE-SPEC"
                .
                .
                .
          RW>  END-REPORT
Result:   Enter PLEASE SUPPLY FILE-SPEC:
```

- Right:

```
DTR> REPORT ON *.FILE-SPEC
               .
               .
               .
         RW>  END-REPORT
Result:  Enter FILE-SPEC:
```

If your prompt-name contains any spaces, you must enclose it in quotation marks. For example:

- Wrong:
```
DTR> REPORT ON *.FILE-SPEC OR DEVICE NAME
```
  Result:
```
Expected end of report statement, encountered "OR"
```
- Right:
```
DTR> REPORT ON *."FILE SPECIFICATION OR DEVICE NAME"
        .
        .
        .
RW>  END-REPORT
```
  Result:
```
Enter FILE SPECIFICATION OR DEVICE NAME
```

When you do not specify an output device in the REPORT command, the Report Writer displays the report on your terminal.

You can specify only one output device in a REPORT command.

Examples

1.
```
DTR> REPORT YACHTS WITH PRICE NE 0 SORTED BY
DTR> RIG ON RIG.LST
```

2.
```
DTR> REPORT ALL YACHTS WITH BUILDER = "PEARSON" AND RIG
DTR> NE "KETCH" SORTED BY DESCENDING BEAM, DESC LOA ON TI:
```

3.
```
DTR> REPORT FIRST 10 YACHTS WITH RIG = "KETCH"-
DTR> SORTED BY BEAM, PRICE ON *."DEVICE NAME"
```

7.4.2 SET Statement

Function

The Report Writer SET statement produces the report header and defines the size of report pages and the length of the report. In a SET statement you can specify:

1. Report header
   - Report name
   - Date
   - Format of date
   - Page numbering
2. Size of report pages
   - Number of columns per page
   - Number of lines per page
3. Length of report
   - Maximum number of lines
   - Maximum number of pages

Format

```
SET { REPORT-NAME = "string-1" ["string-2"] . . .
    { DATE [ = "string"]
    { NO DATE
    { NUMBER
    { NO NUMBER
    { COLUMNS-PAGE =
    { LINES-PAGE =     { n                  }
    { MAX-LINES =      { *. prompt-name     }
    { MIN-LINES =
    } [, . . . ]
```

Arguments

REPORT-NAME = "string-1" ["string-2"] . . .

Specifies the name of your report. The report name consists of one or more lines of text, and the Report Writer centers this name at the top of each page of the report. You must enclose the report name in quotation marks:

```
RW> SET REPORT-NAME = "LISTING OF BOATS BY LENGTH AND TYPE"
```

Result: `LISTING OF BOATS BY LENGTH AND TYPE`

To produce a report name of two or more lines, enclose each line of the report name in quotation marks and separate each segment of the name by slashes (/):

```
RW> SET REPORT-NAME = "LISTING OF BOATS"/"BY"/"LENGTH AND TYPE"
```

Result:
```
LISTING OF BOATS
          BY
LENGTH AND TYPE
```

If you omit the SET REPORT-NAME statement from the report specification, the Report Writer prompts you for a report name after you enter the END-REPORT statement:

```
RW> END-REPORT
Enter REPORT-NAME:
```

When you respond to this prompt, follow the above rules for the use of quotation marks and slashes in report names.

To produce a report with no name, set the report name equal to one or more spaces:

```
RW> SET REPORT-NAME = "   "
```

or

```
Enter REPORT-NAME: "   "
```

There is no default report name.

DATE [= "string"]

Can be used to specify a date in the top right corner of each report page. When you specify a string in the SET DATE statement, you can choose any format for the date. You can also use the SET DATE statement to print any single line string in the top right corner of each report page. The string does not have to contain a date:

```
RW> SET DATE = "22 MAR 1981"
```

Result:
```
22 MAR 1981
```

```
RW> SET DATE = "TUESDAY, MARCH 14TH"
```

Result:
```
TUESDAY, MARCH 14TH
```

```
RW> SET DATE = "COMPANY CONFIDENTIAL"
```

Result:
```
COMPANY CONFIDENTIAL
```

If you supply no string or omit the SET DATE statement from the report specification, the default value is the current system date in the format DD-MMM-YY (21-Jan-81).

NO DATE

Suppresses the printing of a date in the top right corner of each report page. If you put the SET NO DATE statement in a report specification, either before or after a SET DATE statement, the SET NO DATE takes precedence, and the string from the SET DATE statement is not printed.

NUMBER

Prints the page numbers in the top right corner of the report page, below the date or the string you specify in a SET DATE statement. If the report specification contains a SET NO DATE statement, the page number is printed by itself in the top right corner of each report page:

```
RW> SET NUMBER
RW> SET DATE
```

Result:
```
22-Mar-81
     Page 1
```

```
RW> SET NO DATE
RW> SET NUMBER
```

Result:
```
Page 1
```

Because the Report Writer numbers the pages whether or not you supply a SET NUMBER statement, you never need put SET NUMBER in a report specification.

NO NUMBER

Suppresses the printing of page numbers in the top right corner of each report page. If you put SET NO NUMBER in a report specification, either before or after a SET NUMBER statement, the SET NO NUMBER statement takes precedence, and no page numbers are printed.

COLUMNS–PAGE = { n | *.prompt-name }

Specifies the width of the report page in columns. The value n is the number of columns per page. It must be an unsigned integer one greater than the length of the longest field in the record definition of the data you want to report. For example, in YACHTS the longest field is BUILDER with a field length of 10. You must set COLUMNS–PAGE = 11 to print a report containing the field BUILDER as the longest field. The maximum value for COLUMNS–PAGE is 255.

If you use *.prompt-name in place of the unsigned integer n, the Report Writer prompts you for a COLUMNS–PAGE value after you enter an END–REPORT statement:

```
RW>  SET COLUMNS-PAGE = *.COL-PAGE
```

Result:
```
Enter COL-PAGE:
```

If you omit a SET COLUMNS–PAGE statement from your report specification, the default value for your report is the COLUMNS–PAGE value of the DATATRIEVE command level at the time the report is produced. If you have not specified a COLUMNS–PAGE value at the DATATRIEVE command level, the default value is the one set when your DATATRIEVE

system was installed. Ask your DATATRIEVE system manager for the default COLUMNS-PAGE value.

LINES-PAGE = { n | *.prompt-name }

Specifies the number of lines per report page. To print more than one detail line per report page, the value n must be an unsigned integer greater than the total number of lines in the report header, the column-headers, the first detail line, the summary line, and the blank lines following the report header and the column headers. Each of these components can have more than one line, depending on the formats you specify in the appropriate Report Writer statements. Do not count the blank lines you specify with SKIP [n] in PRINT or AT statements. The count of a simple report follows.

Here is the report specification:

```
DTR> READY YACHTS
DTR> FIND YACHTS WITH BUILDER = "ALBIN"
DTR> REPORT
RW>  SET DATE = "21-OCT-81"
RW>  SET COLUMNS-PAGE = 40
RW>  SET REPORT-NAME = "YACHTS"/"BY"/"ALBIN"
RW>  PRINT RIG, LOA, PRICE("SUGGESTED"/"PRICE")
RW>  AT BOTTOM OF REPORT PRINT SKIP, "BUY ONE TODAY !"
RW>  SET LINES-PAGE = 13
RW>  END-REPORT
```

The appropriate lines for a report with one record per page are numbered at the right:

```
                    YACHTS                          1
                      BY            21-OCT-81       2
                    ALBIN           Page 1          3
                                                    4
                                                    5
                        LENGTH                      6
                         OVER     SUGGESTED         7
 RIG                     ALL       PRICE            8
                                                    9
 SLOOP                   26       $17,900          10
 SLOOP                   27       $18,600          (Not second record)
 SLOOP                   30       $27,500          (Not third record)
                                                    (Not this SKIP line)
BUY ONE TODAY !                                     11
```

- SET LINES-PAGE = 13 or greater produces a one-page report with all three records.
- SET LINES-PAGE = 12 produces a two-page report with two records on the first page and one on the second.
- SET LINES-PAGE = 11 produces a three-page report with one record per page.

If you omit a SET LINES-PAGE statement from your report specification, the default value is 60 lines per page.

If you put two or more SET LINES-PAGE statements in a report specification, the last one takes precedence.

If your report contains a list as an element in a PRINT statement or an AT statement, the entire list is counted as one detail line. Consequently, a report that prints a list may overflow the report page.

The upper limit of the value you can assign to n is 32,767.

If you use *.prompt-name in place of the unsigned integer n, the Report Writer prompts you for a LINES-PAGE value after you enter an END-REPORT statement.

```
RW>  SET LINES-PAGE = *.LINES-PG
```

Result:

```
Enter LINES-PG
```

MAX-LINES = { n | *.prompt-name }

Specifies the maximum limit on the number of lines in your report. The value n should be an unsigned integer. The Report Writer counts all lines of the report, including the blank lines and the form feeds (one at the top of each page and one at the bottom of the report). The upper limit of the value you can assign to n is 32,767.

The SET MAX-LINES statement can prevent any infinite loops in report specifications from tying up system resources. The Report Writer truncates the report when the number of lines in the output equals the number you have specified in the SET MAX-LINES statement. When the report exceeds the specified limit, you receive this message:

```
Maximum number of lines exceeded-report aborted
Execution failed
DTR>
```

When the report exceeds the maximum number of lines, the truncated report is displayed on your terminal or sent to the file or device you have specified in the REPORT command. If you have specified a disk file for the output, the Report Writer sends only the truncated report to the file and does not include the error message noted above.

If you put two or more SET MAX-LINES statements in a report specification, the last one takes precedence.

You can use *.prompt-name in place of the unsigned integer n, and the Report Writer will prompt you for a MAX-LINES value after you enter an END-REPORT statement.

```
RW>  SET MAX-LINES = *.MAX-LINES
```

Result:

```
Enter MAX-LINES:
```

MAX-PAGES = { n | *.prompt-name }

Specifies the maximum limit on the number of pages in your report. The value n should be an unsigned integer. The upper limit of the value you can assign to n is 32,767.

The SET MAX-PAGES statement can also prevent any infinite loops in report specifications from tying up system resources. The Report Writer truncates the report when the number of pages in the output equals the number you have specified in the SET MAX-PAGES statement. When the report exceeds the specified limit, you receive this message:

```
Maximum number of pages exceeded-report aborted
Execution failed
DTR>
```

When the report exceeds the maximum number of pages, the truncated report is displayed on your screen or sent to the file or device you have specified in the REPORT command. If you have specified a disk file for the output, the Report Writer sends only the truncated report to the file and does not include the error message noted above.

If you put two or more SET MAX-PAGES statements in a report specification, the last one takes precedence.

You can use *.prompt-name in place of the unsigned integer n, and the Report Writer will prompt you for a MAX-PAGES value after you enter an END-REPORT statement.

```
RW>  SET MAX-PAGES = *.MAX-PAGES
```

Result:

```
Enter MAX-PAGES:
```

7.4.3 PRINT Statement

Function

The Report Writer PRINT statement specifies the format for the detail lines in a report. Detail lines are the formatted data lines of a report. With PRINT you specify three characteristics of the detail lines:

1. The content of the detail lines
   - Values of data fields from records in the current collection or in the record stream specified by the rse in the REPORT command
   - Other desired values and text strings
2. The format of the fields in the detail lines
   - Position within the line
   - Edit-string for each field
3. The column headers for the fields in the detail line

The Report Writer PRINT statement produces a detail line in the report for every data record in the current collection or in the record stream. A detail line can cover several lines on the report page, depending on the format and columns per page you specify.

You can include only one PRINT statement in a report specification. If the specification contains no AT statements, then it must contain a PRINT statement. If your report specification contains an AT statement, then it does not have to contain a PRINT statement.

Format

PRINT print-list-element-1 [,print-list-element-2] . . .

Arguments

print-list-element

Specifies the data, its position and format, in the detail line. Table 5–5 describes the available print-list elements in detail. Of the elements listed in that table, only TAB [n] cannot be used in a Report Writer PRINT statement.

Unlike the PRINT statement used at the DATATRIEVE command level, the Report Writer PRINT statement must be followed by at least one print-list element. If you enter PRINT without a print-list element, the Report Writer prompts you for one:

```
RW> PRINT
[Looking for a value expression]
```

The print-list elements you can specify include field names (including COMPUTED BY fields), value expressions, character strings, and horizontal and vertical spacing instructions.

Usage Notes

1. When the data you are reporting includes a list, use an inner print-list in the PRINT statement to specify a format for the list. The format of the list can differ from that of the rest of the data. For details on specifying an inner print-list, see Section 5.29 on the PRINT statement used at the DATATRIEVE command level. The rules described there also apply to inner print-lists in the Report Writer PRINT statement.

2. For each data record in the current collection, the Report Writer produces the detail line by using the print-list elements in the same left-to-right order you specify.

3. If you specify a group field name as a print-list element, the Report Writer includes all fields of the group field in the detail line. The order of those fields follows that in the record definition, and the Report Writer spaces the fields evenly across the page. If the value of COLUMNS–PAGE is too small to accommodate all the fields, the Report Writer carries the overflow fields onto the next line. No field is split between lines, but the column headers of the overflow fields can be lost.

4. If the detail line occupies more than one line in the report, the Report Writer checks ahead to make sure that all the lines of the detail line can be accommodated on the page. If there is not enough room for the whole unit, the Report Writer carries all the lines of the detail line over to the

next page, ignoring the LINES-PAGE value. Lists are exceptions; regardless of length they are counted as one line, and can overflow the report page.

5. If you do not specify a column header for a field in the PRINT statement, the header is the field name from the record definition of the data in the current collection or record stream. The query header, if the record definition contains one, is used as the column header, unless you explicitly define an alternative.

6. To print all the fields of the data in the current collection or record stream, specify in the print-list the most inclusive group field name (the group field name with the lowest level number in the record definition). For example, in the record definition for YACHT, BOAT is the most inclusive group field:

```
RECORD YACHT
USING
01 BOAT
     .
     .
     .
```

   To print all the fields in YACHT, enter the following statement in your report specification:

```
RW>  PRINT BOAT
```

7. If you do not specify positions or edit-strings for any of the fields in a detail line, the Report Writer determines the format for those fields using these criteria:

   - If the record definition contains an edit-string for the field, that edit-string determines the format for the field.
   - If the record definition has no edit-string for the field, the picture clause determines the format for the field.
   - If the record definition has neither an edit-string nor a picture clause for the field, the the Report Writer invents a picture clause to accommodate the data in the field. For instance, in the AT BOTTOM statement, the Report Writer invents a picture for values supplied by the statistical function TOTAL, but the picture has no commas and is four digits longer than the picture of the field whose values it adds. To gain full control over the formats of the fields of your detail lines, explicitly define edit-strings with the USING edit-string modifier.

8. In formatting the detail line, you are not restricted to the order of fields specified in the record definition. You can specify a different order of fields by listing the fields (and their column headers and edit-strings) in the PRINT statement in the desired order. You can also change the order of fields by interspersing value expressions between fields from the records in the current collection.

9. You can use SKIP [n], SPACE [n], and COL n in the Report Writer to specify the position of fields on the report page. Use COL n, not SPACE [n], to position the print-list element first specified in the PRINT statement.

Examples

1. Print the BUILDER, MODEL, DISP, and PRICE fields in the default format. The column header for BUILDER will be MANUFACTURER (the field name), and for DISP will be WEIGHT (the query header for the field):

```
RW>  PRINT BUILDER, MODEL, DISP, PRICE
```

2. Print nine spaces, the fields PRICE and DISP, then the result of the value expression PRICE/DISP under the heading PRICE PER POUND.

```
RW>  PRINT COL 9, DISP, PRICE,
RW>  PRICE/DISP ("PRICE PER POUND") USING $ZZ,ZZZ.99
```

3. Print the model, current price, and a new price 10% higher than the current price, and print the column headers CURRENT PRICE and SUGGESTED PRICE each on two lines:

```
RW>  PRINT MODEL, PRICE("CURRENT"/"PRICE"),PRICE*1.1("SUGGESTED"/"PRICE")
```

7.4.4 AT Statement

Function

The AT statement prints header and summary lines at the tops and bottoms of units of sorted data:

- The AT TOP statement prints header or summary lines at the top of reports, report pages, and control groups (see Section 7.3.4 for details on using control groups). You can include values, text, and special column headers in these header or summary lines.
- The AT BOTTOM statement prints summary lines at the bottoms of reports, report pages, and control groups. You can include values, text, special column headers, and statistical values in these summary lines.

For these header and summary lines, you can also specify the following with the two forms of the AT statement:

- Position of fields within the lines
- Format of fields within the lines

Format

```
                  { REPORT }                  { header-element-1   }[,header-element-2]...
AT TOP OF         { PAGE   } PRINT            {                    }
                  { fld-name }                { summary-element-1  }  [,summary-element-2]...

                  { REPORT }
AT BOTTOM OF      { PAGE   } PRINT  summary-element-1  [,summary-element-2]...
                  { fld-name }
```

Arguments

REPORT

1. In an AT TOP statement, prints the header or summary line at the top of the first page of the report.
2. In an AT BOTTOM statement, prints the summary line at the bottom of the last page of the report.

PAGE

1. In an AT TOP statement, prints the header or summary line at the top of each page of the report.
2. In an AT BOTTOM statement, prints the summary line at the bottom of each page of the report.

fld-name

1. In an AT TOP statement, prints the header or summary line at the top of the control group for which the field-name is the sort-key.
2. In an AT BOTTOM statement, prints the summary line at the bottom of the control group for which the field-name is the sort-key.

header-element
summary-element

Specify the content, format, and position of the fields printed by the AT TOP and AT BOTTOM statements. These elements form three categories:

1. Elements you can use in both Report Writer AT statements, as well as the Report Writer PRINT statement and the PRINT statement used at the DATATRIEVE command level. These elements are fully described in Table 5–5:
   - fld-name [modifier]
   - val-exp [modifier]
   - COL n
   - SPACE [n]
   - SKIP [n]
   - NEW-PAGE

Do not use SPACE [n] as the first header- or summary-element; use COL n instead.

When you specify NEW-PAGE to start a new report page, and the Report Writer prints the header line and summary lines specified by the AT TOP OF PAGE and AT BOTTOM OF PAGE statements (if either has been included in the report specification). If neither statement is present, the Report Writer prints the report header and column headers specified by other statements in the report specification.

2. Elements you can use only in the Report Writer AT statements:

   - NEW-SECTION

     Starts a new report page and starts a new sequence of page numbers, the new page being number one.

   - REPORT-HEADER

     Prints the entire report header, including report name, date, and page number.

   - COLUMN-HEADER

     Prints the column headers as defined by the PRINT statement and the AT statements.

3. Elements you can use in AT and PRINT statements, but have effects in AT BOTTOM statements that differ from those in the AT TOP and PRINT statements. You have greater control over these statistical functions in AT BOTTOM statements than in AT TOP or PRINT statements:

   - COUNT

     In AT BOTTOM, prints the number of detail lines in the report, report page, or control group specified in the statement.

     In AT TOP and PRINT, prints the number of data records in the current collection, regardless of the number of detail lines in the unit of sorted data specified in the AT TOP statement, or, for the PRINT statement, regardless of the number of detail lines on a report page determined by the PRINT statement.

   - TOTAL [val-exp]

     In AT BOTTOM, prints the total of the value expressions for the unit of sorted data specified in the statement. (See Chapter 6 for detailed information on value expressions.)

     In AT TOP and PRINT, prints the total of the value expressions for the current collection, regardless of the totals of the value expressions in the specified units of sorted data.

- AVERAGE [val-exp]

  In AT BOTTOM, prints the average of the value expressions for the unit of sorted data specified in the statement.

  In AT TOP and PRINT, prints the average of the value expressions for the current collection, regardless of the average of the value expressions in the specified units of sorted data.

- MAX [val-exp]

  In AT BOTTOM, prints the maximum of the value expressions for the unit of sorted data specified in the statement.

  In AT TOP and PRINT, prints the maximum of the value expressions for the current collection, regardless of the maximum of the value expressions in the specified units of sorted data.

- MIN [val-exp]

  In AT BOTTOM, prints the minimum of the value expressions for the unit of sorted data specified in the statement.

  In AT TOP and PRINT, prints the minimum of the value expressions for the current collection, regardless of the minimum of the value expressions in the specified units of sorted data.

If you include a statistical element in a PRINT statement, every detail line in the report will contain the same value in the field specified for that element.

When you include a statistical element in an AT statement, the desired value will be printed in the same column as the values upon which it acts. The format of the statistical value will be the same as that specified by the picture clause or edit-string associated with the value's field. However, the statistical value may not fit the picture or edit-string of that field, or the picture or edit-string explicitly specified for the statistical value may be to long to fit under the appropriate column, and your report may be produced with a jumbled format.

Statistical value for value expressions not included in the PRINT statement are printed in new columns. No special column header is printed unless you specify one. When you specify a column header, make sure there is room for it on the column header line. If the special header does not fit, it is suppressed.

The value determined by COUNT is centered under the next available detail field to the right. You can position the value for COUNT where you want it by assigning its position with a COL n and by giving it an edit-string. Assigning its position with a COL n alone is not sufficient; without the accompanying USING edit-string, the COL n is ignored.

Usage Notes

1. The AT TOP OF REPORT suppresses the report header and the column headers on the first page of the report. You can use this statement to print a special header for the first page of the report or to produce a title page.

   To create a title page, specify the format and content of the page by a series of header-elements, and end the statement with a NEW-SECTION element. This last element shifts the report to a new page and begins numbering the pages of the report from that point on.

   To print the report header and column header line specified by the other Report Writer statements, include REPORT-HEADER and COLUMN-HEADER in the list of header-elements in the AT TOP OF REPORT statement.

   A page header printed by an AT TOP OF PAGE statement replaces the report header and column header on all pages. To print the report header and column header line in addition to the special header produced with the AT TOP OF PAGE statement, include REPORT-HEADER and COLUMN-HEADER in the list of header-elements in the AT TOP OF PAGE statement.

2. The AT TOP OF rse and AT BOTTOM OF rse statements print summary lines at the top or bottom of control groups. A control group consists of a series of sorted data records that all have the same value in in at least one specified field. For a discussion of control groups in reports, see Section 7.3.4.

   The rse you specify in the AT TOP OF rse and AT BOTTOM OF rse statements is the name of the control group, and should be a sort-key.

   When you specify AT TOP OF rse PRINT fld-name, the Report Writer prints the value in the specified field of the first detail line in the control group. When you specify AT BOTTOM OF rse PRINT fld-name, the Report Writer prints the value in the specified field of the last detail line in the control group. The following is an example of a report specification and report using an AT TOP OF rse PRINT fld-name and an AT BOTTOM OF rse PRINT fld-name. Note also that the AVERAGE (DISP) prints the average for the whole collection, not that of the control groups:

```
DTR> READY YACHTS
DTR> FIND YACHTS WITH BUILDER = "PEARSON"
DTR> REPORT FIRST 9 OF CURRENT SORTED BY BEAM, LOA
RW>  ATTOP OF BEAM PRINT SKIP, COL 1, "SHORTEST = ", COL 12,
RW>    LOA USING ZZ9, SPACE, "FT"
RW>  AT BOTTOM OF BEAM PRINT COL 1, "LONGEST =", COL 12,
RW>    LOA USING ZZ9, SPACE, "FT", SKIP,
RW>    "-------------------------------------"

RW> AT BOTTOM OF PAGE PRINT SKIP, "OUR MOTTO: SERVICE AND VALUE"
RW>  SET REPORT-NAME = "BOATS"/"BY"/"PEARSON"
RW>  PRINT COL 25, BEAM, LOA, RIG,
RW>    AVERAGE (DISP) ("AVERAGE"/"WEIGHT"/"OF ALL"/"BOATS")
RW>  SET COLUMNS-PAGE = 55
RW>   END-REPORT
```

```
                                  BOATS                          DD-MMM-YY
                                    BY                           Page 1
                                  PEARSON

                                                                  AVERAGE
                  LENGTH                        LENGTH            WEIGHT
                   OVER                          OVER             OF ALL
                   ALL          BEAM             ALL      RIG      BOATS

SHORTEST =     26  FT
                                  08             26      SLOOP     12,141
LONGEST =      26  FT

          ----------------------------------------

SHORTEST =     26  FT
                                  09             26      SLOOP     12,141
                                  09             28      SLOOP     12,141
                                  09             30      SLOOP     12,141
LONGEST =      30  FT

          ----------------------------------------

SHORTEST =     35  FT
                                  10             35      SLOOP     12,141
LONGEST =      35  FT

          ----------------------------------------

SHORTEST =     33  FT
                                  11             33      SLOOP     12,141
                                  11             36      KETCH     12,141
                                  11             37      SLOOP     12,141
LONGEST =      37  FT

          ----------------------------------------

SHORTEST =     39  FT
                                  12             39      SLOOP     12,141
LONGEST =      39  FT

          ----------------------------------------

                OUR MOTTO: SERVICE AND VALUE
```

3. DATATRIEVE checks the sort order of the collection or record stream you want to report. You can establish a sort order with the REPORT command, a previous SORT command, or a SORTED BY clause in a FIND command. If you have not established a sort order for the collection or record stream by any of the methods, the Report Writer prints the following message after encountering the END-REPORT statement:

```
Proceeding to report unsorted records
```

   The Report Writer prints this message on your terminal even if you have sorted the records in the current collection or record stream prior to invoking DATATRIEVE. However, if you have sorted your records prior to entering DATATRIEVE, the Report Writer still makes the breaks between control groups just as though you had done the sorting during your DATATRIEVE session.

If the data is in fact unsorted, the Report Writer divides the detail lines into control groups anyway. A new control group is formed every time the value in the specified field changes. If no values in that field are repeated in consecutive detail lines the Report Writer treats each line as a separate control group and prints the header and summary lines above and below each line as indicated in the record specification.

7.4.5 END-REPORT Statement

Function

The END-REPORT statement signals an end to the report specification.

Format

```
END-REPORT
```

Usage Notes

1. The END-REPORT statement must be the last statement in the report specification.
2. Following the END-REPORT statement, the Report Writer can take three courses of action:
   - Prompt you for the values you specified with a *.prompt-name in the record specification.
   - Send you a message indicating a syntax error in the report specification. If the Report Writer detects any syntax errors in the specification, the report is not produced, and the report specification is not saved. If you entered the report specification interactively, you have to re-enter it completely. If the report specification is in a procedure, use the DATATRIEVE Editor to correct the errors.
   - Produce the report and send it to the device or file you have specified in the REPORT command.

Chapter 8
Using the DATATRIEVE Editor

You can use the DATATRIEVE editor, a subset of the DEC Standard Editor, to modify dictionary objects already defined in your current data dictionary. To edit a dictionary object, you must have C (control) access to it.

As you add to or change the dictionary objects, the DATATRIEVE editor does not check for syntax errors in the definitions. Any such errors become apparent only when you use the modified dictionary object. Hence, you should edit with caution, especially when changing record definitions.

The changes you make with the DATATRIEVE editor apply only to the dictionary object in the data dictionary. Any dictionary object residing in your DATATRIEVE workspace is unchanged by the edits you make to the dictionary entry. For example, the current record definition of a readied domain is unchanged when you edit the dictionary entry of the record definition. The new record definition will not take effect until the associated domain is finished and readied again. Similarly, a description table loaded into the DATATRIEVE workspace is not affected by edits to the dictionary entry for the table; you must release the table and refer to it again before the changes take effect.

8.1 Invoking the Editor

You invoke the DATATRIEVE editor at the DATATRIEVE command level with the following command:

EDIT dictionary-object-name [(passwd) / (*)] [ADVANCED]

Arguments

dictionary-object-name
Is the name of the dictionary object (in the current data dictionary) you want to edit.

(passwd)
(*)
Is an asterisk enclosed in parentheses (*) or the password necessary to gain C (control) access to the dictionary object. If you specify a password, you must enclose it in parentheses. If you specify (*), DATATRIEVE prompts you for the password but does not print the response on your terminal. If you omit this argument, DATATRIEVE uses your log-in UIC/PPN to verify that you have C (control) access privilege to the dictionary object you want to edit.

ADVANCED
Must be included in the EDIT command if you want to edit a domain definition or a record definition.

When you invoke the DATATRIEVE editor, it responds with the following prompt:

```
QED>
```

8.2 Editor Modes

When you first invoke the editor, you are at the editor's command level and are in "edit mode." In this mode, the editor interprets all your input as commands, and you can display and alter the text of the dictionary object. The QED> prompt indicates that you are in edit mode.

The second editor mode, "insert mode," allows you to enter text directly into the dictionary object. You enter the insert mode with the INSERT and REPLACE commands and leave it with CTRL/Z. The editor uses the IN> prompt to indicate that you are in insert mode. In this mode, the editor interprets all your input as new text to be entered into the dictionary object, not as editor commands.

8.3 Line Pointer

The editor uses a line pointer to keep track of the current line. Some 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 line pointer keeps track of your position in the text.

The line pointer points to the entire current line, not to any part of the line. You can display the current line by typing a period (.) and a carriage return in response to the QED> prompt. The line pointer can also point to the end of the text buffer, where you can add text to the end of the dictionary object. The symbol [EOB] marks the end of the text buffer.

8.4 Range Specification

The editor commands for deleting, inserting, replacing, and typing lines, and for substituting strings all contain an optional argument that specifies the range of lines on which the command operates. The range may be a single line, a series of consecutive lines, or a group of nonsequential lines. The range specifiers you can use with the DATATRIEVE editor are summarized in Table 8–1.

Table 8–1: Range Specifiers

Range Specifier	Format	Command Applies To:
[%]ALL	[%]ALL "string"	All lines containing the specified string. The search does not distinguish between upper- and lowercase letters.
[%]AND (or ,)	r-1 { [%]AND / , } r-2[,r-3]... where ranges r-1, r-2 are: { . / BEGIN / END / "string" }	The line indicated by range r-1, then line indicated by range r-2 and so forth.
[%]BEFORE	[%]BEF[ORE]	All lines before the current line and the current line.
[%]BEGIN	[%]BE[GIN]	The first line of the dictionary object.
[%]END	[%]E[ND]	The last line of the dictionary object.
[%]FOR (or ;)	r { [%]FOR / ; } n where range r is: { . / BE[GIN] / "string" }	The number of lines specified by n, starting at the line specified by range r.
[%]REST	[%]R[EST]	The current line and all remaining lines in the dictionary object.
"string"	{ 'string' / "string" }	The current line or the next line containing the specified string. The search does not distinguish between upper- and lowercase letters, and it starts with the current line and continues toward the end of the dictionary object until it finds the specified string. You can use either single or double quotation marks to enclose strings. However, you must use them in pairs; you cannot mix them.

(continued on next page)

Table 8-1: Range Specifiers (Cont.)

Range Specifier	Format	Command Applies To:
[%]WHOLE	[%]WH[OLE]	All lines of the dictionary object.
+	r + n where range r is: { . / BE[GIN] / "string" }	The line that is n lines from the line specified by range r. When "string" is the range specifier, the command applies to the line that is n lines from the next occurrence of the specified string.
.(period)		The current line.

Table 8–2 shows several examples of these ranges. The sample command is TYPE, which serves two purposes: it prints specified lines on your terminal, and it searches for specified strings. The format of the TYPE command shown in these examples is only one of several possible formats. See Section 9.5.7 for a detailed discussion of the TYPE command.

Do not use the THRU or : range specifier when working in the DATATRIEVE editor. If you do, your edits are cancelled and you are returned to the DATATRIEVE command level.

Note that if you want to print the rest of a dictionary object using the %R indicator without a TYPE command, you must specify the percent sign before the R to distinguish it from the abbreviated form of the REPLACE command. For example, the following commands all type the rest of the dictionary object:

```
TYPE REST

TYPE R

REST

T R

%R
```

Entering R by itself in response to the QED> does not type the rest of the dictionary object; it deletes the current line and puts you in the insert mode.

Table 8–2: Examples of Range Specifiers

You Type:	Editor Prints:
TYPE ALL "BEAM"	All lines containing the string "BEAM".
TYPE BEGIN AND END	The first line of the dictionary object and the end-of-text-buffer marker ([EOB]).
TYPE BEGIN, "BEAM"	The first line of the dictionary object and the first line containing the string "BEAM". If the current line is the first line of the dictionary object and contains the string "BEAM", this command prints the current line twice.
TYPE BEFORE	All lines before the current line and the current line.
TYPE BEGIN	The first line of the dictionary object.
TYPE END	The end-of-text-buffer marker ([EOB]).
TYPE . FOR 5	The current line and the four lines following it.
TYPE REST	The current line and all remaining lines in the dictionary object.
TYPE 'BEAM'	The next line containing the string "BEAM". If the current line contains "BEAM", the editor prints the current line.
TYPE WH	All lines in the dictionary object.
TYPE BE+5	The sixth line of the dictionary object.
TYPE "BEAM"+6	The sixth line following the next line containing the string "BEAM". If the current line contains the string "BEAM", the editor prints the sixth line following the current line.
TYPE .	The current line.

8.5 Editor Commands

Table 8–3 contains a summary of the DATATRIEVE editor commands. The commands are described in alphabetical order in this section. Section 8.6 contains a sample editing session illustrating some common uses of these commands.

Table 8-3: Summary of DATATRIEVE Editor Commands

Command	Format	Function
CTRL/Z	CTRL/Z	In edit mode, same as EXIT command. In insert mode, returns control to edit mode.
DELETE	D[ELETE] [range]	Deletes the specified range or the current line if you omit the range. After deletion, the current line is the line following the last line deleted.
EXIT	EX[IT]	Returns you to DATATRIEVE command level. The edited dictionary object replaces previous version in data dictionary.
INSERT	I[NSERT] [range]	Enters the insert mode. The editor inserts lines before the line specified by the range or before the current line if you omit the range. CTRL/Z ends the insert mode and returns you to the editor command level. If you omit the range, the current line does not change when you leave insert mode; if you specify a range, the line specified by the range becomes the current line. Do not use ALL, AND (,), BEFORE, FOR (;), REST, OR WHOLE when specifying the range in this command.
QUIT	QUIT	Returns you to DATATRIEVE command level and leaves the dictionary object unchanged by the editing session.
REPLACE	R[EPLACE] [range]	Deletes the specified range or the current line if you omit the range, and puts you in insert mode. CTRL/Z ends insert mode and returns you to the editor command level, and the current line is the line following the last line deleted. None of the restrictions on specifying ranges in the INSERT command apply to REPLACE.
SUBSTITUTE	S/str-1/[str-2][/[range]]	Substitutes string-2 for all occurrences of string-1 in the specified range or in current line if you omit the range. The line in which the last substitution occurred becomes the current line. If you omit the range, you can also omit the third delimiter. If you omit string-2, the editor deletes the specified string from the range you specify or from the current line if you omit the range. The search for the first occurrence of string-1 starts with the current line and proceeds toward the end of the text buffer. Do not use END to specify the range in this command.
TYPE	[T[YPE]] [range]	Displays the specified range of lines or the line following the current line if you omit the range. The first line displayed becomes the current line, with one exception: if E[ND] plays any part in the range of this command, the line pointer points at the end-of-text-buffer marker ([EOB]).

8.5.1 DELETE Command

Function

Deletes one or more lines from the dictionary object.

Format

D[ELETE][range]

Argument

range
Specifies the range of lines to be deleted. If you omit the range, only the current line is deleted.

Position of Line Pointer

Current line is the line following the last line deleted.

Examples

1. Delete the current line only:

```
QED> D
```

2. Delete all lines containing the string 'PHD':

```
QED> D ALL 'PHD'
```

3. Delete all lines from the next line containing "PHD" to the end of the dictionary object:

```
QED> D "PHD" FOR n
```

where n is greater than or equal to the number of lines from the current line to the end of the text buffer.

4. Delete all lines from the beginning of the dictionary object up to, and including, the current line:

```
QED> D BEFORE
```

8.5.2 EXIT Command

Function

Ends an editing session, and returns you to the DATATRIEVE command level. The edited dictionary object replaces the previous version in the data dictionary.

Format

EX[IT]
CTRL/Z

Argument

None.

Position of the Line Pointer

Not applicable.

Examples

1. End the current editing session with EXIT command:

```
QED> EX
DTR>
```

2. End the current editing session with CTRL/Z:

```
QED> CTRL/Z
DTR>
```

8.5.3 INSERT Command

Function

Enters the INSERT mode, which allows you to enter text directly into the dictionary object. The inserted lines are added before the line specified in the range or before the current line if you omit the range.

Format

I[NSERT] [range]

Argument

range

Is the line before which the inserted lines are added. If the range is omitted, the inserted lines are added before the current line.

Position of the Line Pointer

If you omit the range, the current line does not change when you leave insert mode; if you specify a range, the line specified by the range becomes the current line.

Notes

1. When you issue the INSERT command, the editor prompts with IN> to show that you are in insert mode.

2. Do not use the following range specifiers in the INSERT command:
   - [%]ALL
   - [%]AND (,)
   - [%]BEF[ORE]
   - [%]FOR (;)
   - [%]R[EST]
   - [%]WH[OLE]

3. To leave the insert mode and return to edit mode, enter CTRL/Z.

Examples

1. Insert lines before the current line:

```
QED> I
IN>      [text]
           .
           .
           .
IN>      [text]
IN>      CTRL/Z
QED>
```

2. Insert lines before the first line of the dictionary object:

```
QED> I BEGIN
IN>      [text]
```

3. Insert lines after the last line of the dictionary object:

```
QED> I END
[EOB]
IN>      [text]
```

4. Insert lines before the next line containing the string "BEAM":

```
QED> I 'BEAM'
IN>      [text]
```

5. Insert lines between the fourth and fifth lines from the current line:

```
QED> I .+5
IN>      [text]
```

8.5.4 QUIT Command

Function

Returns you to the DATATRIEVE command level, and leaves the dictionary object unchanged by the editing session.

Format

QUIT

Argument

None.

Position of the Line Pointer

Not applicable.

Notes

1. QUIT aborts an editing session. The dictionary object is left unaffected by any editing changes made during the editing session.
2. The QUIT command cannot be abbreviated.

Example

Abort the current editing session:

```
QED> QUIT
DTR>
```

8.5.5 REPLACE Command

Function

Deletes the specified range of lines in a dictionary object, or deletes the current line if you omit the range, and enters the insert mode, which allows you to enter text directly into the dictionary object.

Format

R[EPLACE] [range]

Argument

Specifies the range of lines to be deleted. If you omit the range, only the current line is deleted.

Position of the Line Pointer

After you leave the insert mode, the current line is the line following the last line deleted; the current line is also the line after the inserted lines.

Notes

1. When you issue the REPLACE command, the editor deletes the lines specified by the range and then prompts with IN> to show that you are in insert mode.
2. To leave the insert mode and return to edit mode, enter CTRL/Z.
3. None of the restrictions on specifying ranges in the INSERT command apply to REPLACE.

Examples

1. Replace the current line with a single line:

```
QED>      FIND YACHTS WITH LOA BETWEEN 36 AND 37
QED> R
IN>       FIND YACHTS WITH LOA > 38
IN>       CTRL/Z
QED> .
          REPORT CURRENT SORTED BY BEAM, LOA, RIG ON *.WHERE
```

2. Delete the first line of a dictionary object and enter insert mode:

```
QED> R BE
IN>      [text]
```

3. Delete all lines containing the string "PRICE" and enter the insert mode at the line following the last line deleted:

```
QED> R ALL "PRICE"
IN>      [text]
```

4. Delete all lines in the dictionary object and enter the insert mode:

```
QED> R WHOLE
IN>      [text]
```

8.5.6 SUBSTITUTE Command

Function

Substitutes a character string for all occurrences of another character string in the specified range or in the current line if you omit the range.

Format

S[UBSTITUTE] /string-1/[string-2][/[range]]

Arguments

string-1
Is the string of characters to be replaced.

/

Is the delimiter that separates the string–1, string–2, and the range. The delimiter can be any printing character not in string–1 or string–2. Only the first two delimiters are required if you omit the range. If you specify a range, you must use all three delimiters, and you must use the same delimiter all three times.

string–2

Is the string of characters to replace string–1. If you omit string–2, then string–1 is deleted from the current line or from the specified range.

range

Specifies the range of lines within which the substitution is to be made. If you omit the range, only the current line is affected. If you specify a range, all occurrences of string–1 within that range are replaced by string–2. If you omit the range, only the first occurrence of string–1 is replaced by string–2, and that first occurrence of string–1 need not be in the current line.

Position of the Line Pointer

If a substitution takes place, the current line is the line in which the last substitution occurred. If there is no match for string–1 within the range, if specified, or if there is no match for it in the current line or any line from that point to the end of the dictionary object, and no substitution takes place, the current line is unchanged.

Notes

1. The editor prints each line in which a substitution occurs.
2. Do not use END to specify the range in this command.
3. The search for the first occurrence of string–1 starts with the current line and proceeds toward the end of the text buffer.

Examples

1. Substitute the string YACHT for the first occurrence of the string YAHCT in the current line only:

```
QED> S /YAHCT/YACHT/ .
```

2. Substitute the string YACHT for every occurrence of YAHCT in the entire dictionary object:

```
QED>S *YAHCT*YACHT* WH
```

This WHOLE range is effective regardless of the position of the current line in the text buffer. If any substitutions are made, the current line is the line in which the last substitution was made. The search for string–1 begins at the first line of the dictionary object and proceeds toward the end of the text buffer.

3. Substitute the string YACHT for the next occurrence of YAHCT:

```
QED> S /YAHCT/YACHT
```

   If there is no occurrence of string–1 between the current line and the end of the text buffer, then the current line is unchanged.

4. Substitute the string YACHT for every occurrence of YAHCT from the current line to the end of the dictionary object:

```
QED> S "YAHCT"YACHT" REST
```

8.5.7 TYPE Command

Function

Displays the specified range of lines or the line following the current line if you omit the range.

Format

[T[YPE]] [range]

Argument

range
   Specifies the range of lines to be displayed. If you omit the range, only the line following the current line is displayed.

Position of the Line Pointer

The first line displayed becomes the current line, with one exception: if E[ND] plays any part in the range of this command, the line pointer points at the end-of-text-buffer marker ([EOB]).

Notes

1. Both the command name and range are optional. If you enter only a carriage return, the editor prints the line following the current line, which then becomes the current line. You can also enter the range specifiers without the command name.
2. The TYPE command performs the searches in the DATATRIEVE editor. You can search for text strings in dictionary objects by enclosing the string you seek in pairs of single or double quotation marks.

Examples

1. Print the current line only:

```
QED> TYPE .
or
QED> .
```

2. Print the next line (the line following the current line):

```
QED> TYPE .+1
or
QED> .+1
or
QED> T
or
QED> RET
```

3. Print the entire dictionary object:

```
QED> T WH
or
QED> WH
```

4. Print the first and last lines of the dictionary object:

```
QED> T BEGIN,END
or
QED> BEGIN,END
or
QED> BE AND E
```

5. Print all lines from the current line to the end of the dictionary object:

```
QED> T REST
or
QED> REST
or
QED> %R
```

8.6 Sample Editing Session

```
DTR> EDIT CTRL
QED> WH
        READY YACHTS
        FIND YACHTS WITH LOA BETWEEN 36 AND 37
        REPORT CURRENT SORTED BY BEAM, LOA, RIG ON *,WHERE
        SET  REPORT-NAME = "YACHTS WITH LENGTH-OVER-ALL"/
           "OF 36 AND 37 FEET"
        AT TOP OF BEAM PRINT COL 1, "BEAM = ", COL 7, BEAM
        AT TOP OF LOA PRINT COL 12, "LENGTH = ", COL 20, LOA, SKIP
        PRINT BUILDER, RIG, DISP
        AT BOTTOM OF RIG PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
           LOA(" "), COL 17, "FOOT ", COL 22, RIG, COL 28,
           "WITH BEAM OF ", COL 40, BEAM, COL 43, " = ", COL 46,
           COUNT USING Z9, SKIP
        AT BOTTOM OF REPORT PRINT SKIP, "LIGHTEST = ", MIN (DISP),
           SKIP, "HEAVIEST = ", MAX (DISP), SKIP,
           "AVERAGE WEIGHT OF ALL BOATS = ", AVERAGE (DISP)
        SET DATE = "DD-MMM-YY"
        SET COLUMNS-PAGE = 60
        END-REPORT
QED> .
        READY YACHTS
QED> RET
        FIND YACHTS WITH LOA BETWEEN 36 AND 37
```

```
QED> R
IN>     FIND YACHTS WITH LOA > 38
IN>     CTRL/Z
QED> .
        REPORT CURRENT SORTED BY BEAM, LOA, RIG ON *.WHERE
QED> S/ON *.WHERE/
        REPORT CURRENT SORTED BY BEAM, LOA, RIG
QED> RET
        SET REPORT-NAME = "YACHTS WITH LENGTH-OVER-ALL"/
QED> RET
        "OF 36 AND 37 FEET"
QED> R
IN>     "GREATER THAN 38 FEET"
IN>     CTRL/Z
QED> .
        AT TOP OF BEAM PRINT COL 1, "BEAM = ", COL 7, BEAM
QED> ALL "AT BOTTOM"
        AT BOTTOM OF RIG PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
        AT BOTTOM OF REPORT PRINT SKIP, "LIGHTEST = ", MIN (DISP),
QED> .
        AT BOTTOM OF RIG PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
QED> "BOTTOM"
        AT BOTTOM OF RIG PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
QED> RET
        LOA(" "), COL 17, "FOOT ", COL 22, RIG, COL 28,
QED> "BOTTOM"
        AT BOTTOM OF REPORT PRINT SKIP, "LIGHTEST = ", MIN (DISP),
QED> .;3
        AT BOTTOM OF REPORT PRINT SKIP, "LIGHTEST = ", MIN (DISP),
           SKIP, "HEAVIEST = ", MAX (DISP), SKIP,
           "AVERAGE WEIGHT OF ALL BOATS = ", AVERAGE (DISP)
QED> .
        AT BOTTOM OF REPORT PRINT SKIP, "LIGHTEST = ", MIN (DISP),
QED> D .;3
QED> .
        SET DATE = "DD-MMM-YY"
QED> I
IN>     SET MAX-PAGES = 40
IN>     CTRL/Z
QED> .
        SET DATE = "DD-MMM-YY"
QED> RET
        SET COLUMNS-PAGE = 60
QED> RET
        END-REPORT
QED> RET
        [EOB]
QED> BE
        READY YACHTS
QED> S/RIG/PRICE/WH
        REPORT CURRENT SORTED BY BEAM, LOA, PRICE
        PRINT BUILDER, PRICE, DISP
        AT BOTTOM OF PRICE PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
           LOA(" "), COL 17, "FOOT ", COL 22, PRICE, COL 28,
```

```
QED> WH
        READY YACHTS
        FIND YACHTS WITH LOA > 38
        REPORT CURRENT SORTED BY BEAM, LOA, PRICE
        SET REPORT-NAME = "YACHTS WITH LENGTH-OVER-ALL"/
        "GREATER THAN 38 FEET"
        AT TOP OF BEAM PRINT COL 1, "BEAM = ", COL 7, BEAM
        AT TOP OF LOA PRINT COL 12, "LENGTH = ", COL 20, LOA, SKIP
        PRINT BUILDER, PRICE, DISP
        AT BOTTOM OF PRICE PRINT SKIP, COL 4, "NUMBER OF ", COL 14,
           LOA(" "), COL 17, "FOOT ", COL 22, PRICE, COL 28,
           "WITH BEAM OF ", COL 40, BEAM, COL 43, " = ", COL 46,
           COUNT USING Z9, SKIP
        SET MAX-PAGES = 40
        SET DATE = "DD-MMM-YY"
        SET COLUMNS-PAGE = 60
        END-REPORT
QED> EX
DTR>
```

Chapter 9
Application Design Tool

The Application Design Tool (ADT) is an interactive feature of DATATRIEVE that aids in the creation of a domain definition, along with the associated record definition and data file. Using your responses to its questions, ADT creates an indirect command file containing the data dictionary definitions of the domain and record, and the file specification of the data file. To enter the definitions in the data dictionary, you merely execute the indirect command file. (See Chapter 12: Procedures and Indirect Command Files.)

NOTE

Because the installation of the Application Design Tool is optional, ADT may not be available at your installation.

ADT provides an easy-to-learn, easy-to-use method of creating a domain and specifying a simple record definition. The record definition contains many, but not all of the features and clauses available in a full record definition. ADT allows you to define the domain, record, and data file easily by eliminating the need to issue the corresponding DATATRIEVE DEFINE commands. Thus, you can create dictionary definitions with little or no prior experience with DATATRIEVE.

To add field definitions or field definition clauses to the record definition produced by ADT, you can edit the indirect command file with a text editor.

The *DATATRIEVE Primer* contains a brief description of ADT. This chapter supplements that information and includes a more detailed description of ADT's features, your dialog with ADT, and the output ADT produces.

9.1 ADT Features

Using ADT you can create a domain definition for an RMS domain and the associated record definition. You create the record definition by defining every field in the record. You also specify the name of the file to contain the data for the domain. The data file can be sequential or indexed sequential. With ADT, however, you cannot create a domain as a view; to create a domain of that type, you must use the DEFINE DOMAIN command.

During an ADT session, you specify the following information:

- The name of the domain.
- The name of the data file for the domain.
- All elementary fields in the record. You cannot include group fields in the record definition. ADT creates one group field with a level number of 01 and assigns the record name as the group field name.
- The format of every data field in the record. A data field can be formatted as a date, a percentage, money, numbers, or characters.
- A query name, if desired, for any field with a name longer than seven characters. You cannot include a query header for any field or a query name for any field with a name of seven or fewer characters.
- The format of the data file: either sequential or indexed sequential. For an indexed sequential file, you indicate the name of the primary key field and up to two alternate key fields. You can also indicate if a key field can have a duplicate value and if an alternate key field can be modified.

As you respond to ADT's questions, ADT checks your responses. For example, ADT verifies that any name you specify (such as a field name) conforms to DATATRIEVE's rules for names. In addition ADT provides on-line assistance, if you type a question mark (?). You can also direct ADT to print detailed or abbreviated questions.

9.2 Invoking and Terminating ADT

You invoke ADT at the DATATRIEVE command level with the ADT command:

```
DTR> ADT
```

ADT then begins its dialog, which consists of a series of questions.

To terminate ADT when you have finished defining the domain, record, and data file, type NO (or N) in response to the question:

```
Do you want to define another domain? (YES or NO):
```

You can also terminate ADT by responding with a CTRL/Z to any ADT question. When you terminate ADT with a CTRL/Z, you return to the DATATRIEVE command level.

9.3 ADT Dialog

ADT's first question is:

```
Do you want help? (YES or NO):
```

If you respond YES (or Y), ADT prints a short explanation of its functions and the on-line assistance available to you.

9.3.1 Responding to ADT Questions

Table 9–1 contains a list of the types of questions ADT asks and the acceptable responses to each type. As shown in the table, in response to any question, you can enter a question mark (?) to receive an additional explanation of the question, an exclamation point (!) to print the fields defined thus far, or CTRL/Z to terminate the dialog.

If you enter an inappropriate response to a question, ADT prints an error message and repeats the question. ADT also prints an error message and repeats the question if one of the following errors occurs:

1. Violation of rules for names. Section 4.4 contains a list of rules for specifying DATATRIEVE names.
2. Duplicate names. A field name cannot duplicate either the domain name or another field name in the domain. Field and domain names must be unique.
3. Duplicate key fields. A key field must be unique; that is, the primary key field must be different from the alternate key field(s).

Table 9–1: ADT Question Types and Responses

Question Type	Response	Additional Explanation
any question	?	ADT prints additional information about the question.
	!	ADT prints the fields that have already been defined.
	CTRL/Z (^Z)	ADT terminates, returns you to the DATATRIEVE command level, and may not have created an indirect command file.
... ? (YES or NO):	YES (or Y) NO (or N)	A single-character response is sufficient for ADT questions of this type.

(continued on next page)

Table 9–1: ADT Question Types and Responses (Cont.)

Question Type	Response	Additional Explanation
What's in fld-name– . . . Enter one of the above:	DATE (or D)	ADT then asks for the format of the date.
	PERCENT (or P)	ADT automatically defines the field as 3 numeric positions with a decimal point implied to the right of the right most digit. The field value is printed with a percent sign (%).
	MONEY (or M)	ADT then asks for the number of digits to the left of the decimal point. (ADT assumes 2 digits to the right of the decimal point.) The field value is printed with a dollar sign ($) preceding the leftmost digit.
	NUMBERS (or N)	ADT then asks for the number of digits to the left and the number of digits to the right of the decimal point. ADT also asks if you want the leading zeros to be suppressed when the field value is printed.
	CHARACTERS (or C)	ADT then asks for the maximum number of characters the field can contain.
... query abbreviation for fld-name?	RET	A carriage return indicates that the field should not have a query name.
	query-name	ADT uses the character string as the query name for the field.

9.3.2 Sample ADT Dialog

Figure 9–1 shows an annotated ADT dialog with detailed questions. The sample dialog illustrates several different types of responses to ADT questions (including a typing error).

Figure 9–1: A Sample ADT Dialog

```
DTR> ADT
Do you want help? (YES or NO) : N
Do you want detailed questions? (YES or NO) : Y
What do you want to name this domain? : PAYROLL
What do you want to name the file where the data
for PAYROLL will be? : PAY.DAT
What do you want to name the first field in PAYROLL? : EMPLOYEE-NAME
What is the query abbreviation for EMPLOYEE-NAME? : NAME
```

(continued on next page)

```
What's in EMPLOYEE-NAME--
        a DATE
        a PERCENT
        MONEY
        NUMBERS used in arithmetic
        or ANYTHING ELSE (CHARACTERS)?

Enter one of the above : CC
Your answer must be DATE, PERCENT, MONEY, NUMBERS
or CHARACTERS.
What's in EMPLOYEE-NAME--
        a DATE
        a PERCENT
        MONEY
        NUMBERS used in arithmetic
        or ANYTHING ELSE (CHARACTERS)?

Enter one of the above : C
How many characters long is EMPLOYEE-NAME ? : 25
Are there any more fields in PAYROLL ? (YES or NO) : Y
What do you want to name the next field in PAYROLL? : SALARY
What's in SALARY--
        a DATE
        a PERCENT
        MONEY
        NUMBERS used in arithmetic
        or ANYTHING ELSE (CHARACTERS)?

Enter one of the above : M
How many digits to the left of the decimal point? : 6
Are there any more fields in PAYROLL ? (YES or NO) : YES
What do you want to name the next field in PAYROLL? : REVIEW-DATE
What is the query abbreviation for REVIEW-DATE? : REVIEW
What's in REVIEW-DATE--
        a DATE
        a PERCENT
        MONEY
        NUMBERS used in arithmetic
        or ANYTHING ELSE (CHARACTERS)?

Enter one of the above : D
Four date formats are available:
        1       MM/DD/YY            6/29/79
        2       DD-MMM-YY          29-JUN-79
        3       DD-MMM-YYYY        29-JUN-1979
        4       DD.MM.YY           29.06.79
Enter format number 1, 2, 3, or 4 : 1
Are there any more fields in PAYROLL ? (YES or NO) : !

 PAYROLL
   EMPLOYEE-NAME (NAME) 25 characters
   SALARY 6 digits left (and 2 digits right) of the decimal, is money
   REVIEW-DATE (REVIEW) is a date

Are there any more fields in PAYROLL ? (YES or NO) : Y
What do you want to name the next field in PAYROLL? : REVIEWER
What is the query abbreviation for REVIEWER? :
What's in REVIEWER--
        a DATE
        a PERCENT
        MONEY
        NUMBERS used in arithmetic
        or ANYTHING ELSE (CHARACTERS)?
```

(continued on next page)

```
Enter one of the above : C
How many characters long is REVIEWER ? : 25
Are there any more fields in PAYROLL ? (YES or NO) : N
An indexed file can handle certain queries based on a key
field very quickly. A sequential file is not as fast and
does not allow records to be ERASEd. BUT, an indexed file
does not allow you to change the primary key field's data.

Do you want your data file to be indexed? (YES or NO) : N
What is the name of the file where the DATATRIEVE
domain and field definitions should go? : PAY.CMD
The DATATRIEVE definitions for your domain are
located in file SY:[311,350]PAY.CMD;1
The record length is 67 bytes.
Do you want to define another domain? (YES or NO) : NO
DTR>
```

9.4 Output from ADT

ADT produces a command file containing the definitions you specified in the dialog.

The indirect command file contains three DATATRIEVE commands:

- DEFINE DOMAIN
- DEFINE RECORD
- DEFINE FILE

If necessary, before you execute the file, you can edit it using a text editor. You cannot edit the file while in DATATRIEVE.

Figure 9–2 contains the general format of the command file contents.

Figure 9–2: General Format of Command File Contents

```
DEFINE DOMAIN domain-name USING domain-name-REC
     ON file-spec;
DEFINE RECORD domain-name-REC USING
01  domain-name-REC.
    15  fld-name-1
            .
            .
            .
   [15 fld-name-n]
;
DEFINE FILE FOR domain-name [KEY=fld-name (opt),...];
```

Figure 9–3 shows the actual command file produced by the sample dialog in Figure 9–1.

Figure 9-3: Sample Command File Contents

```
DEFINE DOMAIN PAYROLL USING PAYROLL-REC
     ON PAY.DAT;
DEFINE RECORD PAYROLL-REC USING
01 PAYROLL-REC.
   15 EMPLOYEE-NAME      PIC IS X(25)
      QUERY-NAME IS NAME.
   15 SALARY     PIC IS S9(6)V99
      EDIT-STRING IS $$$$,$$$.$$.
   15 REVIEW-DATE        USAGE IS DATE
      EDIT-STRING IS NN/DD/YY
      QUERY-NAME IS REVIEW.
   15 REVIEWER           PIC IS X(25).
;
```

ADT inserts the appropriate field definition clauses in the record definition. These clauses can be:

- EDIT-STRING for a date, percentage, numeric, or money field.
- USAGE for a date field.
- PIC for a percentage, money, numeric, or character field.
- QUERY-NAME for any field with a name longer than seven characters, if you specified a query name in the dialog.

ADT also adds a group field with level number 01 and assigns a level number of 15 to each elementary field definition.

9.5 Clauses and Other Record Attributes Not Available in ADT

Some of the limitations of ADT have already been mentioned: you cannot define group fields or query headers, you can have no USAGE clauses other than that for dates, and you can define query names only for fields with names greater than seven characters.

In addition, you have no control over edit-strings beyond that which ADT offers you in the dialog, and you cannot insert filler between fields.

The record definitions ADT creates do not contain any of the following clauses:

- COMPUTED BY
- OCCURS
- REDEFINES
- SIGN
- VALID IF

If you want to insert any of these clauses in the record definition that ADT creates, you must exit from DATATRIEVE and use a text editor to change the indirect command file.

9.6 Executing the Indirect Command File

You execute the file produced by ADT as an indirect command file at the DATATRIEVE command level (see Chapter 12: Procedures and Indirect Command Files).

```
DTR> @file-spec
```

DATATRIEVE then executes each DEFINE command in the file and prints the command on your terminal as it is executed. Figure 9–4 shows the execution of the command file created earlier in this chapter. The definitions are stored in the current data dictionary. To enter data in the records, you can ready the domain for WRITE or EXTEND access (see Section 5.26) and use the STORE statement (Section 5.36).

Figure 9–4: Execution of Sample Indirect Command File

```
DTR> @PAY.CMD
DEFINE DOMAIN PAYROLL USING PAYROLL-REC
     ON PAY.DAT;
DEFINE RECORD PAYROLL-REC USING
01 PAYROLL-REC.
   15 EMPLOYEE-NAME      PIC IS X(25)
      QUERY-NAME IS NAME.
   15 SALARY     PIC IS $9(6)V99
      EDIT-STRING IS $$$$,$$$.$$.
   15 REVIEW-DATE        USAGE IS DATE
      EDIT-STRING IS NN/DD/YY
      QUERY-NAME IS REVIEW.
   15 REVIEWER           PIC IS X(25).
;
[Record PAYROLL-REC is 67 bytes long]
DEFINE FILE FOR PAYROLL;
DTR>
```

Chapter 10
Creating and Maintaining Data Dictionaries

Each time you invoke DATATRIEVE, you are automatically connected to a data dictionary. A data dictionary is an RMS file created by DATATRIEVE for the storage of definitions and protection information. The file is created on disk.

This chapter summarizes the steps involved in creating a data dictionary. See Chapter 5 for a description of the DEFINE DICTIONARY command which you use in this process. The present chapter also describes how to maintain a data dictionary and compress it, if necessary, to eliminate unused disk areas. First, however, it will be helpful for you to understand the contents of a data dictionary.

10.1 Contents of a Data Dictionary

A data dictionary contains the definitions and protection information for the following DATATRIEVE data structures:

- Domains
- Records
- Procedures
- Description tables

Each definition describes the contents of the data structure:

- A domain definition contains the name of the domain, the name of the record definition associated with the domain, and the file specification of the file containing the data for the domain.
- A record definition contains the name of the record and a definition for each field in the record. Record and field definitions are described in Chapter 11.
- A procedure definition is the procedure itself, including the procedure name and all commands and statements in the procedure. Chapter 12 describes procedures and procedure definitions.

- A description table definition is the description table itself, including its name and all code-and-description pairs. Description tables and their definitions are described in Chapter 13.

Associated with each of these definitions is a password table. It protects the one definition with which it is associated and restricts its use. Password tables supplement the protection features of your operating system. They are described in Chapter 14.

10.2 Creating a Data Dictionary

You create a data dictionary using the DEFINE DICTIONARY command. The format of this command is shown below; you can find more information on its use in Section 5.6.

```
DEFINE DICTIONARY  file-spec
```

DEFINE DICTIONARY creates an empty indexed sequential RMS file that is suitable for use as a data dictionary. You supply the file specification, and the operating system creates an entry for it in your directory. You can choose any extension for the file when you create it; but, if you use the default extension (.DIC), the dictionary file will be easier for you to identify when you list your directory.

As soon as you enter the DEFINE DICTIONARY command, DATATRIEVE creates a file, if it can, and establishes the new dictionary as your current data dictionary, just as if you had done a SET DICTIONARY command yourself. You can then begin entering definitions immediately. If DATATRIEVE cannot create a file because, for example, you do not have write access to the disk, or the disk is not mounted, it prints a message on your terminal and leaves you connected to your current dictionary.

10.3 Changing Dictionaries

When you invoke DATATRIEVE, you are automatically connected to a default data dictionary. At the beginning of your DATATRIEVE session, that dictionary is your current dictionary. You can be connected to only one dictionary at any given time during a session.

To find out the name of your current dictionary, use the following command:

```
DTR> SHOW DICTIONARY
```

SHOW DICTIONARY prints the file specification of the current data dictionary.

If you want to use a different dictionary, use the following format of the SET command:

```
DTR> SET DICTIONARY file-spec
```

To change from the default dictionary to another dictionary, you must specify at least one element of the file specification of the other data dictionary. For example, if you specify only the device name, DATATRIEVE searches that device for a directory with your UIC/PPN and a file named QUERY.DIC. (See Section 5.31 for the defaults of the SET DICTIONARY command.) If DATATRIEVE does not find the file you specify, it prints an error message on your terminal, and the dictionary current before you issued the SET DICTIONARY command is still current. At no time are you left in DATATRIEVE without being connected to a data dictionary.

If you have readied a domain in your current dictionary, and you change dictionaries, that domain is still available when you are connected to the new dictionary. This feature allows you to move records from that readied domain into another domain in the new current dictionary. For a description of this process of transferring records, see Chapter 16.

If you want to return to the default data dictionary, to which you were connected at the beginning of your DATATRIEVE session, issue the SET DICTIONARY command without a file specification:

```
DTR> SET DICTIONARY
```

The following dialog illustrates the setting and displaying of data dictionaries. Note that if you try to SET a dictionary that has not been defined, DATATRIEVE prints an error message on your terminal. The dictionary must be defined first with a DEFINE DICTIONARY command. Note also that DATATRIEVE returns the DFN> prompt when you omit the file specification from the DEFINE DICTIONARY command. In this example ddn1: represents the name of the device on which the default DATATRIEVE dictionary is stored, and ddn2: is the name of the device on which your file directory is stored.

```
DTR> SHOW DICTIONARY
The current dictionary is ddn1:[1,2]QUERY.DIC;2
DTR> SET DICTIONARY NEWDIC
File "NEWDIC" not found
%OPEN failed
DTR> DEFINE DICTIONARY
DFN> NEWDIC
DTR> SHOW DICTIONARY
The current dictionary is ddn2:[200,200]NEWDIC.DIC;1
DTR> SET DICTIONARY
DTR> SHOW DICTIONARY
The current dictionary is ddn1:[1,2]QUERY.DIC;2
DTR>
```

Note that an explicit SET DICTIONARY NEWDIC command is not needed after the DEFINE DICTIONARY NEWDIC, because that DEFINE sets the dictionary to NEWDIC automatically.

RSTS/E systems suppress the PPN when showing the default dictionary. VAX/VMS systems may use your name as UIC of the dictionary in your directory in place of the six-digit number shown in the example.

10.4 Maintaining a Data Dictionary

10.4.1 Displaying Dictionary Contents

You can list the names of all domains, records, procedures, and description tables defined in the current data dictionary by using the SHOW command:

```
DTR> SHOW ALL
```

SHOW ALL also lists the names of all established collections and readied domains.

To show the definition of a dictionary object, you must have R (read) access to it. Use the SHOW command in the following format:

```
                  { domain-name      }
DTR> SHOW         { rec-name         }  [ (passwd) ]
                  { proc-name        }  [ (*)      ]
                  { desc-table-name  }
```

The SHOW command in this format prints the definition of the specified dictionary object on your terminal. It does not, however, print the password table associated with that dictionary object. To print the password table, use the SHOWP command:

```
                  { domain-name      }
DTR> SHOWP        { rec-name         }  [ (passwd) ]
                  { proc-name        }  [ (*)      ]
                  { desc-table-name  }
```

You must have C (control) access to the dictionary item to print its password table.

10.4.2 Modifying Dictionary Contents

To modify the definition of a dictionary object, you must have C (control) access to the dictionary object. To modify the dictionary definition of a domain, record, procedure, or description table, use one of the following methods:

- Delete the definition from the data dictionary using the DELETE command. Then, issue a DEFINE command to create a new definition.
- Copy the definition to an indirect command file using EXTRACT. Edit the indirect command file using a text editor. You cannot edit an indirect command file with the DATATRIEVE editor. After making the needed changes, return to DATATRIEVE, and execute the indirect command file as explained in Chapter 12.

  The EXTRACT command adds both a DELETE command and a DEFINE command to the beginning of the indirect command file. The DELETE removes the old definition of the dictionary object from the current data dictionary, and the DEFINE creates the new one.

You can also modify procedures and description tables by using the DATATRIEVE editor (see Chapter 8).

You should use extreme caution when changing record definitions. You can change a record definition in such a way that you can no longer access the data. When changing a record definition:

- You should use the same record length.
- You should not add or delete elementary fields.
- You should not change the data type of an elementary field.
- You can add or delete group fields and COMPUTED BY fields.
- You can add VALID IF clauses.
- You can change edit-strings.
- You can change the names of fields, and add or delete query names and query headers.

10.4.3 Deleting Dictionary Contents

To remove the definition of a dictionary object from the current data dictionary, you must have C (control) access to the dictionary object. To remove a dictionary definition use the DELETE command:

```
                  { domain-name      }
DTR> DELETE       { rec-name         }  [ (passwd) ]  ;
                  { proc-name        }  [ (*)      ]
                  { desc-table-name  }
```

Remember to terminate the DELETE command with a semicolon.

DELETE removes from the dictionary both the definition of the dictionary object and its associated password table. The data or records in the data file for the domain are not affected by this command.

10.5 Compressing a Dictionary

As definitions are added to and deleted from a dictionary, unused areas of disk space accumulate. This wasted disk space can be reclaimed by running the utility program QCPRS. QCPRS compresses any indexed sequential file, and preserves the file's contents while eliminating unused disk space.

To compress a data dictionary, invoke QCPRS in response to the prompt of your operating system (not in response to any DATATRIEVE prompt). QCPRS prints an identification message and requests a command with the following prompt:

```
CPR>
```

Enter the name of the dictionary to be compressed in the following format:

```
new-file=old-file
```

The argument new-file is the file specification for the compressed copy of the dictionary. The argument old-file is the file specification of the dictionary to be compressed. For both arguments, if you omit a field in the file specification, QCPRS uses the following defaults:

Field	Default
dev:	SY: (the system device)
UFD	Your login UIC/PPN
filename	QUERY
.ext	.DIC

Under all operating systems but RSTS/E, the file specifications for new-file and old-file can be the same. QCPRS merely creates a new copy of the file with the next higher version number.

Under RSTS/E, the file specifications for new-file and old-file must be different, and you should rename the dictionary file before invoking QCPRS. For instance, you might change the file extension to .BAK. When you invoke QCPRS, you can give the new file the original file specification.

After you have entered the file specifications, QCPRS asks you to specify a number of disk blocks as an allocation for the new version of the dictionary:

```
ENTER ALLOCATION FOR AREA 0:
```

Enter an estimate of the number of disk blocks needed for the compressed version. If your guess is too low, the file is automatically extended to hold the contents of the original file. If your guess is too high, the extra blocks remain in the file and give room for contiguous expansion of the dictionary.

QCPRS then prompts again with CPR>, and you can compress another dictionary file or terminate QCPRS with CTRL/Z.

Because QCPRS does not alter the contents of the original file, you can save the file as a backup, or you can delete it.

The following dialog illustrates the use of QCPRS after renaming the dictionary to be compressed from QUERY.DIC to QUERY.BAK:

```
RUN QCPRS
QUERY FILE COPY - COMPRESS UTILITY
CPR> QUERY.DIC=SY:[1,2]QUERY.BAK
ENTER ALLOCATION FOR AREA 0: 250
CPR> CTRL/Z
```

Chapter 11
Creating Record Definitions

A record is a group of related data. For example, in the YACHTS domain, each record contains the data relating to one yacht. A record must contain at least one unit of data, called a "field." Usually, though, a record contains several fields.

Every domain has an associated record definition that describes the fields in its records. You can create a record definition using the DEFINE RECORD command, described in Chapter 5, or the Application Design Tool, described in Chapter 9. This chapter explains the concepts and techniques for creating record definitions and explains how to define records that store data in the form you need it.

If you have defined records using COBOL, then DATATRIEVE's record definition will look familiar to you. But, a knowledge of COBOL is not necessary to create record definitions in DATATRIEVE.

11.1 The Parts of a Record Definition

A record definition consists of one or more field definitions. A field definition describes a field in the record and its relationship to other fields. Every field definition contains at least two parts:

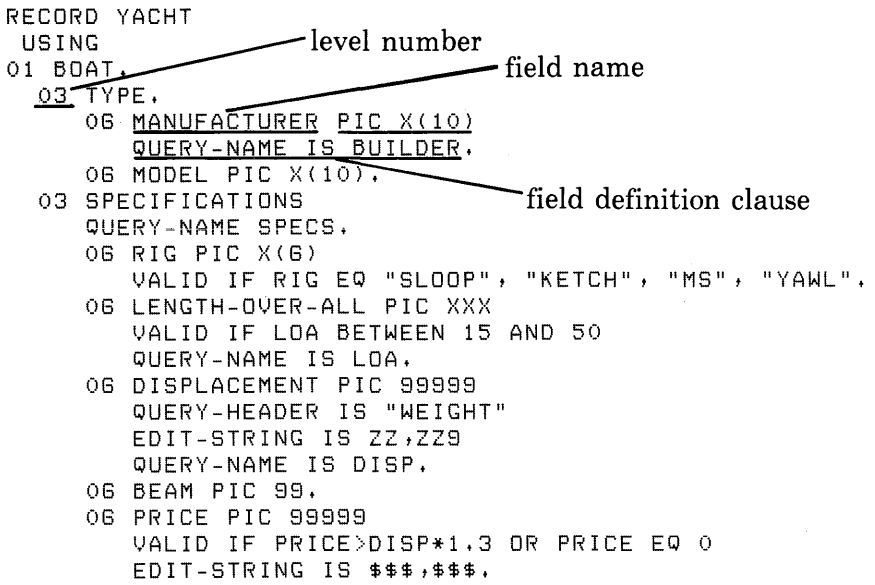
- A level number, which indicates the relationship between the field and other fields in the record
- A field name, which identifies the field to you and to DATATRIEVE

Some fields need a third part for their definition:

- One or more field definition clauses

Figure 11–1 shows the full YACHT record definition and illustrates the three parts of a record definition.

Figure 11-1: Three Parts of YACHT Record




```
RECORD YACHT
 USING
01 BOAT.
  03 TYPE.
     06 MANUFACTURER PIC X(10)
        QUERY-NAME IS BUILDER.
     06 MODEL PIC X(10).
  03 SPECIFICATIONS
     QUERY-NAME SPECS.
     06 RIG PIC X(6)
        VALID IF RIG EQ "SLOOP", "KETCH", "MS", "YAWL".
     06 LENGTH-OVER-ALL PIC XXX
        VALID IF LOA BETWEEN 15 AND 50
        QUERY-NAME IS LOA.
     06 DISPLACEMENT PIC 99999
        QUERY-HEADER IS "WEIGHT"
        EDIT-STRING IS ZZ,ZZ9
        QUERY-NAME IS DISP.
     06 BEAM PIC 99.
     06 PRICE PIC 99999
        VALID IF PRICE>DISP*1.3 OR PRICE EQ 0
        EDIT-STRING IS $$$,$$$.
```

The rest of this chapter describes the two types of fields (elementary and group), the parts of the record definition, and the types of data that a field can contain. A record definition clause, which is optional, is explained in Appendix E.

11.2 Elementary and Group Fields

If you have been working with the YACHTS domain, you have probably seen that a record can contain two types of fields: elementary fields and group fields. An elementary field is a basic unit of data. It contains no other field within it. A group field, on the other hand, contains one or more other fields.

For example, the YACHT record contains seven elementary fields:

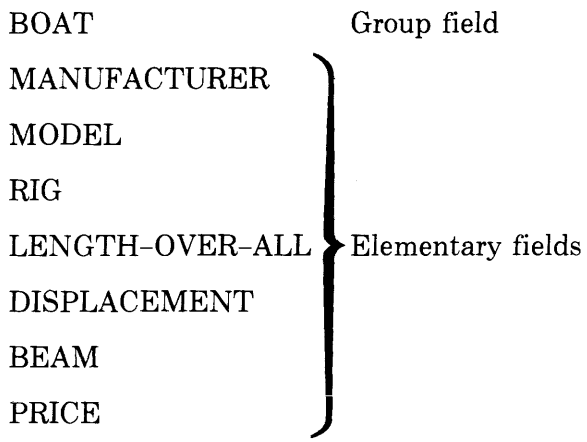
MANUFACTURER
MODEL
RIG
LENGTH-OVER-ALL
DISPLACEMENT
BEAM
PRICE

} Elementary fields

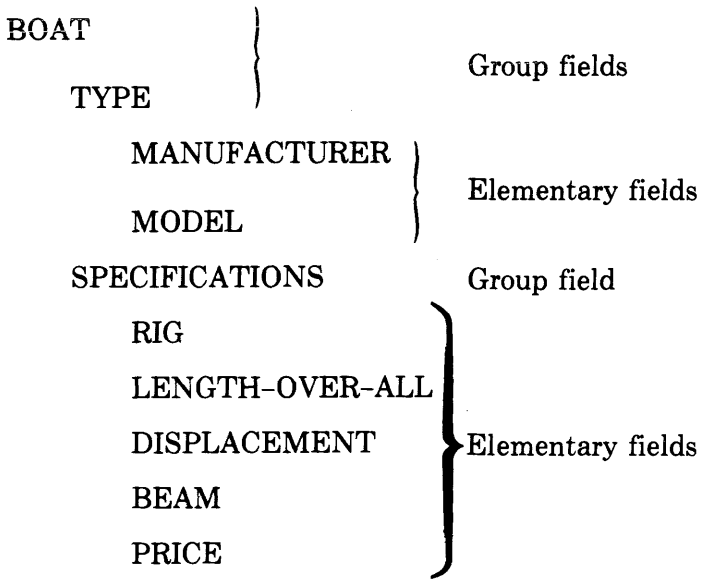
A record definition must contain at least one elementary field.

The YACHT record also contains group fields. A group field provides a convenient way to refer to more than one field in a record. If a record definition

contains a group field, you can refer to the group field in a statement, instead of referring to each field it contains. For example, in the YACHT record, the group field BOAT allows you to refer to every field in the record by the group field name BOAT.



Also for convenience, the YACHT record definition groups the first two elementary fields under a single name: TYPE. TYPE then refers to both MANUFACTURER and MODEL. The remaining five fields are grouped under the name SPECIFICATIONS:



Thus, you can use the group field name TYPE in a DATATRIEVE statement instead of specifying both MANUFACTURER and MODEL. For example, the following two PRINT statements provide the same results.

```
PRINT TYPE
PRINT MANUFACTURER, MODEL
```

Every record definition must meet the following requirements for including elementary and group fields:

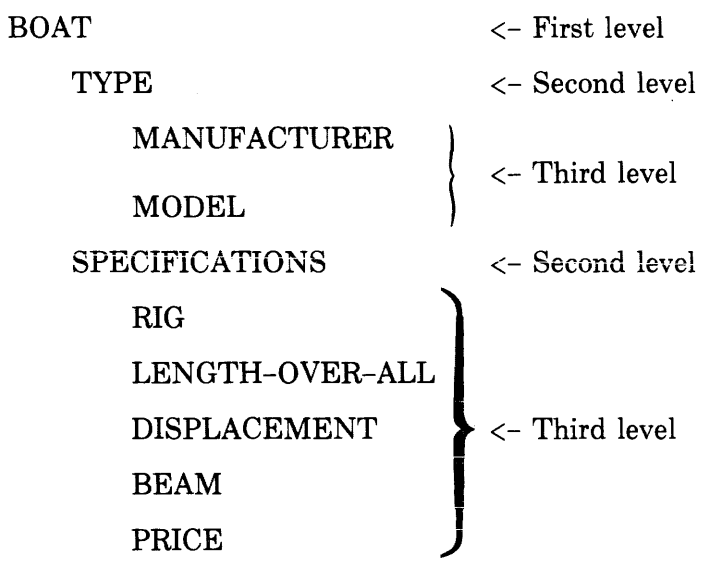
1. A record definition must contain at least one elementary field.
2. A record definition must contain a group field that includes all other fields in the record.
3. A group field must contain at least one field — either elementary or group.
4. A group field can contain both elementary and group fields.

11.3 Field Levels and Level Numbers

11.3.1 Field Levels

Every field in a record definition has an associated level, which describes the relationship of that field to other fields in the record. A group field that contains all other fields in a record is at the first (or highest) level. In the YACHT record, BOAT contains all other fields in the record and is at the first level.

A field that is contained within a first-level field (and no other field) is subordinate to the first-level field. It is at the second level in the record. For example, TYPE and SPECIFICATIONS are both at the second level, subordinate to BOAT. Any field subordinate to a second-level field is at the third level, and so on. Thus, MANUFACTURER and MODEL are at the third level.



11.3.2 Level Numbers

In a record definition, you must specify every field's level with a level number. A level number is a one- or two-digit number (from 1 to 65) that precedes the field name. It describes the field's level. The lower the field's level number,

the more inclusive is the field. For example, a level number of 1 indicates a field that includes all other fields. The first field in a record must have the lowest level number (usually 1) and it must be a group field that contains all the other fields.

The fields in the YACHT record definition illustrate levels and level numbers:

```
01  BOAT
    03  TYPE
        06  MANUFACTURER
        06  MODEL
    03  SPECIFICATIONS
        06  RIG
        06  LENGTH-OVER-ALL
        06  DISPLACEMENT
        06  BEAM
        06  PRICE
```

BOAT is at the highest level and has the lowest level number (01). The next higher level number is 03; therefore, TYPE and SPECIFICATIONS are at the second level, directly subordinate to BOAT. Within TYPE, two fields have a higher level number: MANUFACTURER and MODEL. These two are directly subordinate to TYPE. Similarly, SPECIFICATIONS contains five subordinate fields: RIG, LENGTH-OVER-ALL, DISPLACEMENT, BEAM, and PRICE.

The level numbers in YACHT illustrate several rules about assigning level numbers to a field:

1. Level numbers need not be consecutive.

   Only the relative value of level numbers determines the level and subordination of a field. For example, TYPE and SPECIFICATIONS could be at level number 02 and the fields they contain at any higher level number.

2. All fields directly subordinate to a group field must have the same level number. For example, MANUFACTURER and MODEL have the same level number since they are both directly subordinate to TYPE. And, the five fields subordinate to SPECIFICATIONS have the same level number. (That level number can be different from the level number of MANUFACTURER and MODEL.) Because TYPE and SPECIFICATIONS are directly subordinate to the same group field (BOAT), they have the same level number.

3. Only level numbers determine the level of a field.

   All the examples of YACHT thus far have shown field names indented to show the level of fields. Although indentation can make a record definition easier to read, it has no effect on the level of a field.

11.4 Field Names

In addition to a level number, every field in a record must have a field name. You use the field name to identify the field in DATATRIEVE statements. If you include the QUERY-NAME clause in the field definition, you can use the query name instead of the field name. DATATRIEVE also uses the field name when printing the field's content: if there is no QUERY-HEADER clause in the field definition, DATATRIEVE uses the field name as a column header for the field data. If you include the QUERY-HEADER clause in the field definition, DATATRIEVE uses the query header, instead of the field name, to make the column header. You can override the printing of the field name or query header by including a column header in the PRINT statement.

A field name must conform to DATATRIEVE's rules for names, described in Chapter 4. In summary, these rules are:

1. A name can consist of letters, digits, hyphens (-), and underscores (_).
2. The name must begin with a letter.
3. The name must end with a letter or digit.
4. The name cannot duplicate a DATATRIEVE keyword. See Appendix A for a list of keywords.
5. The name must be 30 characters or less.
6. The name can be continued from one line to another only by using a hyphen.

Although a field name can be 30 characters long, short field names are easier to use than long ones. If you must use a long field name, such as LENGTH-OVER-ALL, you may want to include a query name, such as LOA, in the field definition. The QUERY-NAME clause is described in Section 11.13. You can then use either the field name or the query name in a statement to refer to the field.

11.4.1 Qualifying Field Names

Field names need not be unique; two or more fields can have the same field name in a record definition. But, duplicate field names must be in different group fields.

If you use a duplicated field name in a command or statement, DATATRIEVE uses the first field in the record with that name. For example, the following record definition defines two fields named BIRTHDAY:

```
05  OWN.
    07  BIRTHDAY USAGE DATE.
    07  HEIGHT PIC 9V99.
05  SPOUSE.
    07  BIRTHDAY USAGE DATE.
    07  WEDDING-DATE USAGE DATE.
```

You can print the value in the first BIRTHDAY field with the following statement:

```
PRINT BIRTHDAY
```

You can refer to the second BIRTHDAY field in the record by qualifying the field name. You qualify a field name by preceding it with a group name and a period. For example, the following statement prints the value in the BIRTHDAY field that is subordinate to the group field SPOUSE:

```
PRINT SPOUSE.BIRTHDAY
```

The level of qualification you specify in a statement depends on the level of group fields in the record definition. For example, a record definition can contain duplicate group field names as well as duplicate elementary field names.

```
03  SON.
    05  OWN.
        07  BIRTHDAY USAGE DATE.
        07  HEIGHT PIC 9V99.
    05  SPOUSE.
        07  BIRTHDAY USAGE DATE.
        07  WEDDING-DATE USAGE DATE.
03  DAUGHTER.
    05  OWN.
        07  BIRTHDAY USAGE DATE.
        07  HEIGHT PIC 9V99.
    05  SPOUSE.
        07  BIRTHDAY USAGE DATE.
        07  WEDDING-DATE USAGE DATE.
```

You can refer to the first BIRTHDAY field in the record in any of the following ways:

```
BIRTHDAY

OWN.BIRTHDAY

SON.OWN.BIRTHDAY
```

You can refer to the first BIRTHDAY field in the group field DAUGHTER using either of the following qualified field names:

```
DAUGHTER.BIRTHDAY

DAUGHTER.OWN.BIRTHDAY
```

In summary, if a field name is duplicated and you refer to it without a qualifying group name, DATATRIEVE uses the first field in the record with that name. If you qualify a field name with a group field name, DATATRIEVE searches the first group field in the record with that name for the specified field.

11.4.2 FILLER Field Name

You can specify the keyword FILLER as the name of an elementary or group field. FILLER specifies a field that you will never refer to in a statement. Like other fields, a FILLER field must have a level number and can contain field definition clauses. Unlike other fields, you can use the field name FILLER more than once within the same group field. DATATRIEVE does not print the values stored in FILLER fields.

If the FILLER field is an elementary field, you cannot refer to its content in a DATATRIEVE statement. For example, in the following record definition, you can never refer to the third field in OWN:

```
03  SON.
    05  OWN.
        07  BIRTHDAY USAGE DATE.
        07  HEIGHT PIC 9V99.
        07  FILLER   PIC X(5).
```

If the FILLER field is a group field, you can refer to any field in the group, but not to the FILLER field itself. In the following example, you can refer to the SON, OWN, BIRTHDAY, and HEIGHT fields, but not the group or elementary FILLER fields.

```
02  FILLER.
    03  SON.
        05  OWN.
            07  BIRTHDAY USAGE DATE.
            07  HEIGHT PIC 9V99.
            07  FILLER   PIC X(5).
```

11.5 Field Classes

DATATRIEVE classifies every field by the type of data it contains or the way in which it stores data. DATATRIEVE has four classes of fields.

- Alphanumeric fields can store any combination of characters. The MANUFACTURER and MODEL fields in the YACHT record are examples of alphanumeric fields. If an alphanumeric field contains only digits, it can be used in arithmetic computations.
- Numeric fields are made up of digits and an optional sign (+ or –). The YACHT record, for example, contains the numeric fields DISPLACEMENT, BEAM, and PRICE. Numeric fields can be used in computations.
- DATE fields contain a date. A DATE field can be used in some computations. For example, you can subtract one date from another to get the number of days elapsed during a time period.
- COMPUTED BY fields do not store data, but merely describe a computation. A COMPUTED BY field does not occupy space in a record; it exists solely in the record definition. DATATRIEVE computes the value of the field when you refer to it in a command or statement.

An elementary field can be an alphanumeric, numeric, DATE or COMPUTED BY field. A group field, however, is always considered alphanumeric. You cannot use a group field in any arithmetic computation.

Table 11-1 summarizes the field classes and their acceptable content.

Table 11-1: Field Classes

Field Type	Class	Content
Elementary field	Alphanumeric	Any combination of characters
	Numeric	Any combination of digits and an optional sign (+ or -)
	DATE	A date
	COMPUTED BY	None; the definition specifies a computation, but no value is stored in the record
Group field	Alphanumeric	The fields subordinate to the group field

11.6 Field Definition Clauses

You define a field by specifying its level number, its field name, and, if it is an elementary field, one or more field definition clauses. A field definition clause is a keyword (such as PICTURE or QUERY-NAME) followed by a character string or value expression. A field definition clause can describe any of the following:

- The class and size of data a field contains and the way in which the data is stored
- The format of the data when it is printed
- The values to be accepted when data is entered in the field
- The way numeric values are computed when you refer to the field
- An alternate name for the field when you refer to the field or print its content
- An alternate way to handle the field

An elementary field must have at least one field definition clause. A group field can, but need not, have a field definition clause.

Table 11-2 contains a list of all field definition clauses. Three clauses (OCCURS, QUERY-NAME, and REDEFINES) can be used for both elementary and group fields. All other clauses can be used for elementary fields only.

Table 11-2: Summary of Field Definition Clauses

Clause	Valid for:	Purpose
COMPUTED BY	Elementary field	Describes a COMPUTED BY field
EDIT-STRING	Elementary field	Specifies the format of a field value when it is printed
OCCURS	Elementary or group field	Defines multiple occurrences of a field or group of fields
PICTURE	Elementary field	Specifies the format of a field value as it is stored in a record
QUERY-HEADER	Elementary field	Specifies the column header for the field value when it is printed
QUERY-NAME	Elementary or group field	Specifies an alternate name for a field
REDEFINES	Elementary or group field	Creates an alternate definition for a field
SIGN	Numeric elementary field	Specifies the location and representation of the sign in a numeric field
USAGE	Numeric or date elementary field	Specifies the internal storage format of a field or specifies a date field
VALID IF	Elementary field	Tests a value before storing it in a field

An elementary field must contain one of the following clauses:

PICTURE

COMPUTED BY

USAGE DATE

USAGE COMP-1 (or REAL)

USAGE COMP-2 (or DOUBLE)

When you define a field, you must end its definition with a period. If the field is a group field with no field definition clause, place the period immediately after the field name. For example, the group field BOAT contains no field definition clauses:

```
01 BOAT.
```

If the field definition contains clauses, place the period after the last field definition clause. For example, the LENGTH-OVER-ALL field contains three field definition clauses:

```
06  LENGTH-OVER-ALL PIC XXX
    VALID IF LOA BETWEEN 15 AND 50
    QUERY-NAME IS LOA.
```

The first field definition clause can be on the same input line as the level number and field name, as shown in the preceding example, or on a separate

input line. Delimit all clauses with a space, tab, or carriage return. Entering the clauses on separate lines and indenting them with spaces or tabs improve the readability of the record definition.

The rest of this chapter describes each field definition clause. The clauses are described in alphabetical order, but you can specify the clauses in any order in the field definition.

11.7 COMPUTED BY Clause

Function

Describes a COMPUTED BY field.

Format

```
COMPUTED BY val-exp
```

Arguments

val-exp
: Is a value expression consisting of one or more field or query names, numeric literals, and arithmetic operators.

Restrictions

1. This clause is valid for elementary fields only.
2. You cannot use an OCCURS clause or a VALID IF clause in a COMPUTED BY field.

Description

A COMPUTED BY field is useful if you will be printing an arithmetic computation frequently. It allows you to specify the computation once in the record definition, then print its value simply by referring to its field name. Generally, the computation includes the name of one or more fields in the record definition. When you refer to the COMPUTED BY field in a statement, DATATRIEVE evaluates the value expression using the field values in that record.

A COMPUTED BY field does not exist or occupy space in a record. It exists solely in the record definition. The value of a COMPUTED BY field is derived only when you use it in a statement.

Because it does not exist in the record, you cannot assign a value to a COMPUTED BY field. STORE and MODIFY statements cannot refer to a COMPUTED BY field.

You cannot redefine a COMPUTED BY field with the REDEFINES clause.

You cannot use a COMPUTED BY field in a sort list. That is, a COMPUTED BY field cannot be a sort key in a SORT or SUM statement or a record selection expression. In addition, you cannot use the field as a control group in the Report Writer statements AT TOP OF and AT BOTTOM OF.

If you include a COMPUTED BY clause in a field definition, you do not have to include any other field definition clause. However, you may want to specify the format of the computation when it is printed using an EDIT-STRING clause.

Examples

1. Compute the price per pound of a yacht as the price divided by the displacement. In this case, both the PRICE and DISP fields are defined in the record definition.

```
06 PRICE-PER-POUND
   EDIT-STRING $$$,$$9.99
   COMPUTED BY PRICE/DISP.
```

   When the PRICE-PER-POUND field is used in a command or statement, DATATRIEVE divides the value of the record's PRICE field by the value of its DISP field. The result of the computation is then the value of the PRICE-PER-POUND field.

2. Derive the value of the SALESMAN field from a description table named SALESMEN.

```
06 SALESMAN
   EDIT-STRING IS X(20)
   COMPUTED BY MANUFACTURER VIA SALESMEN.
```

   In this example, DATATRIEVE uses the value of the MANUFACTURER field in the current record to search the description table SALESMEN for a matching code. If one is found, DATATRIEVE uses its description as the value of the SALESMAN field.

11.8 Edit-String Clause

Function

Specifies the format of the field value when it is printed.

Format

EDIT-STRING [IS] edit-string

Arguments

edit-string
: Is one or more edit-string characters describing the format of the field value when it is printed.

Restrictions

This clause is valid for elementary fields only.

Description

The EDIT-STRING clause specifies the default format DATATRIEVE uses when printing a field value. The PICTURE or USAGE clause specifies the field's internal format. If there is no EDIT-STRING clause, the PICTURE clause determines the default format. However, DATATRIEVE does not print a sign or an implied decimal point specified in a PICTURE clause, unless you specify those characters in an edit string. (To override the default format specified by a PICTURE or EDIT-STRING clause, specify an edit string in the PRINT statement.)

You specify the format of the field value with a string of one or more edit characters. Specify the edit characters as a string without embedded spaces. In general, each edit character corresponds to one character position in the printed output. For example, 999999 specifies that the output will be six digits in six character positions.

To enter more than one of the same edit character, you can shorten the edit string by placing a repeat count in parentheses following the edit character. For example, the edit string 9(6) is equal to 999999.

Table 11-3 contains a list of edit-string characters. The edit-string characters you can specify for a field depends on the class of the field: alphanumeric, numeric, or date.

Table 11-3: Edit-String Characters

Field Class	Edit-String Character	Description
Alphanumeric	X	Each X is replaced by one character from the field's content.
	B	A space is inserted in that character position.
	/ (slash)	A slash is inserted in that character position.
	- (hyphen)	A hyphen is inserted in that character position.
	T	Indicates text. The number of Ts is the length of a line. DATATRIEVE prints the entire field using lines of the length you indicate. Words are not broken; if a word is longer than the line length it is printed on a line of its own. Edit strings containing a T cannot contain other characters.
Numeric	Z	If a Z matches a leading zero in the field's content it is replaced by a blank. If not, Z is replaced by a digit from the field's content.
	9	Each 9 is replaced by one digit from the field's content.
	* (asterisk)	If an asterisk matches a leading zero in the field's content a * is placed in that character position. If not, it is replaced by a digit from the field's content.
	0 (zero)	A 0 is placed in that character position.
	, (comma)	If all the digits to the left of the comma are suppressed zeros, the comma is replaced by a blank. If not, a comma is inserted in that character position.

(continued on next page)

Table 11-3: Edit-String Characters (Cont.)

Field Class	Edit-String Character	Description
	. (period)	A decimal point is inserted in that character position. An edit string for a numeric field can contain only one period.
	- (minus)	If only one minus sign is specified, it is replaced by either a blank (if the field's content is positive) or a minus sign (if it is negative). If more than one minus sign is specified (such as - -), the minus sign is a floating character.
	+ (plus)	If only one plus sign is specified, it is replaced by either a plus sign (if the field's content is positive) or a minus sign (if it is negative). If more than one plus sign is specified (such as ++), the sign (plus or minus) is a floating character.
	$	If only one dollar sign is specified, it is replaced by a $ in that character position. If more than one dollar sign is specified (such as $$), the dollar sign is a floating character.
	CR	If the field's content is positive, the letters CR are inserted. If the field's content is negative, CR is replaced by 2 blanks. (An edit string can contain only one CR.)
	DB	If the field's content is negative, the letters DB are inserted. If the field's content is positive, DB is replaced by 2 blanks. (An edit string can contain only one DB.)
	%	A percent sign is inserted in that character position.
Date	D	Each D is replaced by the corresponding digit of the day of the month. An edit string should contain no more than 2 Ds; the use of DD is recommended.
	M	Each M is replaced by the corresponding letter of the name of the month. An edit string of M(9) prints the entire name of the month. An edit string should contain no more than 9 Ms.
	N	Each N is replaced by a digit of the number of the month. An edit string should contain only two Ns; the use of NN is recommended.
	Y	Each Y is replaced by the corresponding digit of the numeric year. An edit string should contain no more than 4 Ys; the use of YY or YYYY is recommended.
	J	Each J is replaced by the corresponding digit of the Julian date. An edit string should contain no more than 3 Js; the use of JJJ is recommended.
	W	Each W is replaced by the corresponding letter from the name of the day of the week. An edit string of W(9) prints the entire day. An edit string should contain no more than 9 Ws.
	B	Each B is replaced by a space in that character position.
	/ (slash)	A slash is inserted in that character position.
	- (hyphen)	A hyphen is inserted in that character position.
	. (period)	A period is inserted in that character position.

11.8.1 Alphanumeric Fields

The edit string for an alphanumeric field specifies the number and type of characters to be printed, except for T edit strings, which print all the characters. The edit character X is replaced by one character from the field's content. The characters are transferred from the field to the output in left-to-right order.

If the edit string contains X and the field's content has more characters than its edit string, only the leftmost characters are printed. If it has fewer characters than its edit string, DATATRIEVE pads the output with blanks on the right.

If the field contains only digits, you may use a numeric edit string. The restrictions on edit strings for numeric fields are explained in Section 11.8.2. If you use a field in any arithmetic computations, you should define it as a numeric field. You may perform computations with alphanumeric fields that contain only digits, but, because alphanumeric fields are padded with spaces on the right, the results may not be valid.

The edit-string characters B, slash, and hyphen allow you to insert those characters in the output, in the position you indicate in the edit string. The B and slash characters are also valid for numeric edit strings, but not T edit strings. The hyphen is only valid for a field with an edit string containing X.

The following example contains some sample edit strings and the format of DATATRIEVE's output for two field values: CHALLENGER and 123. The example also shows the picture-string characters used in the field's PICTURE clause to specify the internal format of the field value.

Picture String	Edit String	Output If Field Value Is: CHALLENGER	123
X(10)	X(10)	CHALLENGER	123#######
X(10)	X(3)	CHA	123
X(10)	XX/X(8)	CH/ALLENGER	12/3#######
X(10)	X(5)/X(5)	CHALL/ENGER	123##/#####
X(10)	X(5)-XX	CHALL-EN	123##-##

Note: # represents a space.

Printing Long Field Values

The edit character T allows you to print alphanumeric field values on one or more lines. The primary use of T is to print fields containing large amounts of text. The number of Ts in the edit string indicates the maximum number of characters to be printed on one line. For example, the edit string T(20) indicates that a line of output will contain no more than 20 characters (unless a single word contains more than 20 characters).

If the field contains more characters than specified in the edit string, DATATRIEVE prints as many full words on the line as possible. (A word, in

this sense, is a string of characters delimited by a space.) DATATRIEVE then prints the remaining characters on the next line and following lines, if necessary. DATATRIEVE prints only full words; it does not divide or hyphenate words. DATATRIEVE does not print out trailing spaces when you use a T edit string.

For example, the following field definition describes a field containing 150 alphanumeric characters. The edit string specifies that when the field is printed, an output line will contain no more than 20 characters.

```
03 YACHT-DESCRIPTION
   PIC IS X(150)
   EDIT-STRING IS T(20).
```

The content of the YACHT-DESCRIPTION field could be a definition of "yacht":

A relatively small sailing or mechanically driven ship with a sharp prow and graceful lines, ordinarily used for pleasure cruising or racing.

The following example shows the format of the output when it is printed:

```
DTR> PRINT YACHT-DESCRIPTION

       YACHT
    DESCRIPTION

A relatively small
sailing or
mechanically driven
ship with a sharp
prow and graceful
lines, ordinarily
used for pleasure
cruising or racing.
```

If a word in the field content is longer than the edit string, DATATRIEVE prints the word, even though it exceeds the length of the edit string. For example, the following field definition specifies that an output line will contain no more than 10 characters:

```
03 YACHT-DESCRIPTION
   PIC IS X(150)
   EDIT-STRING IS T(10).
```

The same field content would be printed as:

```
DTR> PRINT YACHT-DESCRIPTION

       YACHT
    DESCRIPTION

A
relatively
small
sailing or
mechanically
driven
ship with
a sharp
prow and
graceful
lines,
ordinarily
used for
pleasure
cruising
or racing.
```

When you print a long field value such as this, the field name should be the last item in the print list. If another print item follows this field in the PRINT statement, its value will be printed on the same line as the last line of text — not the first line of text.

11.8.2 Numeric Fields

An EDIT–STRING clause prints numeric field values in a format that is easy to read. With edit strings you can suppress leading zeros and print dollar signs, percent signs, commas, decimal points, and plus or minus signs. For example, in the YACHT record, the EDIT STRING for the PRICE field clause causes DATATRIEVE to print the value 09870 as $9,870.

You can use three types of edit characters in edit strings for numeric fields: replacement characters, insertion characters, and floating characters.

Replacement Characters

The replacement characters for numeric fields are 9, Z, * (asterisk), and 0 (zero). Each 9, Z, or asterisk causes one digit from the field value to be printed. Each zero causes a zero to be printed, and the corresponding digit in the field value is discarded.

A leading zero is any zero in a field value which has only zeros to the left of it. If there is an implied decimal place to the left of a zero, then it is not a leading zero. If a Z corresponds to a leading zero in a field then a space is printed instead of a zero. If an asterisk corresponds to a leading zero in a field then an asterisk is printed instead of a zero. The following example shows the differences between the various replacement characters.

Picture String	Edit String	Field Content	Output
99999	9(5)	04092	04092
99999	Z(5)	04092	#4092
99999	*(5)	04092	*4092
99999	ZZZ00	04092	#4000
99V99	99.99	0001	00.01
99V99	ZZ.ZZ	0001	##.01

Note: # represents a space.

Before printing a field, DATATRIEVE computes the number of digits to the left of the decimal point. (If there is no V in the picture string, all digits are to the left of the decimal.) If there are more digits to the left of the decimal than the edit string specifies (either with replacement characters or floating characters) DATATRIEVE prints an asterisk in each character position specified by the edit string. If the field has fewer digits than the edit string specifies DATATRIEVE pads the output with leading zeros. (These are suppressed if you use Zs or floating characters.)

For example, a PICTURE clause could specify four digits for a field, and its edit string only two digits:

```
03 MODEL-NUMBER
   PICTURE IS 9999
   EDIT-STRING IS 99.
```

If the field value is 1234, DATATRIEVE prints two asterisks (**) for the field value. Therefore, be careful to specify edit strings that are long enough for numeric fields.

If you define a numeric field with the clauses PIC 9(4) and USAGE IS COMP, then the field can contain numbers as high as 32,768. To print the field when it contains a five digit number, you must use an edit string that specifies five digits.

Insertion Characters

With insertion characters you can print the sign of a field, a decimal point, a dollar sign, DB for a negative value, CR for a positive value, a percent sign, commas, and slashes.

Printing a Sign

To print a sign (+ or –) in a field value (indicated by an S in its PICTURE clause), you must specify a + or – in the edit string for that field. The sign must be the first or last character in the edit string.

If you specify only one sign in the edit string, DATATRIEVE prints the sign in the position you indicate. If you specify more than one sign, the sign is a floating character.

You can also use the edit characters DB and CR to indicate the sign of a field value. You can only use one DB or CR in an edit string, and it must be the first or last character in the edit string.

The following example shows the use of the edit characters +, –, DB, and CR:

Picture String	Edit String	Field Content	Output
S9999	(None)	–1234	1234
S9999	–9999	–1234	–1234
S9999	–9999	+1234	#1234
S9999	9999+	–1234	1234–
S9999	+9999	+1234	+1234
S9999	9999DB	–1234	1234DB
S9999	9999CR	–1234	1234##
S9999	CR9999	+1234	CR1234

Note: # represents a space.

Printing a Decimal Point

To print an implied decimal point in a field value (indicated by a V in its PICTURE clause), you must specify a period (.) in the edit string for that field. When DATATRIEVE prints the field content, it aligns all output on the decimal point.

DATATRIEVE matches the decimal point in the edit string with the implied decimal place in the field. If the edit string contains fewer digits to the right of the decimal, the extra digits are not printed. If the edit string contains fewer digits to the left of the decimal, DATATRIEVE prints asterisks. Thus it is best to place the period in the edit string in the same position as the V in the picture string.

The following example shows printing decimal points in several different numeric fields.

Picture String	Edit String	Field Content	Output
99V99	(None)	1234	1234
99V99	Z9.99	1234	12.34
99V99	999.9	1234	012.3
99V99	9.999	1234	*.***
99V99	9.999	0123	1.230
99V99	Z(4)	1234	12

If the last character of the edit string is a period and is not followed by other input on the same line, DATATRIEVE treats the period as the termination of the field definition and not as part of the edit string. If the EDIT STRING clause is the last part of the field definition, specify two periods; the first period is part of the edit string and the second period ends the field definition. If the EDIT STRING clause is not the last part of the field definition, place the next clause on the same line or place a hyphen at the end of the line.

Printing Other Characters

To print a comma, slash, percent sign, or dollar sign, specify that character in the edit string. If there are only spaces to the left of a comma, DATATRIEVE prints a space instead of a comma. If you include more than one dollar sign, it is a floating character. The following example shows how these characters are used:

Picture String	Edit String	Field Content	Output
99	99%	45	45%
9(6)	$999,999	100000	$100,000
9(6)	ZZZ,ZZZ	000040	#####40
9(6)	999/999	123456	123/456

Note: # represents a space.

Floating Characters

You can specify three edit-string characters ($, –, and +) as floating characters. A floating character replaces all but the last leading zero with spaces. The last leading zero is replaced by the edit-string character ($, –, or +). Floating characters that correspond to digits are replaced by those digits. Since one character is replaced by a t, –, or $, you must specify one more character than the number of digits in the field.

To use a floating character, you must specify two or more of the same character. You can specify only one floating character in an edit string. For example,

you cannot have both a floating minus sign and a floating dollar sign in the same edit string. The floating characters must be the leftmost characters in an edit string.

The following example shows the use of floating characters in edit strings:

Picture String	Edit String	Field Content	Output
S9(4)	++++9	+0187	#+187
S9(4)	++++9	-5764	-5764
S9(4)	----9	+0187	##187
S9(4)	----9	-0001	###-1
9(5)V99	$9(5).99	0015786	$00157.86
9(5)V99	$$$,$$$.99	0015786	##$157.86
9(5)V99	$$$,$$$.00	0015786	##$157.00
S9(5)	$$$,$$$CR	+54362	$54,362CR

Note: # represents a space.

11.8.3 Date Fields:

A field defined as USAGE IS DATE is stored internally as a binary value. To print this field, DATATRIEVE must convert it to another format. If you do not include an EDIT-STRING clause in the field definition for a date field, DATATRIEVE prints the field value in the following format:

DD-MMM-YY

To print the date in any other format, you must include an EDIT-STRING clause in the field's definition. (You can also specify the output format using the PRINT statement when you print the field.) The edit string for a date field gives you several formatting choices for printing a date. (You can also use the PICTURE clause to specify a date field's output format. However, the PICTURE clause does not provide the many formatting options of the EDIT-STRING clause.)

For example, you can print the date January 1, 1980, in any of the following formats:

```
1 Jan 80

1980/001

Tuesday /January 01

Jan 01 80

01/01/80

01-Jan-1980 Tue
```

There are many more formats that you can use to print a date. The date can include:

- The name of the day of the week (Monday, Tuesday, etc.) or just the first characters of the name (such as Mon, Tue, Wed)
- The day of the month (1, 2, 3, etc.)
- The name of the month (January, February, etc.) or just the first characters of the name (such as Jan, Feb, Mar)
- The number of the month (for example, 1 for January and 12 for December)
- The year (such as 1980) or just the last two digits of the year (80)
- The Julian date (for example, 001 for January 1 and 366 for December 31 in a leap year)
- Delimiters to separate the parts of the date (such as a slash after the month and day)

You specify the characters to be printed (letters, digits, spaces, slashes, hyphens, or periods) and the order in which the parts of the date are to appear (such as month, day, year). The edit-string characters that you use to specify the format of the date are shown in Table 11–3.

DATATRIEVE always outputs the number of characters you specify in the edit string. If the content of the field is shorter than its edit string specifies, DATATRIEVE pads the output with spaces or zeros until the length of the output equals the length of the edit string. For example, an edit string can specify a nine-letter month: M(9). If the field contains a month with a name shorter than nine letters (such as June), DATATRIEVE pads the output on the right with blanks. In this case, June would be printed as:

June#####

For numeric date fields, DATATRIEVE pads the output with zeros on the left.

The following table contains some edit strings and the format of the output for two field values: June 4, 1980 and November 27, 1978.

Output if Field Value Is:

Edit String	June 4, 1980	November 27, 1978
DD-MMM-YY	#4-Jun-80	27-Nov-78
MMMBDDBY(4)	Jun##4#1980	Nov#27#1980
M(9)BDDBY(4)	June#######4#1980	November##27#1978
NN/DD/YY	6/04/80	11/27/78
W(9)	Wednesday	Monday###
YYYY/JJJ	1980/156	1978/331
DDBMMMBYY/WWW	#4#Jun#80/Wed	27#Nov#78/Mon
DD.NN.YY	#4.06.80	27.11.78

Note: # represents a space.

11.9 OCCURS Clause

The OCCURS clause defines multiple occurrences (or repetitions) of a field or group of fields. The multiple occurrences, called a list, create a hierarchy in the domain. (See Chapter 16 for more information on hierarchies.)

The OCCURS clause has two formats: one format for a fixed number of occurrences; one for a variable number of occurrences. Each is described in the following sections.

11.9.1 Fixed Number of Occurrences

Function

Defines a fixed number of occurrences for a field or group of fields.

Format

OCCURS n TIMES

Arguments

n
Is a positive integer specifying the number of occurrences for the field.

Restrictions

A field definition cannot contain both an OCCURS and a COMPUTED BY clause. (You cannot specify multiple occurrences of a COMPUTED BY field.)

Description

This format of the OCCURS clause defines a list with a fixed number of occurrences. It reserves enough space in each record to contain all the occurrences of the field (or fields), whether or not they contain data. For example, the following field definition reserves enough space in every record for two occurrences of the elementary field KIDS-NAMES; each occurrence is 10 characters long.

```
03 KIDS-NAMES PIC X(10)
   OCCURS 2 TIMES.
```

Thus, you can store up to two names (containing up to 10 characters each) in any record.

The following definition specifies that the group field KIDS-NAMES is repeated twice. Each group field contains two fields: FIRST-NAME (10 characters long) and MIDDLE-INIT (1 character). A total of 22 characters is reserved in every record for the group field.

```
03 KIDS-NAMES OCCURS 2 TIMES.
   05 FIRST-NAME PIC X(10).
   05 MIDDLE-INIT PIC X.
```

In the record, the fields are stored in the following order:

```
KIDS-NAMES
   FIRST-NAME
   MIDDLE-INIT
KIDS-NAMES
   FIRST-NAME
   MIDDLE-INIT
```

A field in a group field with an OCCURS clause can contain an OCCURS clause. The result is a sublist: a list nested within a list. For example, you can reserve enough space in a record to store up to three nicknames for each KIDS-NAMES:

```
03 KIDS-NAMES OCCURS 2 TIMES.
   05 FIRST-NAME PIC X(10).
   05 MIDDLE-INIT PIC X.
   05 NICKNAME PIC X(10)
      OCCURS 3 TIMES.
```

The fields are stored in the following order:

```
KIDS-NAMES
   FIRST-NAME
   MIDDLE-INIT
   NICKNAME
   NICKNAME
   NICKNAME
KIDS-NAMES
   FIRST-NAME
   MIDDLE-INIT
   NICKNAME
   NICKNAME
   NICKNAME
```

A record definition can contain any number of OCCURS clauses in this format.

11.9.2 Variable Number of Occurrences

Function

Defines a variable number of occurrences of a group of fields.

Format

OCCURS min TO max TIMES DEPENDING ON fld-name

Arguments

min
: Is a non-negative integer specifying the minimum number of occurrences of the field. This value can be 0. DATATRIEVE does not check this value.

max
: Is a positive integer specifying the maximum number of occurrences of the field. This value must be greater than 0.

fld-name
Is the name of a field in the same record definition. The value of the field determines the number of occurrences of this field. The field must be a numeric field containing a non-negative integer; it cannot be defined with digits to the right of the decimal point.

Restrictions

1. No other field definition can follow the last elementary field in the group field containing this clause.
2. A record definition can contain only one OCCURS clause in this format.
3. A field definition cannot contain both an OCCURS and a REDEFINES clause or an OCCURS and a COMPUTED BY clause.

Description

This format of the OCCURS clause defines a list with a variable number of occurrences.

The number of occurrences of the group field in any record is equal to the value of the field specified in the OCCURS clause. Therefore, the sizes of records in the domain can vary. If you use the MAX argument when you define the data file, all records in the file have the same length. The DEFINE FILE command is described in Section 5.8.

The FAMILY-REC record definition shows the use of an OCCURS clause in this format:

```
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC 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 X(10) QUERY-NAME IS KID.
         09 AGE PIC 99 EDIT-STRING IS Z9.
```

The number of occurrences of the KIDS field depends on the value of the NUMBER-KIDS field. If the value is 0, there are no occurrences of the field; if it is 1, there is one occurrence, and so on. Each occurrence of KIDS contains three fields: the group field EACH-KID and the elementary fields KID-NAME and AGE.

A group field containing this format of the OCCURS clause can contain one or more fields with an OCCURS clause. The nested OCCURS clause, however, must specify a fixed number of occurrences of the field.

11.10 PICTURE Clause

Function

Specifies the format of the field value as it is stored.

Format

```
{PICTURE}
{PIC    } [IS]picture-string
```

Arguments

picture-string
Is one or more picture-string characters describing the format of the field value as it is stored.

Restrictions

This clause is valid for elementary fields only.

Description

The picture string controls the storage format of the field's content. In general, each character in the string corresponds to one character position in the field value. (For numeric fields, you can also include a USAGE clause to specify the internal format of the digits.) For example, for a numeric field without a USAGE clause, 999999 specifies six digits in six character positions, occupying six bytes of storage.

To enter more than one of the same picture character, you can shorten the string by placing a repeat count in parentheses following the picture character. For example, the picture string 9(6) is equal to 999999.

Specify the picture-string characters as a string. Do not embed a space in the picture string.

Table 11–4 contains a list of the picture-string characters. The picture-string characters you specify for a field depend on the class of the field: alphanumeric or numeric.

Table 11-4: Picture-String Characters

Field Class	Picture Character	Meaning
Alphanumeric	X	Each X represents one character in the field.
Numeric	9	Each 9 represents one digit in the field. You can specify from 1 to 18 digits for a numeric field.
	S	An S indicates that a sign (+ or –) is stored in the field. A picture string can have only one S and it must be the leftmost character. If there is no SIGN clause for the field, the sign shares the rightmost character position with the lowest-valued digit.
	V	A V indicates an implied decimal point. The decimal point does not occupy a character position in the field, although DATATRIEVE uses its location to align data in the field. A picture string can contain only one V.
	P	Each P specifies a decimal scaling position. Each P represents a "distance" in digits from an implied decimal point. (A P does not count toward the limit of 18 digits per field.) A P can appear at the right or left of the picture string. A V is unnecessary for any picture string containing a P.

11.10.1 Alphanumeric Fields

The picture string for an alphanumeric field specifies the number of characters in the field. Only the picture character X is valid in the picture string for an alphanumeric field. Each X corresponds to a single character position in the field. For example, the following field definition specifies the MODEL field contains ten alphanumeric characters:

```
06 MODEL PIC X(10).
```

11.10.2 Numeric Fields

A numeric field can contain the characters 9, S, V, and P in its picture string to specify: the number of digits in the field, a sign, an implied decimal point, and a decimal scaling factor.

Specifying the Number of Digits

The picture character 9 represents one digit in the field value. For example, the picture string 9(4) indicates four digits; the field value can range from 0 to 9999. There is one exception to this; if you specify PIC 9999 and USAGE IS COMP, then it possible to store numbers as large as 32,768. You can specify from 1 to 18 digits for a numeric field.

The following field definition specifies that the BEAM field contains two digits:

```
03 BEAM PIC 99.
```

Specifying a Sign

To specify that a numeric field can contain a sign (+ or -), you must include an S in its picture string. The S must be the leftmost character in the picture string; a picture string can contain only one S. For example, the picture string S9(4) indicates a signed field, four digits in length; the field value can range from -9999 to +9999.

The sign specified with the PICTURE clause is not printed unless you include an EDIT-STRING clause with the field definition. For example, the following field definition specifies a sign in the picture string, which is printed following the last digit of the field value:

```
03 CURRENT-BALANCE
   PICTURE IS S9999V99
   EDIT-STRING IS $$$$9.99-.
```

Specifying a Decimal Place

The picture character V specifies the position of an "implied" decimal point. For example, the picture string 9(5)V99 specifies a seven-digit field; the last two digits of the field value follow the decimal point.

The decimal point does not occupy a character position in the record; DATATRIEVE uses the implied decimal point in computations, Boolean expressions, and other arithmetic operations.

The implied decimal point is not printed unless you specify a period (.) in the EDIT-STRING clause for the field definition. For example, the following record definition specifies that a decimal point be printed before the last two digits in the PRICE field:

```
03 PRICE PIC 999V99
   EDIT-STRING IS $999.99.
```

If there is no V in the picture string, DATATRIEVE treats the field value as an integer (that is, as if a V were specified to the right of the rightmost digit). Thus, the picture strings 999 and 999V are equal.

Scaling Factor

The picture character P specifies a decimal scaling factor; that is, the number of digits from an implied decimal point. Each P in the picture string represents one position from the decimal point.

The P (or multiple Ps) must be the leftmost or rightmost characters in the picture string. If the P is the leftmost character(s), the decimal point is assumed to be to the left of the leftmost P. For example, the picture strings PPP99 and VPPP99 are equal. (If you specify a P in a picture string, the V character is optional.) If the P is the rightmost character(s), The decimal point is assumed to be to the right of the rightmost P. Thus, 999PP and 999PPV are equal.

DATATRIEVE treats each P in a picture string as a zero. The string PPP99 specifies that the field can contain values in the range 0 to .00099; the three leftmost digits are assumed to be zeros. The range of values for a field with a picture string of 999PP is 0 and 100 to 99,900, but the two rightmost digits are assumed to be zeros.

Scaling positions are not printed unless you include an EDIT-STRING clause. The implied decimal point in a picture string containing the P character is not printed unless you specify a period (.) in the EDIT-STRING clause in the field definition.

11.11 QUERY-HEADER Clause

Function

Specifies the column header to be used when the field is printed by the DATATRIEVE or Report Writer PRINT statement.

Format

QUERY-HEADER[IS]"header-1"[/"header-2"]...

Arguments

"header"
Is the column header to be printed above the column of field data. If you specify only one character string literal, that string is printed on one line above the column. If you specify more than one character string literal, they must be separated by a slash (/). The literals are printed on successive lines, centered above the column.

Restrictions

This clause is valid for elementary fields only.

Description

If you omit this clause, DATATRIEVE uses the field name as the default column header when printing this field.

The column header can include any character except carriage return, line feed, or control character. To include a quotation mark in a column header, precede it with a second quotation mark. (See Example 3, below.)

You can perform the same functions of the QUERY-HEADER clause with the DATATRIEVE or Report Writer PRINT statement. The following PRINT statement prints a column header above the specified field data.

```
PRINT fld-name("header-1"[/"header-2"],,,)
```

If the size of a record definition is a consideration, you might choose to use this format of the PRINT statement instead of including a lengthy query header in the record definition. See Section 5.25 for more information on the DATATRIEVE PRINT statement and Chapter 7 for the Report Writer PRINT statement.

Examples

1. When the DISPLACEMENT field is printed, use a column header of WEIGHT above the column of data.

```
06 DISPLACEMENT PIC 99999
    QUERY-HEADER IS "WEIGHT"
    EDIT-STRING IS ZZ,ZZ9
    QUERY-NAME IS DISP.
```

2. When the LENGTH-OVER-ALL field is printed, use a column header of LENGTH (IN FEET) on two separate lines.

```
06 LENGTH-OVER-ALL PIC XXX
   QUERY-HEADER IS "LENGTH" /"(IN FEET)".
```

DATATRIEVE prints the column header as:

```
 LENGTH
(IN FEET)
```

3. When the LENGTH-OVER-ALL field is printed, use a column header of Length-Over-All ("LOA") on two separate lines.

```
06 LENGTH-OVER-ALL PIC XXX
   QUERY-HEADER IS "Length-Over-All"/"(""LOA"")".
```

DATATRIEVE prints the column header as:

```
Length-Over-All
    ("LOA")
```

11.12 QUERY-NAME Clause

Function

Specifies an alternate name for the field.

Format

QUERY-NAME IS query-name

Arguments

query-name
Is the query name. The rules for forming and using a query name are the same as for those for a field name.

Restrictions

None. This clause is valid for both group and elementary fields.

Description

Use the QUERY-NAME clause when a field name is too long to use easily.

The query name must conform to the rules for names (see Section 11.4). Like a field name, a query name can duplicate another query name (or a field name) in the record. A query name can also be qualified by other query names or field names.

Examples

1. Define DISP as a alternate name for the DISPLACEMENT field.

```
06 DISPLACEMENT PIC 99999
   QUERY-NAME IS DISP.
```

2. Define SPECS as an alternate name for the group field SPECIFICATIONS.

```
03 SPECIFICATIONS
   QUERY-NAME SPECS.
```

3. Define a query name of SPECIAL-HANDLING for the field DELINQUENT-ACCOUNT-STATUS.

```
09 DELINQUENT-ACCOUNT-STATUS
   PIC X
   QUERY-NAME IS SPECIAL-HANDLING.
```

11.13 REDEFINES Clause

Function

Provides an alternate way to define a field.

Format

```
lev-no fld-name-1 REDEFINES fld-name-2
```

Arguments

lev-no
: Is the level number of fld-name-1. The level number is not part of the REDEFINES clause, but is shown in the format only to clarify its relative position.

fld-name-1
: Is the name of the field that provides the redefinition. The field name is not part of the REDEFINES clause, but is shown in the format only to clarify its relative position and function.

fld-name-2
: Is the name of the field being redefined.

Restrictions

1. The field to be redefined (fld-name-2) must appear before its redefinition (fld-name-1) in the record definition. Both fields must have the same level number.
2. The definition of fld-name-2 cannot contain a REDEFINES clause. However, it can be subordinate to a group field with a REDEFINES clause.
3. Neither fld-name-1 nor fld-name-2 can be defined with (or contain a field defined with) an OCCURS ... DEPENDING clause.
4. Neither field-name-1 nor fld-name-2 can contain a COMPUTED BY clause. (You cannot redefine a COMPUTED BY field.)
5. The REDEFINES clause must immediately follow the name of the field containing the redefinition (fld-name-1). No other clause can be used between the field name and the keyword REDEFINES.
6. The field providing the redefinition (fld-name-1) cannot describe an area larger than the area of fld-name-2. However, the area can be smaller than that of fld-name-2.

Description

The REDEFINES clause redefines an elementary or group field. The redefinition refers to the same area of the record as the original definition, but it uses the content of the field in a different way. For example, if you need to refer to parts of a numeric field as well as the field itself, you can redefine the field as a group field. The subordinate fields of the group fields would contain the parts of the numeric field value that you will refer to. Thus, the REDEFINES clause allows you to redefine a numeric field as a group field. (A group field cannot be numeric; a group field is always alphanumeric.)

The following record definition shows a redefinition of the field PART-NUMBER. PART-NUMBER is a numeric field containing ten digits. Two group fields redefine PART-NUMBER: PART-NUMBERS-PARTS and PART-NUMBER-GROUPS. Each redefinition specifies a group field containing a total of ten digits (the total number of digits in all subordinate fields).

```
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 SUB-ASSEMBLY PIC 99.
   07 PART-DETAIL PIC 999.
05 PART-NUMBER-GROUPS REDEFINES PART-NUMBER.
   07 PRODUCT-GROUP-ID PIC 9(4).
   07 PART-DETAIL-ID PIC 9(6).
```

In this example, the field PRODUCT-GROUP refers to the lowest-valued digits of PART-NUMBER; PRODUCT-YEAR to the next two lowest-valued digits, and so on.

11.14 SIGN Clause

Function

Specifies the location and representation of a sign (+ or –) in a numeric elementary field.

Format

```
SIGN[IS] { LEADING  } [SEPARATE]
         { TRAILING }
```

Arguments

LEADING
TRAILING
: Indicates that the sign is at the left (LEADING) or right (TRAILING) of the field value.

SEPARATE
: Indicates that the sign occupies its own character position in the field. If this argument is omitted, the sign shares a character position with the field's leftmost (if it is LEADING) or rightmost (TRAILING) digit.

Restrictions

1. This clause can be used only with numeric elementary fields.
2. A field definition cannot contain both a SIGN and a USAGE clause (other than USAGE IS DISPLAY).

Description

If you do not include a SIGN clause with a numeric field, the sign shares a character position with the field's rightmost digit. You must include this clause if a program written in COBOL (or other language) uses the record and requires that the sign be a separate character or share the leftmost character position. A sign clause does not affect the printing characteristics of a field.

Examples

1. Define the field CURRENT-BALANCE as a six-digit signed field, with the sign sharing the leftmost character position.

```
03 CURRENT-BALANCE
   PIC IS S9999V99
   EDIT-STRING IS $$$$9.99-
   SIGN IS LEADING.
```

2. Define the field NEW-PRICE as a four-digit signed field. The sign is a separate character in the rightmost character position.

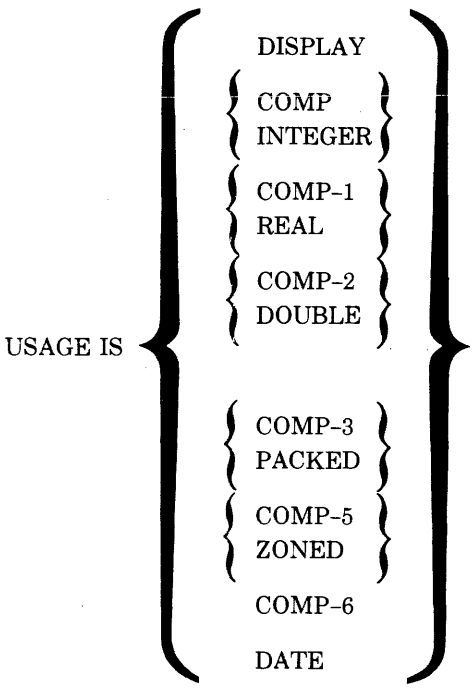
```
03 NEW-PRICE PIC S99V99
   SIGN TRAILING SEPARATE
   EDIT-STRING +++.99.
```

11.15 USAGE Clause

Function

Specifies the internal format of a numeric field or specifies a date field.

Format



Arguments

DISPLAY

Indicates that each digit occupies one byte of storage. DISPLAY is the default if you do not include a USAGE clause.

{COMP / INTEGER}

Indicates that the field value is stored in binary format. INTEGER is a synonym for COMP.

{COMP-1 / REAL}

Indicates that the field value is stored in single-precision real format. REAL is a synonym for COMP-1.

{COMP-2 / DOUBLE}

Indicates that the field value is stored in double-precision real format. DOUBLE is a synonym for COMP-2.

{ COMP-3 }
{ PACKED }

Indicates that the field value is stored in packed-decimal format. PACKED is a synonym for COMP-3.

{ COMP-5 }
{ ZONED }

Indicates that the field value is stored in signed decimal format. ZONED is a synonym for COMP-5.

COMP-6

Indicates that the field value is stored in PDP-11 COBOL V3 COMPUTATIONAL format. (COMP-6 has no synonym.)

DATE

Indicates that the field is a date field.

Restrictions

1. This clause is valid for elementary numeric or date fields only.
2. A field definition cannot contain both a USAGE clause (other than USAGE IS DISPLAY) and a SIGN clause.

General Description

If a numeric field definition does not have a USAGE clause, each digit in the field value occupies one character position in the record. Use the appropriate form of the USAGE clause when a program written in COBOL, BASIC-PLUS-2, or other language uses the record and requires a different internal format.

The USAGE IS DATE clause identifies the field as a DATATRIEVE date field.

The USAGE clause can cause DATATRIEVE to align fields on hardware storage boundaries. Refer to Appendix E for more information on alignment.

11.15.1 COMP (or INTEGER) Fields:

A COMP (or INTEGER) field stores its value in binary format. The size of a COMP (or INTEGER) field depends on the number of digit positions specified in its PICTURE clause:

PIC Clause	Size of Field
9(1) to 9(4)	2 bytes
9(5) to 9(9)	4 bytes
9(10) to 9(18)	8 bytes

11.15.2 COMP-1 (or REAL) Fields

A COMP-1 (or REAL) field stores its value in single-precision real (floating point) format. COMP-1 fields are four bytes long.

11.15.3 COMP-2 (or DOUBLE) Fields

A COMP-2 (or DOUBLE) field stores its value in double-precision real (floating point) format. COMP-2 fields are eight bytes long.

11.15.4 COMP-3 (or PACKED) Fields

A COMP-3 (or PACKED) field stores its value in packed-decimal format. The value is stored two digits per byte. The value of a COMP-3 field must contain a sign. The sign occupies the four low-ordered bits in the rightmost byte. The size of the field depends on the number of digit positions specified by the field's PICTURE clause:

$$\text{size (in bytes)} = \frac{\text{digit-positions} + 1}{2}$$

For example, a field with three digit positions is two bytes long. If the field contains an even number of digits, the size is rounded up. Thus, a six-digit field is stored in four bytes.

11.15.5 COMP-5 (or ZONED) Fields

A COMP-5 (or ZONED) field stores its value in signed decimal format. A value in a COMP-5 field is stored one digit per byte. Therefore, the size of a COMP-5 field is the number of digit positions specified in its PICTURE clause.

The sign of a COMP-5 value shares the rightmost byte with lowest-valued digit of the value. The lowercase letters p through y represents a negative sign for the values 0 through 9.

11.15.6 COMP-6 Fields

A COMP-6 field stores its value in PDP-11 COBOL V3 COMPUTATIONAL format. The size of a COMP-6 field depends on the number of digit positions specified in its PICTURE clause:

PIC Clause	Size of Field
9(1) to 9(4)	2 bytes
9(5) to 9(9)	4 bytes
9(10) to 9(14)	6 bytes
9(15) to 9(18)	8 bytes

11.15.7 DATE Fields

A DATATRIEVE DATE field stores a date as an eight-byte binary value. Other languages will not interpret the date field correctly. The date is expressed as the number of clunks (100-nanosecond units) since the base date of 00:00:00 AM on November 17, 1858.

When you print a date field, DATATRIEVE translates the number of clunks to the format you specify in the EDIT-STRING clause (or to the default format if no EDIT-STRING clause is included in the field definition). The default format is DD-MMM-YY for dates in the 20th century, and DD-MMM-YYYY for all other dates.

When you enter a date value, DATATRIEVE translates the input value to clunks before storing it.

Examples

1. Define the field SALE-DATE as a date field, to be printed in the default format for date fields.

```
06 SALE-DATE USAGE IS DATE.
```

2. Define the field SALE-PRICE as a REAL (COMP-1) field.

```
05 SALE-PRICE PIC 9(5)
   USAGE REAL
   EDIT-STRING IS $(6).
```

11.16 VALID IF Clause

Function

Validates a value for the field before it is stored in the record.

Format

VALID IF bool-exp

Arguments

bool-exp
Is any Boolean expression.

Restrictions

A field definition cannot contain both a VALID IF and a COMPUTED BY clause.

Description

When a value is entered for the field with a MODIFY or STORE statement, DATATRIEVE evaluates the Boolean expression. If the Boolean expression is "True," DATATRIEVE stores the value in the field. If it is "False," DATATRIEVE prints an error message and reprompts for the field value.

Examples

1. Compare the value entered for the RIG field to the character strings SLOOP, KETCH, MS, and YAWL. Store the value in the field if it is one of those character strings.

```
06 RIG PIC X(6)
   VALID IF RIG EQ "SLOOP", "KETCH", "MS", "YAWL".
```

2. Store a value in the field LENGTH-OVER-ALL if it is between 15 and 50.

```
06 LENGTH-OVER-ALL PIC XXX
   VALID IF LOA BETWEEN 15 AND 50
   QUERY-NAME IS LOA.
```

3. Store a value in the PRICE field if it is greater than 1.3 times the displacement or if it is zero.

```
06 PRICE PIC 99999
   VALID IF PRICE>DISP*1.3 OR PRICE EQ 0
   EDIT-STRING IS $$$,$$$.
```

Chapter 12
Procedures and Indirect Command Files

A procedure is a fixed sequence of DATATRIEVE commands, statements, clauses, or arguments that you create, name, and store in the data dictionary. A procedure can contain command and statement sequences that you use frequently or that should be available to other users at an installation.

Once a procedure is defined in the data dictionary, you need only invoke the procedure to execute all the commands and statements it contains. Thus, a procedure can save a great deal of time because you need not reenter commonly used sequences of commands and statements. You avoid the typing errors that result from issuing DATATRIEVE commands and statements interactively. Furthermore, the complex command sequences in procedures are available to other users, and need not be reinvented each time they are needed.

An indirect command file can contain a fixed sequence of DATATRIEVE commands and statements, but it is stored in your directory, not in the data dictionary. ADT (Application Design Tool) creates an indirect command file containing the domain, record, and file definitions, and you can create similar files of your own. You can also put frequently used sequences of commands and statements in indirect command files. You can use these files to backup definitions of dictionary objects, to move dictionary objects from one dictionary to another, and to develop complex procedures. Although somewhat similar to procedures, indirect command files have some restrictions that procedures do not. Section 12.2 explains the uses and limitations of indirect command files.

12.1 Procedures

12.1.1 Creating a Procedure

To create a procedure, issue the DEFINE PROCEDURE command. (See Section 5.9 for more information on the DEFINE PROCEDURE command.)

```
DEFINE PROCEDURE  proc-name
```

DATATRIEVE then prompts with DFN> to indicate that it expects a procedure definition. Enter the commands, statements, clauses, or arguments that form the procedure definition. DATATRIEVE continues to prompt with DFN> until you enter the keyword END-PROCEDURE on a line by itself. DATATRIEVE does not check for syntax errors while you enter the procedure definition; the syntax is checked only when you invoke the procedure.

```
DTR> DEFINE PROCEDURE proc-name
DFN>           .
DFN>           .
DFN>           .
DFN> END-PROCEDURE
DTR>
```

As soon as you enter END-PROCEDURE, DATATRIEVE stores the procedure definition in the current data dictionary and creates a password table for the procedure.

12.1.2 Contents of a Procedure

A procedure can contain any number of the following DATATRIEVE elements:

- Full DATATRIEVE commands and statements
- Command and statement clauses and arguments

12.1.2.1 Commands and Statements — A procedure can contain all but five of the DATATRIEVE commands and statements listed in Chapter 5 or any statement listed in Chapter 7. If you enter a full command or statement in a procedure, you must follow the format for that command or statement as shown in Chapter 5 or 7. And, you must observe any restriction on the command or statement that is listed in its description.

You can include any DATATRIEVE command or statement in the procedure except the following:

- DEFINE DOMAIN
- DEFINE PROCEDURE
- DEFINE RECORD
- DEFINE TABLE
- EDIT

For example, the following command can be entered at the DATATRIEVE command level to establish a collection named BIGGIES. The collection contains records of all yachts with a length-over-all greater than 40 and is sorted by the builder name.

```
DTR> FIND BIGGIES IN YACHTS WITH LOA GT 40 SORTED BY BUILDER
[8 records found]
DTR>
```

The same full command can be entered in a procedure:

```
DTR> DEFINE PROCEDURE BIG-YACHTS
DFN> FIND BIGGIES IN YACHTS WITH LOA GT 40 SORTED BY BUILDER
DFN> END-PROCEDURE
DTR>
```

When you execute the procedure BIG–YACHTS, the results are the same as entering the FIND command at the command level:

```
DTR> :BIG-YACHTS
[8 records found]
DTR>
```

12.1.2.2 Clauses and Arguments — In addition to containing full commands and statements, a procedure can contain just a clause or argument from a command or statement. For example, instead of putting a full FIND command into a procedure, you can include just a record selection expression:

```
DTR> DEFINE PROCEDURE BIG-YACHTS-RSE
DFN> BIGGIES IN YACHTS WITH LOA GT 40 SORTED BY BUILDER
DFN> END-PROCEDURE
DTR>
```

Then, you can use the procedure name as the argument of a FIND command:

```
DTR> FIND :BIG-YACHTS-RSE
[8 records found]
DTR>
```

In fact, you can use this procedure in any command or statement containing an rse argument, such as the PRINT statement:

```
DTR> PRINT ALL :BIG-YACHTS-RSE
                                        LENGTH
                                         OVER
MANUFACTURER      MODEL          RIG     ALL      WEIGHT    BEAM     PRICE

 CHALLENGER       41             KETCH   41       26,700     13     $51,228
 COLUMBIA         41             SLOOP   41       20,700     11     $48,490
 GULFSTAR         41             KETCH   41       22,000     12     $41,350
 ISLANDER         FREEPORT       KETCH   41       22,000     13     $54,970
 NAUTOR           SWAN 41        SLOOP   41       17,750     12
 NEWPORT          41 S           SLOOP   41       18,000     11
 OLYMPIC          ADVENTURE      KETCH   42       24,250     13     $80,500
 PEARSON          419            KETCH   42       21,000     13
```

```
DTR> :BIG-YACHTS-QUERY
Enter MIN LOA: 39
```

12.1.2.3 Comments — You can include comments in a procedure as you create it, but comments are not stored in the data dictionary.

If you want to print comments on the terminal when you or another user executes your procedure, you can use the PRINT command, as shown in the following example:

```
DTR> DEFINE PROCEDURE YACHTS-REPORT
DFN> PRINT "THIS REPORT REQUIRES AN ESTABLISHED COLLECTION"
DFN> PRINT "SORTED BY BUILDER"
DFN>          .
DFN>          .
DFN>          .
```

The first two lines of this procedure print a message on your terminal when the procedure is executed.

```
DTR> :YACHTS-REPORT
THIS REPORT REQUIRES AN ESTABLISHED COLLECTION
SORTED BY BUILDER
```

12.1.3 Invoking a Procedure

As you have seen from the previous examples, you invoke a procedure by preceding its name with a colon (:).

```
:proc-name
```

To invoke a procedure, you must have E (execute) access to it based on your UIC/PPN.

The content of a procedure determines where you can invoke it. In general, you can invoke a procedure anywhere that you can use the commands, statements, clauses, or arguments contained in the procedure. For example, if the procedure contains only DATATRIEVE commands and statements, you can invoke it at the DATATRIEVE command level.

```
DTR> :proc-name
```

You cannot invoke a procedure in any of the following commands:

- ADT
- DEFINE DOMAIN
- DEFINE RECORD
- DEFINE TABLE
- EDIT
- SET GUIDE

12.1.4 Debugging and Editing a Procedure

When you invoke a procedure, DATATRIEVE executes each command or statement in the procedure as if it were entered directly at DATATRIEVE

command level. Some errors may occur during execution of the procedure. A typing error, for instance, can result in a syntax error in an otherwise correctly formatted command. If an error occurs during execution, DATATRIEVE prints an error message and terminates the procedure. You can correct the error by using the DATATRIEVE editor. Invoke the editor with the following command:

```
DTR> EDIT proc-name
QED>
```

When you find the error, use the appropriate editor commands to correct it. (See Chapter 8 for more information about the DATATRIEVE editor.)

12.1.5 Nesting Procedures

You can nest procedures by invoking a procedure within another procedure. However, do not allow a procedure to invoke itself, either directly or indirectly; you can create an infinite loop.

The following procedure calculates the price per pound of a yacht and establishes a format for printing:

```
DTR> DEFINE PROCEDURE PRICE-PER-POUND
DFN> PRICE/DISPLACEMENT ("PRICE"/"PER"/"POUND") -
DFN>    USING ZZ9.99
DFN> END-PROCEDURE
DTR>
```

The PRICE-PER-POUND procedure can then be invoked in another procedure that prints the builder, model, and price per pound of all yachts in the CURRENT collection.

```
DTR> DEFINE PROCEDURE PRICE-REPORT
DFN> PRINT ALL BUILDER, MODEL, :PRICE-PER-POUND
DFN> END-PROCEDURE
DTR>
```

When the procedure PRICE-REPORT is executed, it prints the three fields on the terminal. The following example uses the BIG-YACHTS procedure to establish the CURRENT collection and PRICE-REPORT to print a short report.

```
DTR> :BIG-YACHTS;:PRICE-REPORT
[8 records found]

                          PRICE
                           PER
MANUFACTURER    MODEL     POUND

 CHALLENGER     41        $1.91
 COLUMBIA       41        $2.34
 GULFSTAR       41        $1.87
 ISLANDER       FREEPORT  $2.49
 NAUTOR         SWAN 41    $.00
 NEWPORT        41 S       $.00
 OLYMPIC        ADVENTURE $3.31
 PEARSON        419        $.00
```

The maximum depth of procedure nesting depends on such factors as the amount of DATATRIEVE workspace available, the number and size of established collections, the size of all procedures invoked, and the number and type of readied domains.

12.1.6 Using a Procedure in a Loop

You can invoke a procedure in a REPEAT statement to execute it a number of times or in a FOR statement to apply it to a collection of records. You must, however, use care when invoking a procedure in these statements. For example, the following statement is syntactically correct, but will produce results you may not expect:

```
DTR> REPEAT 5 :proc-name
```

This statement does not execute the procedure five times; it executes the *first* command or statement in the procedure five times, then executes the remainder of the procedure once. When DATATRIEVE encounters the first complete statement in the procedure, it assumes that the REPEAT statement is also complete. Therefore, it executes the first command or statement in the procedure five times. DATATRIEVE then executes the remaining commands or statements in the procedure.

To repeat the entire procedure, enclose the procedure call or the procedure definition in a BEGIN–END block. For example, the following sequence of statements puts a procedure call in a BEGIN–END block and repeats the procedure five times:

```
DTR> REPEAT 5 BEGIN
DTR> :proc-name
DTR> END
```

The following example includes a BEGIN–END block in a procedure definition and invokes the procedure in a REPEAT statement.

```
DTR> DEFINE PROCEDURE proc-name
DFN> BEGIN
DFN>    .
DFN>    .
DFN>    .
DFN> END
DFN> END-PROCEDURE
DTR> REPEAT 5 :proc-name
```

If you invoke a procedure in a FOR statement, you must use the same technique: enclose the call or the procedure definition in a BEGIN–END block. For instance:

```
DTR> FOR rse BEGIN
DTR> :proc-name
DTR> END
```

Also, remember that if you use a procedure in this way, it cannot include a FIND or SELECT statement; FIND and SELECT cannot be used in BEGIN-END blocks.

12.1.7 Aborting Procedures

You can abort the execution of a procedure by including an ABORT statement (Section 5.1) in the procedure definition. If the abort conditions are met and SET ABORT is in effect, DATATRIEVE aborts the procedure and prints a message on your terminal. If SET NO ABORT is in effect, DATATRIEVE aborts the command or statement which contains the ABORT, but continues to execute the other commands and statements in the procedure.

You can insure that SET ABORT is in effect by including that statement in the procedure definition. For example:

```
DTR> PROCEDURE BIG-YACHTS-QUERY
DFN> SET ABORT
DFN> DECLARE LENGTH PIC 99
DFN>   VALID IF LENGTH GT 35,
DFN> LENGTH = *,"MIN LOA"
DFN> IF LENGTH GT 42 THEN ABORT "NO BOATS THAT BIG"
DFN> FIND BIGGIES IN YACHTS WITH LOA GE LENGTH SORTED BY BUILDER
DFN> PRINT BUILDER, RIG, LOA, PRICE OF BIGGIES
DFN> END-PROCEDURE
```

If you invoke BIG-YACHTS-QUERY and supply a LENGTH of 35 or smaller, DATATRIEVE reprompts you for a valid LENGTH. If you supply a LENGTH greater than 42, the procedure aborts, prints the specified abort message, and returns you to the DATATRIEVE command level:

```
DTR> :BIG-YACHTS-QUERY
Enter MIN LOA: 35
Validation error for LENGTH
Re-enter MIN LOA: 43
ABORT: NO BOATS THAT BIG
Execution terminated by "ABORT" statement
```

If you assign a value between 36 and 42 to LENGTH, the appropriate collection is printed:

```
                          [(passwd)]
DTR> EDIT proc-name       [       ]
                          [(*)    ]
QED> WH                   [       ]

                            LENGTH
                             OVER
MANUFACTURER    RIG          ALL      PRICE

 BLOCK I.       SLOOP        39
 CHALLENGER     KETCH        41       $51,228
 COLUMBIA       SLOOP        41       $48,490
 GULFSTAR       KETCH        41       $41,350
 ISLANDER       KETCH        41       $54,970
 LINDSEY        MS           39       $35,900
```

(continued on next page)

```
NAUTOR          SLOOP       41
NEWPORT         SLOOP       41
OLYMPIC         KETCH       42      $80,500
PEARSON         SLOOP       39
PEARSON         KETCH       42
```

12.1.8 Maintaining Procedures

12.1.8.1 Displaying Procedure Names Stored in the Data Dictionary — You can list the names of all procedures in the current data dictionary with the SHOW command:

```
DTR> SHOW PROCEDURES
```

12.1.8.2 Displaying Procedures — If you want to print a procedure on your terminal, you can use the SHOW command and specify the name of the procedure to be displayed. You must have R (read) access privilege to the procedure:

```
DTR> SHOW proc-name [ (passwd) ]
                    [ (*)      ]
```

You can also use the DATATRIEVE editor to print a copy of the procedure on your terminal. In this case, you must have C (control) access privilege to the procedure:

```
DTR> EDIT proc-name [ (passwd) ]
                    [ (*)      ]
QED> WH
```

12.1.8.3 Deleting Procedures — You can delete a procedure from the current data dictionary with the DELETE command. You must have C (control) access privilege to the procedure.

```
DTR> DELETE proc-name [ (passwd) ] ;
                      [ (*)      ]
```

12.1.9 Protecting Procedures

When you create a procedure, DATATRIEVE stores the procedure definition in the current data dictionary and creates a password table for the procedure. DATATRIEVE automatically stores one UIC/PPN in the password table with full access privileges [R (read), W (write), E (execute), M (modify), and C (control) access]. The actual UIC/PPN that is stored in the password table is installation-dependent.

If you create a procedure, you have full access privileges to it. If you make no explicit restrictions, the access privileges of other users are governed by the defaults established at installation.

You can modify the password table to give various kinds of access privilege to other users. (See Chapter 14 for information on modifying password tables.) However, you cannot establish a password for E (execute) access to a procedure; E (execute) access to a procedure is determined solely by the UIC/PPN with which a user logs in.

To guard against accidental deletion of your procedure (especially if it is a long one), you should maintain a backup copy of it. You can use the DATATRIEVE EXTRACT command (described in Section 5.18) to copy your procedure to an indirect command file for backup.

12.1.10 A Sample Procedure

Figure 12–1 shows the creation of a procedure that uses the Report Writer to write a summary report of yacht data. The example illustrates not only the process of creating a procedure but also some statements and elements that are useful in procedures.

The PRINT statement is used for displaying a message to the user who invokes the procedure.

The prompting value expression *."YES OR NO" requires the user to respond to the question: HAVE YOU ESTABLISHED A COLLECTION? The Boolean expression CONTAINING checks the user's response to the question. If the response contains a letter N anywhere, the procedure aborts.

The prompting value expression *."OUTPUT DEVICE OR FILE" allows the user to select the device or file to contain the report when the procedure is executed.

Figure 12–1: Sample Procedure

```
DTR> DEFINE PROCEDURE YACHT-SUMMARY
DFN> SET ABORT
DFN> PRINT "THIS REPORT REQUIRES AN ESTABLISHED COLLECTION,"
DFN> PRINT "SORTED BY LOA AND BEAM."
DFN> PRINT "HAVE YOU ESTABLISHED A COLLECTION?"
DFN> IF *."YES OR NO" CONTAINING "N" THEN ABORT "SORRY, NO COLLECTION."
DFN> REPORT ON *."OUTPUT DEVICE OR FILE"
DFN> SET REPORT-NAME="EXAMPLE: REPORT FROM A PROCEDURE"
DFN> SET LINES-PAGE=55, COLUMNS-PAGE=60
DFN> PRINT BUILDER, MODEL, LOA, BEAM, PRICE
DFN> AT BOTTOM OF LOA PRINT SKIP, "AVERAGE PRICE =",
DFN>   AVERAGE (PRICE), SKIP
DFN> AT BOTTOM OF REPORT PRINT COL 13,"NUMBER OF BOATS = ",
DFN>   COL 33, COUNT, SKIP, "AVERAGE PRICE OF ALL BOATS =", AVERAGE (PRICE)
DFN> END-REPORT
DFN> END-PROCEDURE
DTR>
```

If you invoke YACHT-SUMMARY and answer NO to the first prompt, the procedure aborts:

```
DTR> :YACHT-SUMMARY
THIS REPORT REQUIRES AN ESTABLISHED COLLECTION,

SORTED BY LOA AND BEAM.
```

```
HAVE YOU ESTABLISHED A COLLECTION?

Enter YES OR NO: NO
ABORT: SORRY, NO COLLECTION.
Execution terminated by "ABORT" statement
```

If you answer YES to the first prompt, but, in fact, you do not have a current collection, the Report Writer aborts the procedure and prints an error message:

```
DTR> :YACHT-SUMMARY
THIS REPORT REQUIRES AN ESTABLISHED COLLECTION,

SORTED BY LOA AND BEAM.

HAVE YOU ESTABLISHED A COLLECTION?

Enter YES OR NO: YES
A current collection has not been established
```

If you make a collection of YACHTS with LOA between 36 and 37 and price not equal to zero, the following report results:

```
DTR> READY YACHTS
DTR> FIND YACHTS WITH LOA BETWEEN 36 37 AND PRICE NE 0
[5 records found]
DTR> SORT CURRENT BY LOA, BEAM
DTR> :YACHT-SUMMARY
THIS REPORT REQUIRES AN ESTABLISHED COLLECTION,

SORTED BY LOA AND BEAM.

HAVE YOU ESTABLISHED A COLLECTION?

Enter YES OR NO: YES
Enter OUTPUT DEVICE OR FILE: TI:

             EXAMPLE: REPORT FROM A PROCEDURE     DD-MMM-YY
                                                  Page 1

                                        LENGTH
                                         OVER
        MANUFACTURER            MODEL    ALL    BEAM   PRICE

          ISLANDER              36         36      11   $31,730
          I. TRADER             37         36      12   $39,500

       AVERAGE PRICE =                                 $35,615

          IRWIN                 37 MARK II 37      11   $36,950
          NORTHERN              37         37      11   $50,000
          ALBERG                37 MK II   37      12   $36,951

       AVERAGE PRICE =                                 $41,300

            NUMBER OF BOATS =       5
AVERAGE PRICE OF ALL BOATS =                           $39,026
```

12.2 Indirect Command Files

12.2.1 Creating an Indirect Command File

When you use ADT (Application Design Tool), it creates an indirect command file for you, and that file contains domain, record, and file definitions.

You can also create an indirect command file with a text editor. Invoke a text editor, give your file a name and file type, and enter the sequence of DATATRIEVE commands and statements just as you would in DATATRIEVE. Do not include the prompts (DTR>, DFN>, QED>, or RW>). When you complete the sequence of commands and statements and exit from the editor, the indirect command file is stored in your directory.

You can save time by giving your indirect command file a file type of .CMD; when you invoke an indirect command file and omit the file type, DATATRIEVE supplies .CMD as the default file type.

12.2.2 Contents of an Indirect Command File

An indirect command file can contain all DATATRIEVE commands and statements, including the five not allowed in procedures (DEFINE DOMAIN, DEFINE RECORD, DEFINE PROCEDURE, DEFINE TABLE and EDIT).

12.2.2.1 ADT, EDIT, and SET GUIDE — If you put an ADT, EDIT, or SET GUIDE command in the file, you cannot specify any responses to the prompts that follow the entry of those commands in DATATRIEVE. If you put any of these three commands into an indirect command file, DATATRIEVE puts you into the requested mode and waits for you to respond to the prompt.

When you then exit from the editor, GUIDE mode, or ADT, DATATRIEVE executes the next line in the indirect command file. If that line is a valid response to the prompts of the editor, GUIDE mode, or ADT, but not a valid DATATRIEVE command or statement, the line is not executed. DATATRIEVE then prints an error message on your terminal and returns you to the DATATRIEVE command level.

12.2.2.2 Incomplete Commands and Statements — You can begin or end an indirect command file with incomplete DATATRIEVE commands and statements, but there are limits you must observe:

- Complete each command element or statement element (for example, neither TWEEN for BETWEEN nor REL for RELEASE is permitted).
- If your indirect command file ends with an incomplete DATATRIEVE command or statement, end the fragment at a point where DATATRIEVE accepts a carriage return without terminating the execution of the command or statement.

When typed alone on a line, ADT, EXIT, FINISH, HELP, PRINT, REPORT, and SELECT are complete in themselves. ERASE and MODIFY must have contexts in which to work, and EXTRACT interprets a carriage return as an incorrect file specification. When typed alone on a line all the other commands and statements either prompt you for the missing elements and return a DTR> prompt, or simply return the DTR> prompt and wait for you to supply the missing elements.

- If your indirect command file begins with an incomplete DATATRIEVE command or statement, use a fragment that completes the command or statement for which DATATRIEVE is expecting an end. For example:

```
DTR> READY YACHTS
DTR> FIND YACHTS WITH LOA = 42
[2 records found]
DTR> PRINT BUILDER, RIG,
[Looking for next element in list]
DTR> @PRT
LOA, PRICE OF CURRENT

                    LENGTH
                     OVER
MANUFACTURER  RIG    ALL     PRICE

 OLYMPIC      KETCH   42    $80,500
 PEARSON      KETCH   42
```

- Do not put incomplete commands or statements in the middle of an indirect command file. DATATRIEVE looks at the next entry in the indirect command file for the completion of the incomplete command or statement. When it fails to find one, DATATRIEVE prints an error message and does not execute subsequent valid commands or statements in the indirect command file. For example, the indirect command file PRT.CMD contains the following DATATRIEVE commands and statements:

```
READY YACHTS
FIND FIRST 5 YACHTS
PRINT BUILDER, RIG,
READY FAMILIES
FIND FIRST 5 FAMILIES
PRINT CURRENT
```

The PRINT statement is incomplete. When DATATRIEVE does not find a proper conclusion in the next line, it prints an error message and does not execute any of the subsequent commands or statements in the indirect command file:

```
DTR> @PRT
READY YACHTS
FIND FIRST 5 YACHTS
[5 records found]
PRINT BUILDER, RIG,
READY FAMILIES
Expected end of statement, encountered "FAMILIES"
DTR>
```

12.2.2.3 Comments — If you put comments into an indirect command file, they are printed on your terminal when you invoke the file. This use of comments is an advantage indirect command files have over procedures.

If your indirect command file defines a procedure and you put comments into the procedure definition, the comments are not stored in the data dictionary and are not printed when you invoke the procedure. Those comments are printed, however, each time you invoke the indirect command file. If you want comments to be printed when you invoke the procedure, put the comments in PRINT statements, as suggested in Section 12.2.3.

12.2.3 Invoking an Indirect Command File

To invoke an indirect command file, precede the file specification with an at sign (@). When you invoke an indirect command file, it must be entered on a line by itself (except as noted below):

```
DTR> @ddn:[200,200] PRT.CMD
```

If the file type is .CMD, and the file is in your default directory, you need only put the file name:

```
DTR> @PRT.CMD
```

or

```
DTR> @PRT
```

In only one situation can you invoke an indirect command file and have another command on the same line. This special case only arises when you first call DATATRIEVE. In response to the system level prompt, you type:

```
DTR @file-spec
```

Invoking an indirect command file in this special way differs from invoking one in response to the DTR> prompt. All the commands and statements in the indirect command file are executed as though you had entered them interactively. However, no DATATRIEVE identification message is printed on your terminal, and after the execution of the last command or statement in the file, DATATRIEVE adds an EXIT command, and you automatically exit from DATATRIEVE. You can apply this feature to routines you use regularly, and to routines deferred for batch processing.

Similarly, you can include a string of DATATRIEVE commands, statements, and procedures on the same line in which you invoke DATATRIEVE. The length of the whole line cannot exceed eighty (80) characters, and the commands, statements, and procedures must be separated by semicolons:

```
DTR READY YACHTS;FIND YACHTS WITH RIG="MS";PRINT BOAT
```

After executing the commands and statements, DATATRIEVE returns you to the operating system command level.

If the indirect command file is in another user's directory, you invoke it by specifying all the necessary information in the following format:

dev:[ufd]filename.ext;ver

where:

dev:	=	device name
[ufd]	=	user file directory
filename	=	file name
.ext	=	file type
;ver	=	version number

To invoke an indirect command file in another user's directory, you must have R (read) access to that directory.

You cannot, however, invoke an indirect command file from a procedure. When you try to enter the at sign and file specification in response to the DFN> prompt, DATATRIEVE enters the statements of the indirect command file as though you had typed them yourself. The resulting procedure contains the same sequence of statements as the indirect command file, and succeeds or fails depending on the logic of that sequence.

You can invoke an indirect command file in response to the RW> prompt of the Report Writer. The file must begin with valid report statements; if you complete the report specification in the file with an END-REPORT statement, you can follow the report specification with other valid DATATRIEVE commands or statements.

You cannot invoke indirect command files while you are in ADT, GUIDE mode, or the editor.

12.2.4 Debugging and Editing an Indirect Command File

When you invoke an indirect command file, DATATRIEVE prints each command or statement on your terminal and executes it as if you had entered it directly from your keyboard. If an error occurs, DATATRIEVE prints an error message and terminates the execution of the indirect command file. You can correct the error by exiting from DATATRIEVE and using a text editor to make the needed changes.

You cannot use the DATATRIEVE editor to change an indirect command file.

When you correct the error, you must return to DATATRIEVE, ready the necessary domains, establish the appropriate collection, and execute the indirect command file again. These extra steps required for editing an indirect command file are a disadvantage compared to the steps needed to edit a DATATRIEVE procedure.

12.2.5 Nesting Indirect Command Files

You can invoke both procedures and indirect command files from within an indirect command file. Do not allow an indirect command file to invoke itself, either directly or indirectly; you can create an infinite loop.

12.2.6 Using an Indirect Command File in a Loop

Indirect command files act like procedures when used in loops. See Section 12.1.6 for the use of the REPEAT, BEGIN-END, and FOR statements to form loops containing indirect command files.

12.2.7 Aborting Indirect Command Files

You can abort the execution of an indirect command file by including an ABORT statement in the file. If the abort conditions are met and SET ABORT is in effect, DATATRIEVE aborts the indirect command file and prints a message on your terminal. If SET NO ABORT is in effect, DATATRIEVE aborts the command or statement which contains the ABORT, but continues to execute the other commands and statements in the file.

12.2.8 Maintaining Indirect Command Files

12.2.8.1 Displaying the Names of Indirect Command Files — The names of indirect command files are stored in your directory, not in the data dictionary as the names of procedures are. If you adopt the convention of using .CMD as the file type for indirect command files, you can list your indirect command files by requesting a directory listing of *.CMD at the system level. You can adopt any convention you wish and use the wild card in the same manner.

12.2.8.2 Displaying Indirect Command Files — You can display the contents of an indirect command file with the system level command that prints a file on your terminal.

12.2.8.3 Deleting Indirect Command Files — You can delete an indirect command file from your directory the same way you delete any other file.

12.2.9 Protecting Indirect Command Files

Indirect command files have less protection from unauthorized use than DATATRIEVE procedures do. Each procedure has its own password table that DATATRIEVE checks when the procedure is invoked. Indirect command files, on the other hand, have only the operating system's protection to prevent other users from reading and then executing your indirect command files.

DATATRIEVE accepts input from an indirect command file as though it were entered from a terminal. As a result, DATATRIEVE itself does not provide any protection against unauthorized use of indirect command files. To prevent other people from using your indirect command files, you must change the system level protection to deny R (read) access to them.

By using indirect command files you can protect your DATATRIEVE operations from accidental corruption of the data dictionary. In fact, your backup files of domain, record, file, and procedure definitions can be used as indirect command files if the data dictionary were corrupted and you needed to restore the definitions it had previously contained.

12.2.10 Sample Indirect Command File

Figure 12–2 shows an indirect command file similar to the procedure in Section 12.1.10. This example illustrates both similarities and differences between procedures and indirect command files. The indirect command file here is YSUM.CMD and is invoked with the at sign and without the file type.

Instead of using the PRINT statement for displaying comments, the comments are preceded by exclamation points.

In contrast to the sample procedure, each statement and command in the indirect command file is printed as DATATRIEVE executes it.

When DATATRIEVE encounters the statement with the *."YES OR NO" prompting value expression, it pauses in the execution of the indirect command file to wait for the user's response to the question: HAVE YOU ESTABLISHED A COLLECTION? As in the previous example, the Boolean expression, CONTAINING, checks the user's response to the question, and if the response contains a letter N anywhere, the procedure aborts.

When DATATRIEVE encounters the *."OUTPUT DEVICE OR FILE" prompting value expression, it pauses again to wait for the user to select the device or file for output of the report.

Note that, except for the report name, the report produced by the indirect command file is the same as the one produced by the procedure YACHT-SUMMARY.

Figure 12–2: Sample Indirect Command File

The file YSUM.CMD contains the following sequence of command and statements:

```
SET ABORT
!THIS REPORT REQUIRES AN ESTABLISHED COLLECTION,
!SORTED BY LOA AND BEAM.
!
!HAVE YOU ESTABLISHED A COLLECTION?
IF *."YES OR NO" CONTAINING "N" THEN ABORT "SORRY, NO COLLECTION."
REPORT ON *."OUTPUT DEVICE OR FILE"
```

(continued on next page)

```
SET REPORT-NAME="SAMPLE REPORT"/"FROM AN INDIRECT COMMAND FILE"
SET LINES-PAGE=55, COLUMNS-PAGE=60
PRINT BUILDER, MODEL, LOA, BEAM, PRICE
AT BOTTOM OF LOA PRINT SKIP, "AVERAGE PRICE =",
  AVERAGE (PRICE), SKIP
AT BOTTOM OF REPORT PRINT COL 13,"NUMBER OF BOATS = ",
  COL 33, COUNT, SKIP, "AVERAGE PRICE OF ALL BOATS =", AVERAGE (PRICE)
END-REPORT
```

When the domain is readied and the appropriate collection established, the indirect command file is invoked with an at sign and without the .CMD file type. DATATRIEVE prints each command and statement as it executes it:

```
DTR> READY YACHTS
DTR> FIND FIRST 5 YACHTS WITH LOA BETWEEN 36 37 AND PRICE NE 0
[5 records found]
DTR> SORT BY LOA, BEAM
DTR> @YSUM
SET ABORT
!THIS REPORT REQUIRES AN ESTABLISHED COLLECTION,
!SORTED BY LOA AND BEAM.
!
!HAVE YOU ESTABLISHED A COLLECTION?
IF *,"YES OR NO" CONTAINING "N" THEN ABORT "SORRY, NO COLLECTION."
Enter YES OR NO: YES
REPORT ON *,"OUTPUT DEVICE OR FILE"
SET REPORT-NAME="SAMPLE REPORT"/"FROM AN INDIRECT COMMAND FILE"
SET LINES-PAGE=55, COLUMNS-PAGE=60
PRINT BUILDER, MODEL, LOA, BEAM, PRICE
AT BOTTOM OF LOA PRINT SKIP, "AVERAGE PRICE =",
  AVERAGE (PRICE), SKIP
AT BOTTOM OF REPORT PRINT COL 13,"NUMBER OF BOATS = ",
  COL 33, COUNT, SKIP, "AVERAGE PRICE OF ALL BOATS =", AVERAGE (PRICE)
END-REPORT
Enter OUTPUT DEVICE OR FILE: TI:

                    SAMPLE REPORT                DD-MMM-YY
              FROM AN INDIRECT COMMAND FILE      Page 1

                                          LENGTH
                                           OVER
        MANUFACTURER          MODEL        ALL    BEAM  PRICE

          ISLANDER           36             36      11  $31,730
          I. TRADER          37             36      12  $39,500

       AVERAGE PRICE =                                  $35,615

          IRWIN              37 MARK II     37      11  $36,950
          NORTHERN           37             37      11  $50,000
          ALBERG             37 MK II       37      12  $36,951

       AVERAGE PRICE =                                  $41,300

            NUMBER OF BOATS =       5
 AVERAGE PRICE OF ALL BOATS =                         $39,026
```

Chapter 13
Description Tables

A description table is a set of code-and-description pairs that you create and store under a description table name in the data dictionary.

By using a description table, you can enter a single value in a command or statement and cause DATATRIEVE to look for that value in the table. If the value matches one of the codes in the table, DATATRIEVE can give you two responses:

1. Tell you that the value equals a code in the table.
2. Use the description associated with the matching code.

If the value does not match one of the codes in the table, DATATRIEVE can give you two other responses:

1. Tell you that the value does not equal any code in the table.
2. Use a default description, which you define when creating the table.

Using a description table, you can save a great deal of record space because you need store only the codes in a field. For example, you can store a one character code in a field and print a multiword description as the field's value. Short codes, for which only you need know the cipher, can be translated into text meaningful to other people.

You can also associate stable information with other information that changes frequently. For instance, rather than store prices in several data files, you might create a description table with part or model numbers as codes and prices as the associated descriptions. To print a current price, you need only cite the model number and refer to the description table. Rather than modify all your data files, you need only update the description table to reflect price changes.

Description tables can also save the time involved in creating and entering complex DATATRIEVE command and statement sequences.

For example, you want to print one of seven character strings, depending on the value of a value expression. On one hand, you can use a series of IF-THEN-ELSE and PRINT statements to print the appropriate character string. On the other hand, you might put each value and its corresponding character string in a description table.

The following procedure prints the name of the state corresponding to one of seven telephone area codes. The procedure prompts for an area code, then prints the state with that area code. If the user enters an area code that is not in the New England region, the procedure prints an error message.

```
PROCEDURE AREA-CODES
DECLARE AREA-CODE PIC 999,
AREA-CODE = *,"AREA CODE"
IF AREA-CODE EQ 207 THEN PRINT "MAINE" ELSE
IF AREA CODE EQ 603 THEN PRINT "NEW HAMPSHIRE" ELSE
IF AREA CODE EQ 802 THEN PRINT "VERMONT" ELSE
IF AREA-CODE EQ 617 THEN PRINT "EASTERN MASSACHUSETTS" ELSE
IF AREA-CODE EQ 413 THEN PRINT "WESTERN MASSACHUSETTS" ELSE
IF AREA-CODE EQ 401 THEN PRINT "RHODE ISLAND" ELSE
IF AREA-CODE EQ 203 THEN PRINT "CONNECTICUT" ELSE
PRINT "NOT A VALID AREA CODE"
END-PROCEDURE
```

On the other hand, you can enter the area codes and corresponding state names in a description table. Then, you can refer to the description table in a PRINT statement. The description table for these seven area codes contains the following entries:

```
TABLE AREA-CODE-TABLE
207 : MAINE,
603 : "NEW HAMPSHIRE",
802 : VERMONT,
617 : "EASTERN MASSACHUSETTS",
413 : "WESTERN MASSACHUSETTS",
401 : "RHODE ISLAND",
203 : CONNECTICUT,
ELSE "NOT A VALID AREA CODE"
END-TABLE
```

You can now prompt for an area code and print the corresponding state in a single statement.

```
DTR> PRINT *,"AREA CODE" VIA AREA-CODE-TABLE USING X(21)
Enter AREA CODE: 203
CONNECTICUT
```

Because description tables can be modified using the DATATRIEVE editor, this description table could be expanded to include other area codes. For large numbers of code-and-description pairs, you can save time by creating a long description table rather than a lengthy procedure. Furthermore, because description table references are contained in Boolean and value expressions, the tables you create can be used by many DATATRIEVE commands and statements.

13.1 Creating a Description Table

To create a description table, issue the DEFINE TABLE command. (Section 5.11 contains more information on the DEFINE TABLE command.)

```
DEFINE TABLE desc-table-name
```

DATATRIEVE then prompts with DFN> and waits for your description table definition. Enter the code-and-description pairs, in the following format:

```
{"code"}     {"description"}
{ code  }  :  { description } [,]
```

The code and description must be separated by a colon (:); one or more spaces before and after the colon are optional.

Each code-and-description pair must end with a comma (,), unless it is the last entry in the description table. If an ELSE clause is the last line of the table, then all code-and-description pairs, including the last one, must end with a comma (,). Do not put a comma after the ELSE clause.

If the code or the description conforms to the rules for DATATRIEVE names (see Section 4.4), you do not have to enclose it in quotation marks. However, remember that DATATRIEVE converts any lowercase letters to uppercase.

If the code or the description does not conform to the rules for DATATRIEVE names, or if you want to preserve lowercase letters in it, you must enclose it in quotation marks and the rules for character string literals apply; that is, neither the code nor the description can exceed 250 characters, and neither can contain a carriage return, line feed, or control character.

After the last code-and-description pair, you can enter an optional ELSE clause:

```
ELSE {"description"}
     { description }
```

If an ELSE clause ends your description table, DATATRIEVE substitutes the description in the ELSE clause for any values not found in the table. If your description table does not end with an ELSE clause, DATATRIEVE prints an error message on your terminal when a value is not found in the table.

The ELSE clause has no effect when using a description table to match values with codes in the table.

The rules for a description in an ELSE clause are the same as for a description in a code-and-description pair: the description either must conform to the rules for DATATRIEVE names or must be enclosed in quotation marks and conform to the rules for character string literals.

To end a description table definition, enter the keyword END–TABLE on a line by itself. The END–TABLE statement must follow the ELSE clause, if specified, or the last code-and-description pair if there is no ELSE clause.

After you enter the END–TABLE statement, DATATRIEVE stores the definition in the current data dictionary and creates a password table for the description table.

Figure 13–1 summarizes the process for creating a description table.

Figure 13–1: Summary of DEFINE TABLE

```
DTR> DEFINE TABLE desc-table-name
DFN> code-1 : description-1,
                .
                .
                .
DFN> [ELSE description-n]
DFN> END-TABLE
DTR>
```

Figure 13–2 shows the creation of the description table RIG–TABLE, which is stored in the sample data dictionary for YACHTS.

Figure 13–2: Creating RIG–TABLE

```
DTR> DEFINE TABLE RIG-TABLE
DFN> "SLOOP" : "ONE-MAST",
DFN> "KETCH" : "TWO MASTS, BIG ONE IN FRONT",
DFN> "YAWL"  : "SIMILAR TO KETCH",
DFN> "M/S"   : "SAILS AND BIG MOTOR",
DFN> ELSE "SOMETHING ELSE"
DFN> END-TABLE
DTR>
```

As you create a description table, DATATRIEVE checks for syntax errors. For example, if you forget to end a description with a comma, DATATRIEVE prints an error message and aborts the DEFINE TABLE command. If a syntax error occurs, all code-and-description pairs are lost and no description table is created. To complete the definition of the table, you must begin with the DEFINE TABLE command and reenter the entire description table definition.

To avoid this duplication of effort, you can use the DATATRIEVE editor to create the largest part of a long description table. First, issue the DEFINE TABLE command, enter one or two code-and-description pairs, and end the description table with END–TABLE. Then call the DATATRIEVE editor and use the insert mode to continue entering code-and-description pairs.

Remember that while you are using the editor, DATATRIEVE does not check for syntax errors in the table, such as missing quotation marks or missing commas. You should, therefore, use care when entering the rest of the description table. If the table does not work as expected because of an error in its definition, you can use the DATATRIEVE editor again to correct the mistake.

13.2 Using a Description Table

References to description tables are contained in either value expressions or Boolean expressions, and you can use those expressions wherever the formats of DATATRIEVE commands and statements allow them.

13.2.1 Using IN with a Description Table

You can use description tables with the relational operators IN and NOT IN to make collections, to set conditions in IF–THEN–ELSE statements, and to validate data. Value expressions that use IN or NOT IN and refer to description tables compare the value of a specified field in a record only with the codes in the table. The following is the general format for using IN and NOT IN with a description table:

```
{field-name    }
{*.prompt      }  [NOT]IN   desc-table-name
{variable-name }
```

1. Making collections.
   In a record selection expression that establishes a collection or a record stream, you can use a Boolean expression containing WITH and a reference to a description table:

   ```
   DTR> FOR YACHTS WITH RIG IN RIG-TABLE PRINT BOAT
   ```

   or

   ```
   DTR> FIND YACHTS WITH RIG NOT IN RIG-TABLE
   [5 records found]
   ```

2. Setting conditions in IF–THEN–ELSE statements.
   You can combine IN with a description table reference to set the conditions of an IF–THEN–ELSE statement:

   ```
   DTR> FOR FIRST 7 YACHTS
   DTR> IF RIG NOT IN RIG-TABLE THEN PRINT BOAT

                                LENGTH
                                 OVER
   MANUFACTURER    MODEL  RIG   ALL  WEIGHT  BEAM  PRICE

    AMERICAN      26-MS   MS     26    5,500   08  $18,895

   DTR>
   ```

Remember that you must establish a context for the IF–THEN–ELSE statement by preceding it with a SELECT statement or by nesting it within a FOR statement.

3. Validating data.
   By referring to a description table in a VALID IF clause, you can validate data in a field before storing the data in a record. The VALID IF clause must be part of the record definition. The following definition of PHONE–REC illustrates this method of automatic data validation:

```
DTR> DEFINE RECORD PHONE-REC USING
DFN> 01 PHONE.
DFN>     02 NAME PIC X(20).
DFN>     02 NUMBER PIC 9(7) EDIT-STRING IS XXX-XXXX.
DFN>     02 LOCATION PIC X(9).
DFN>     02 DEPARTMENT PIC XX VALID IF
DFN>             DEPARTMENT IN DEPT-TABLE.
DFN> ;
DTR>
```

13.2.2 Using VIA with a Description Table

You can refer to description tables in value expressions by combining the table name with VIA. These references cause DATATRIEVE to compare the codes in the table with the value you supply, and if the value matches one of the codes, DATATRIEVE uses the corresponding description in the table as you have directed in the command or statement. Use the following format to refer to a description table in a value expression with VIA:

```
{field-name    }
{*.prompt      }   VIA desc-table-name
{variable-name }
```

You can refer to an entry in RIG–TABLE by using a prompting value expression,as shown in the following PRINT statement:

```
DTR> PRINT *.TYPE-OF-BOAT VIA RIG-TABLE USING X(30)
Enter TYPE-OF-BOAT: KETCH
TWO MASTS, BIG ONE IN FRONT
```

If you refer to a description table in a PRINT statement, include a USING clause to specify the number of characters to be printed. If you omit the USING clause, DATATRIEVE prints only the first ten characters of the description.

You can also use a field name as the value expression, as shown in this example:

```
DTR> FOR FIRST 7 YACHTS
DTR> PRINT MODEL, RIG, RIG VIA RIG-TABLE USING X(30)
```

(continued on next page)

```
  MODEL      RIG                  RIG

37 MK II     KETCH   TWO MASTS, BIG ONE IN FRONT
79           SLOOP   ONE MAST
VEGA         SLOOP   ONE MAST
BALLAD       SLOOP   ONE MAST
26           SLOOP   ONE MAST
26-MS        MS      SOMETHING ELSE
30/32        SLOOP   ONE MAST
```

The USING X(30) clause indicates that the third column of data will be 30 characters wide.

13.2.3 Description Tables and DATATRIEVE Workspace

When you first refer to a description table, DATATRIEVE searches the current data dictionary for the table and loads it into the DATATRIEVE workspace. DATATRIEVE then evaluates the value expression, if necessary, and compares the value with the codes in the table. The comparison is case sensitive and proceeds character-by-character. Thus, a value expression of 5 will not match a code of 05, and a value expression of Rig will not match a code of RIG.

Regardless of whether or not there is a match, the description table remains in the DATATRIEVE workspace until you either relinquish it with the RELEASE command or end the DATATRIEVE session. When you switch data dictionaries, DATATRIEVE does not release the description tables to which you have referred. In fact, you can still use those tables until you explicitly release them or end your DATATRIEVE session. (See Appendix B for a discussion of space saving techniques.)

13.3 Maintaining Description Tables

13.3.1 Displaying Description Table Information

You can list the names of all description tables in the current data dictionary and all tables in the DATATRIEVE workspace with the SHOW command.

```
DTR> SHOW TABLES
```

13.3.2 Displaying Description Tables

To print a description table on your terminal, use the SHOW command and specify the name of the description table to be displayed. You must have R (read) access privilege to the description table. Use the following format to display a description table:

```
DTR> SHOW desc-table-name  [ (passwd) ]
                           [   (*)   ]
```

13.3.3 Modifying Description Tables

You can modify a description table in the current data dictionary by using the DATATRIEVE editor. You must have C (control) access privilege to the description table. See Chapter 8 for information about the editor.

13.3.4 Deleting Description Tables

You can delete a description table from the current data dictionary with the DELETE command. You must have C (control) access privilege to the description table. Use the following format:

```
DTR> DELETE desc-table-name [ (passwd) ]
                            [   (*)    ]  ;
```

13.4 Protecting Description Tables

When you create a description table, DATATRIEVE stores its definition in the current data dictionary and creates a password table for it. DATATRIEVE automatically stores one UIC/PPN in the password table with full access privileges. The actual UIC/PPN that is stored in the password table is installation-dependent. Any user logged in under that UIC/PPN and the creator of the table have R (read), W (write), E (execute), M (modify), and C (control) access to the description table. Depending on the UIC/PPN in the password table, other users in the installation may be able to delete, modify, or print the description table.

If you want to grant additional privileges to other users or further restrict the use of the description table, you must modify its password table. Refer to Chapter 14 for more information on protecting dictionary definitions, including description tables, and on modifying password tables. Refer to Chapter 5 for the formats of the commands used in the process.

To guard against accidental deletion of your description table, you can maintain a backup copy of it by using the DATATRIEVE EXTRACT command (described in Section 5.18) to copy your table to a disk or tape file.

13.5 Example

This example works with a domain of WIDGETS, which consists of various items that need to be reordered from time to time. The procedure TAB finds the items no longer in stock, and uses ORDER-TABLE to list the people to contact when reordering.

```
DTR> READY WIDGETS
DTR> PRINT ALL WIDGETS

                                                        PARTS
                                                 PART     IN
        VENDOR                        ITEM      NUMBER  STOCK

ACME ASPHALT & SHINGLE      ASPHALT              1001      3
ACME ASPHALT & SHINGLE      SHINGLES             1002      0
ACME ASPHALT & SHINGLE      CUBE WALLS           1003   9999
PURGE SYSTEMS               ERASER-CHALK         3001      3
PURGE SYSTEMS               ERASER-PENCIL        3002      1
PURGE SYSTEMS               WHITE-OUT            3003   8645
PURGE SYSTEMS               MAGNETS-20 OZ.       3004      0
QUERY ENTERPRISES           LISTINGS             2001      4
QUERY ENTERPRISES           REPORTS              2002      0

DTR> SHOW ORDER-TABLE
TABLE ORDER-TABLE
"ACME ASPHALT & SHINGLE"  : "L. LANDFILL-999-555-1234",
"QUERY ENTERPRISES"       : "T. ABMOW-111-555-4321",
"PURGE SYSTEMS"           : "T. SKWAIRDEE-123-555-9876"
END-TABLE
DTR> SHOW TAB
PROCEDURE TAB
FIND WIDGETS WITH STOCK = 0 SORTED BY PART
FOR CURRENT PRINT ITEM, PART,
VENDOR VIA ORDER-TABLE ("CALL") USING X(30)
END-PROCEDURE
DTR> :TAB

                    PART
        ITEM       NUMBER          CALL

SHINGLES             1002  L. LANDFILL-999-555-1234
REPORTS              2002  T. ABMOW-111-555-4321
MAGNETS-20 OZ.       3004  T. SKWAIRDEE-123-555-9876

DTR>
```

Chapter 14
Security and Protection

In addition to the protection features offered by your operating system and by your file services, DATATRIEVE provides protection for your data and dictionary definitions through password tables. Password tables, which are stored in the data dictionary, regulate the type of access a user has to any object in the data dictionary.

Every dictionary object (domain, record, procedure, and description table definition) has an associated password table. This chapter describes the contents of a password table and the commands you use to maintain a table. The chapter also offers guidelines for developing a protection strategy to help ensure the integrity of your data.

Password tables can be effective against "browsing" and accidental corruption of the dictionary or data. They should be used to augment an overall security system for an installation.

14.1 Contents of a Password Table

A password table consists of one or more entries. Each entry contains the following information:

- A sequence number
- A lock type
- A key
- One or more access privileges

Figure 14–1 shows a sample password table containing three entries and illustrates the parts of a password table entry.

Figure 14–1: Sample Password Table

Sequence Number	Lock	Key	Privileges
1	UIC	[254,203]	W
2	PW	SWORDFISH	RM
3	UIC	[*,*]	R

14.1.1 Sequence Numbers

A sequence number is an integer, assigned by DATATRIEVE, that identifies a password table entry. You use sequence numbers to identify an entry you are adding to or deleting from the password table. These numbers are sequential, beginning with 1.

14.1.2 Lock Types

Each entry in the password table has a lock type that indicates whether you access the associated dictionary object by specifying a password or by using a UIC/PPN. There are two lock types: PW (for password) and UIC (for UIC/PPN).

14.1.3 Keys

When you attempt to access a dictionary object, you must provide the correct password for a PW lock, or own the correct UIC/PPN for a UIC lock. These are called keys.

Password Keys

If the lock type is PW, the key is a one- to ten-character password. Examples of passwords include: FISH-FRY, SWORDFISH, 1234, and PASSWD-9.

To access a dictionary object using a command or statement that includes an optional password argument, you can include the password directly in the command or statement. For example, the domain YACHTS might contain only one password table entry:

```
1   PW    SWORDFISH    RWEMC
```

To delete YACHTS from the dictionary, you must include the password in the DELETE command:

```
DTR> DELETE YACHTS (SWORDFISH);
```

For security reasons, you can suppress the printing of the password at the terminal. To do this, specify an asterisk enclosed in parentheses instead of the password. DATATRIEVE will then prompt for the password but not print it:

```
DTR> DELETE YACHTS (*);
Enter password for YACHTS:
```

Use this more secure technique for specifying a password, especially if you have a hard copy terminal.

UIC Keys

If the lock type is UIC (for UIC/PPN), the key is an account number known to the operating system. Under RSTS/E, an account number is called a project-programmer number, or PPN. Under all other operating systems, the account number is called the user identification code, or UIC. The UIC/PPN consists of a three digit octal group number followed by a three digit octal user number, separated by a comma and enclosed in brackets. For example:

[253,201]

A UIC/PPN lock can include either specific UIC/PPN numbers or asterisks; asterisks allow all users or all users in a specified group to access a dictionary object. For example, the following are all valid password table lock entries:

[253,201] (only the user with the account number [253,201] has access)

[253,*] (any user with group number 253 has access)

[*,*] (any user has access)

For a lock type of UIC, you do not specify a UIC or PPN in a command or statement. Rather, DATATRIEVE checks the UIC/PPN you used logging in to verify that you have access. For example, if the password table for the procedure BIG-YACHTS contains just the following entry, then you must log in under [253,201] in order to access the procedure:

```
1   UIC   [253,201]   RWEMC
```

But, if the password table contains the following entry, you can access the procedure regardless of your UIC/PPN:

```
1   UIC   [*,*]   RWEMC
```

NOTE

The UIC/PPN lock type is particularly useful in protecting procedures; you cannot use a password when invoking a procedure.

14.1.4 Access Privileges

An access privilege determines the type of access you have to the associated dictionary object and the operations you can perform on it. An access privilege is designated by a single letter, a string of letters, or a blank. Table 14–1 contains a list of access privileges and their meanings.

Table 14-1: Access Privileges

Access Privilege	Meaning
R	Read. User can SHOW or EXTRACT the associated dictionary object. For a domain, user can ready the domain for READ access only.
W	Write. User can ready the domain for READ, EXTEND, MODIFY, or WRITE access to retrieve, modify, store, or erase records.
E	Extend or Execute. For a domain, user can ready the domain for EXTEND access only to store records. For a procedure, user can execute the procedure. For a description table, user can refer to the description table (using VIA or IN). You must have E access to a record to ready the associated domain.
M	Modify. User can ready the domain for READ or MODIFY access to read or change records in the domain, but not to add or delete.
C	Control. User can issue the commands DEFINEP, DELETE, DELETEP, EDIT, and SHOWP.
blank	No access. User cannot access the dictionary object.

Each entry in a password table can include from one to five access privileges, or designate "no access." (A blank indicates that no access is permitted to a user with the corresponding key.) For example, the single letter R specifies that a user with the corresponding key has read access only. The letters RW specify that the user has both read and write access. Full access, designated by RWEMC, allows a user complete access to the dictionary object.

Each access privilege allows you to issue certain commands and statements. For example, with Read privilege, you can issue an EXTRACT command or ready a domain for READ access. (If you ready a domain for READ, you can then issue FIND and PRINT statements.) Table 14-2 contains a list of commands that you can issue if you are granted the corresponding access privilege. That table also shows the commands and statements you can issue after readying a domain. For example, if you are granted only Read privilege to a domain, you can ready that domain for READ, then issue a FIND statement to establish a collection. But, you cannot ready the domain for any other type of access.

Table 14-2: Commands/Statements by Privilege

Privilege for Domain	Command(s) Permitted	Additional Command(s) or Statement(s) Permitted
R	EXTRACT	
	READY...READ	FIND PRINT SELECT SORT SUM

(continued on next page)

Table 14-2: Commands/Statements by Privilege (Cont.)

Privilege for Domain	Command(s) Permitted	Additional Command(s) or Statement(s) Permitted
	SHOW	-
W	READY...READ	FIND PRINT SELECT SORT SUM
	READY...MODIFY	FIND MODIFY PRINT SELECT SORT SUM
	READY...WRITE	ERASE FIND MODIFY PRINT SELECT SORT STORE SUM
	READY...EXTEND	STORE
E	READY...EXTEND	STORE
M	READY...READ	FIND PRINT SELECT SORT SUM
	READY...MODIFY	FIND MODIFY PRINT SELECT SORT SUM
	READY...EXTEND	STORE
C	DEFINEP	-
	DELETE	-
	DELETEP	-
	EDIT	-
	SHOWP	-

Note: You must have E access to the associated record to ready a domain.

Any user with a group (or project) code of 1 (that is, a UIC/PPN in the form [1,n]) is automatically granted C (control) access in all password tables. Only users with C (control) access to a dictionary object can access its associated password table, edit or delete the dictionary object.

14.2 Creating Password Tables

When you define a dictionary object, DATATRIEVE automatically creates a password table for that object. The password table initially contains only one entry, a UIC/PPN that is granted full access privileges to the dictionary object. The specific UIC/PPN stored in the table is installation-dependent and is determined when the DATATRIEVE software is installed. The entry can be in one of the following formats:

[m,*]

Full access privileges are granted to any user with the same group (or project) code (m) as the creator of the definition. For example, if you log in under [253,201] and create a procedure definition, then an entry for [253,*] is stored in the password table for the procedure. Any user with a group (or project) code of 253 (such as [253,222]) also has full access privileges to the procedure.

[m,n]

Full access privileges are granted to any user with the same UIC/PPN as the creator of the dictionary definition. For example, if you log in under [253,201] and create a procedure definition, then an entry for [253,201] is stored in the password table for the procedure. Only users with a UIC/PPN of [253,201] can access the procedure.

[*,*]

All DATATRIEVE users, regardless of their UIC/PPN, are granted full access to the dictionary object.

Regardless of the actual UIC/PPN stored in the password table, the creator of the definition is always granted full access privileges (RWEMC) to the dictionary object.

You can grant privileges to additional users or further restrict the use of the dictionary object by changing its password table. The commands you use to change the table are summarized in Section 14.4 and described fully in Chapter 5.

NOTE

These commands are so important to the protection of data that only users with C (control) privilege can issue them.

14.3 DATATRIEVE's Processing of Password Tables

Whenever you use a table, invoke a procedure, or issue one of the following commands, DATATRIEVE checks the password table to verify that you have appropriate access privilege:

DEFINEP
DELETE
DELETEP
EDIT
EXTRACT
READY
SHOW
SHOWP

To verify that you have the correct access privilege, DATATRIEVE searches the password table, checking each entry until a match is found. DATATRIEVE searches the table in the following steps:

Step 1: DATATRIEVE stores your UIC/PPN and determines if you are a privileged user. A privileged user is one with a group (or project) code of 1 (that is, a UIC/PPN in the form [1,n]). At a minimum, DATATRIEVE grants a privileged user C (control) access. It may grant additional privileges, depending on the results of the following steps.

Step 2: DATATRIEVE checks the first entry in the password table.

If the lock type is PW, DATATRIEVE checks to see if you specified a password in the command. If you did and the password matches the entry's key, DATATRIEVE stops searching the password table and grants you the privilege (or privileges) listed with the first entry. If the password does not match or you did not specify a password, DATATRIEVE performs Step 3.

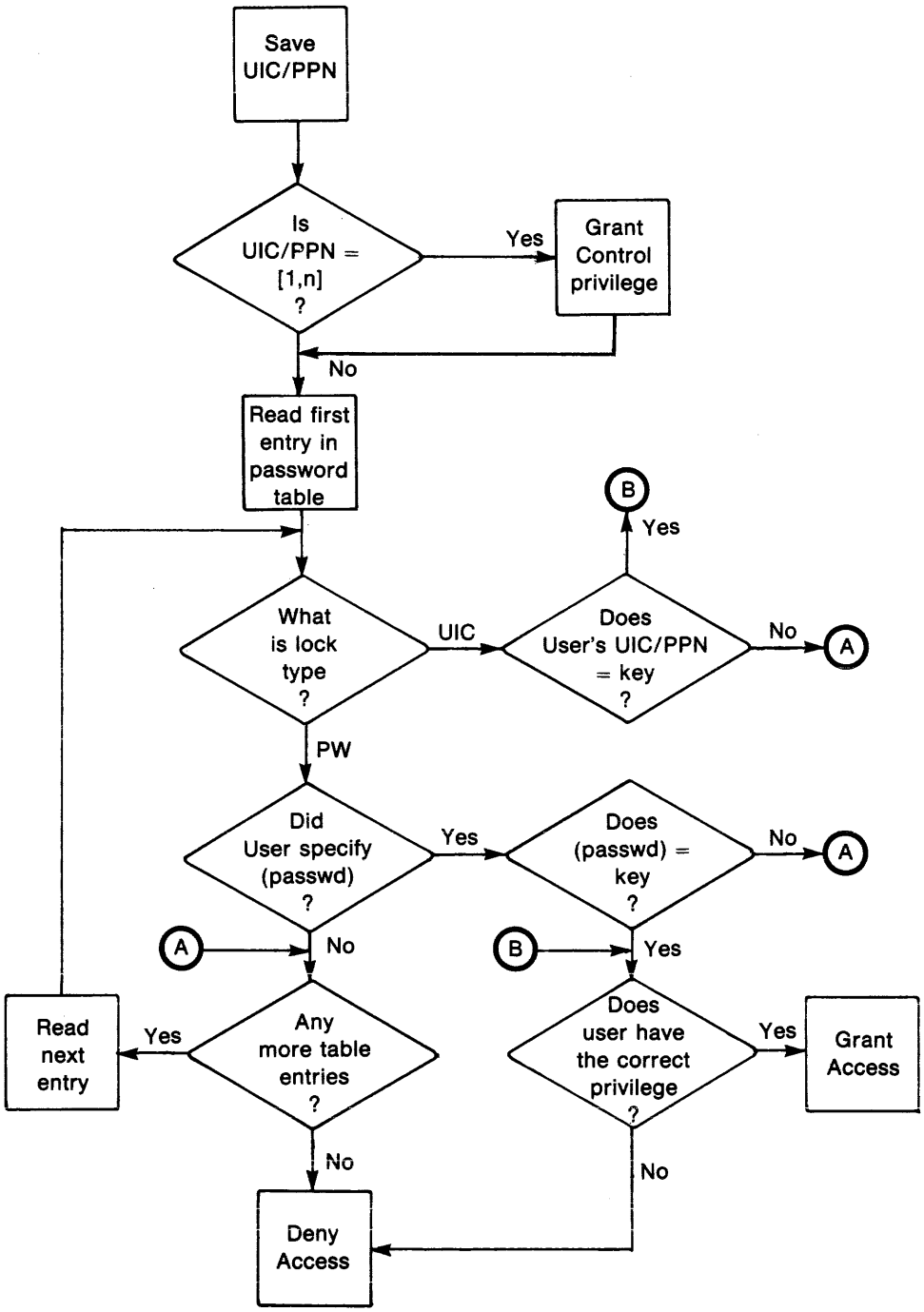
If the lock type is UIC, DATATRIEVE checks to see if your UIC/PPN matches the key. If it does, DATATRIEVE stops searching the table and grants you the privilege(s) listed with the first entry. If the UICs do not match, DATATRIEVE performs Step 3.

Step 3: DATATRIEVE checks the next entry in the table, following the same procedure as in Step 2.

When there are no more entries, DATATRIEVE denies all access to the dictionary object and rejects your command.

Figure 14–2 contains a flowchart that summarizes DATATRIEVE's processing of a password table.

Figure 14-2: Processing of Password Table



The following examples show DATATRIEVE's handling of some user requests. The examples use the following password table:

1	UIC	[253,201]	
2	UIC	[214,217]	CW
3	PW	[FISH-FRY]	M
4	UIC	[*,*]	R

Example 1:

A user with UIC/PPN [253,201] issues the following command:

```
DTR> SHOW YACHTS
```

DATATRIEVE checks the user's UIC/PPN and finds it as the first entry. Since no access privilege is granted, access to the domain is denied to the user.

Example 2:

A user with UIC/PPN [214,217] issues:

```
DTR> READY YACHTS WRITE
```

The first match in the password table (at entry 2) grants the user Write (and Control) privilege. The READY command executes.

Example 3:

A user with UIC/PPN [253,201] issues the following command:

```
DTR> MODIFY YACHTS (FISH-FRY)
```

Because the password FISH-FRY appears after the UIC [253,201], the user is denied Modify access to the domain.

Example 4:

A user with UIC/PPN [234,231] issues the same command:

```
DTR> MODIFY YACHTS (FISH-FRY)
```

Since the user does not have a UIC/PPN that specifically limits access, but does have the password key, access is granted.

14.4 Maintaining a Password Table

To maintain a password table that implements your password strategy, you must have C (control) access to the dictionary object associated with the password table. Control access allows you to display a password table and add or delete table entries.

14.4.1 Guidelines for Ordering Entries

When adding entries to a password table, keep in mind that the order of entries in the password table, and not the user, controls the access to the dictionary. The first match regulates a user's access; no one gets a second chance. Therefore, place the most restrictive entries first in a password table, the least restrictive entries last. The most restrictive entries are those that completely deny access to a specific UIC/PPN. The least restrictive entries allow access by any UIC/PPN. Follow these rules when adding entries to the password table:

1. Place entries that deny all privileges first. Use a lock type of UIC (instead of PW) for these entries.
2. Place restrictive entries that limit access to a specific UIC/PPN next.
3. Place the less restrictive entries (such as those requiring a password) next.
4. If access is allowed for any UIC/PPN (that is, if the key is [*,*]), place its entry last in the table.

14.4.2 Assigning Privileges

If you know which commands or statements you want to permit a user to issue, use Table 14-3 to find the privilege you must assign to that user.

Table 14-3: Privilege Requirements by Command/Statement

Command/Statement	Privilege Required
DEFINEP	Control access to the dictionary object associated with the password table.
DELETE	Control access to the dictionary object.
DELETEP	Control access to the dictionary object associated with the password table.
EDIT	Control access to the procedure or description table.
ERASE	Write access to the domain (to ready it).
EXTRACT	Read access to the dictionary object.
READY...READ	Read, Write, or Modify access to the domain and Execute access to the record.
...MODIFY	Modify or Write access to the domain and Execute access to the record.
...WRITE	Write access to the domain and Execute access to the record.
...EXTEND	Extend, Write, or Modify access to the domain and Execute access to the record.
SHOW domain-name record-name proc-name table-name	Read access to the dictionary object.
SHOWP	Control access to the dictionary object associated with the password table.

Use care when assigning the W privilege, particularly if a more restrictive privilege (such as R or M) would suffice. The Write privilege allows the user to perform the same functions as the R, M, and E privileges, but also allows the user to issue the ERASE command to delete records.

14.4.3 Displaying a Password Table

Use the SHOWP command to display a password table:

```
SHOWP  { domain-name  }  [ (passwd) ]
       { record-name  }  [ (*)      ]
       { proc-name    }
       { table-name   }
```

14.4.4 Adding Entries to a Password Table

Use the DEFINEP command to add an entry to a password table:

```
DEFINEP { domain-name  }  [ (passwd) ]  seq-no, { PW,  new-passwd }, pri
        { record-name  }  [ (*)      ]         { UIC, [m,n]       }
        { proc-name    }
        { table-name   }
```

Before adding any entry to a table, display the password table using SHOWP. The following example illustrates adding one entry to a password table:

```
DTR> SHOWP YACHTS (FISH-FRY)

     1,PW, "FISH-FRY", "RWEMC"

DTR> DEFINEP YACHTS (FISH-FRY) 2,UIC,[201,213],R
DTR> SHOWP YACHTS (FISH-FRY)

     1,PW, "FISH-FRY", "RWEMC"

     2,UIC, [201,213], "R"
```

14.4.5 Deleting Entries from a Password Table

The DELETEP command deletes one entry from a password table:

```
DELETEP { domain-name  }  [ (passwd) ]  seq-no
        { record-name  }  [   (*)    ]
        { proc-name    }
        { table-name   }
```

Use the SHOWP command before deleting an entry to verify that you are deleting the correct entry. For example:

```
DTR> SHOWP YACHTS (FISH-FRY)

     1,PW, "FISH-FRY", "RWEMC"

     2,UIC, [201,213], "R"

DTR> DELETEP YACHTS (FISH-FRY) 2
```

After you delete an entry, DATATRIEVE renumbers the entries so that they are sequential, beginning with 1.

If you delete all entries that have C (control) access, then the only way to change the password table is to log in using a privileged UIC/PPN.

Chapter 15
Hierarchies and Views

This chapter describes two ways to access the data you need: hierarchies and views. A hierarchy allows you to define a record so that fields and their contents are structured in a tree-like manner. A hierarchy organizes fields in a way that clearly shows the subordination of their values. You define a hierarchy when you create a record definition in the data dictionary.

A view is a domain that allows you to use some (or all) fields in some (or all) records in one or more domains. Using a view, you can refer to fields and their values contained in another domain or domains, without duplicating their records or data. You define a view by creating a domain definition for it in the data dictionary.

Some views are also hierarchies. These views are called hierarchical views. All views that use records from more than one domain are hierarchical views.

Because both hierarchies and views require that you create field definitions, you should be familiar with the contents of Chapter 11 ("Creating Record Definitions") before creating and using a hierarchy or view.

15.1 Hierarchies

A hierarchy is a domain whose records contain lists. A list is a repetition of a field or a group of fields. You create a list within a record by using the OCCURS clause (Section 11.9). A hierarchy allows you to print field values in a list and to use a list as a record stream.

For example, you may want to create a domain containing records of families. Each record could contain the names of the father and mother and the names and ages of up to two children. Without using a hierarchy, the record definition can be:

```
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC X(10).
   03 KIDS.
      06 FIRST-KID.
         09 KID-NAME PIC X(10).
         09 AGE PIC 99 EDIT-STRING IS Z9.
      06 SECOND-KID.
         09 KID-NAME PIC X(10).
         09 AGE PIC 99 EDIT-STRING IS Z9.
```

When you print a record with this definition, DATATRIEVE prints the field values in the following order:

```
                          KID                KID
  FATHER      MOTHER      NAME      AGE      NAME      AGE

ARNIE    ANNE           SCOTT         2      BRIAN        0
```

Although this format of the printed output may be useful, a more meaningful format would contain a list of children, including the name and age of each child, such as the following:

```
                          KID
  FATHER      MOTHER      NAME          AGE

ARNIE         ANNE        SCOTT          2
                          BRIAN          0
```

You can group the children into a list by using a hierarchy. The record definition that causes DATATRIEVE to use the preceding format is:

```
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC X(10).
   03 KIDS OCCURS 2 TIMES.
      06 KID-NAME PIC X(10).
      06 AGE PIC 99 EDIT-STRING IS Z9.
```

This record definition indicates that the group fields KIDS is repeated twice (OCCURS 2 TIMES) in each record. Each elementary field subordinate to KIDS is also repeated twice.

15.1.1 A Sample Hierarchy: FAMILIES

To demonstrate the concept and uses of a hierarchy, we will be working with the domain FAMILIES. Like the YACHTS domain, FAMILIES is included in the sample DATATRIEVE data supplied with your system.

Each record in FAMILIES contains data on the parents and children of one family. As in the preceding example, each record includes the name of the father and mother and the name and age of each child. Each family can have from 0 to 10 children. Figure 15–1 shows the record definition associated with FAMILIES:

Figure 15–1: Record Definition for FAMILIES

```
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC 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 X(10) QUERY-NAME IS KID.
         09 AGE PIC 99 EDIT-STRING IS Z9.
```

When you display the fields in FAMILIES, DATATRIEVE identifies the field KIDS as a list:

```
DTR> SHOW FIELDS FOR FAMILIES

     FAMILIES
         FAMILY
             PARENTS
                 FATHER      [Character string]
                 MOTHER      [Character string]
             NUMBER-KIDS     [Number]
             KIDS    [List]
                 EACH-KID
                     KID-NAME (KID)     [Character string]
                     AGE        [Number]
```

15.1.2 Creating a Hierarchy

To create a hierarchy, you must define repetitions (or multiple occurrences) of at least one field in the record. The OCCURS clause (described in Section 11.9) defines a field with the fixed or variable number of occurrences and identifies that field as a hierarchy.

Fixed Number of Occurrences

If you create a hierarchy with a fixed number of occurrences, every record in the domain contains enough space to store the same number of repetitions of the list data. Use an OCCURS clause in the following format to create a fixed number of occurrences of a field:

```
OCCURS n TIMES
```

The following record definition reserves enough space in every record for two occurrences of the KIDS field:

```
01 FAMILY.
   03 PARENTS.
      06 FATHER PIC X(10).
      06 MOTHER PIC X(10).
   03 KIDS OCCURS 2 TIMES.
      06 KID-NAME PIC X(10).
      06 AGE PIC 99 EDIT-STRING IS Z9.
```

This format of the OCCURS clause can be used with an elementary or group field. And, a record definition can contain any number of OCCURS clauses in this format.

Variable Number of Occurrences

If you want to create a hierarchy that can contain a different number of repetitions of a field from one record to another, you must use an OCCURS clause in the following format:

```
OCCURS min TO max TIMES DEPENDING ON fld-name
```

A record definition can contain only one OCCURS clause in this format.

The record definition for FAMILIES illustrates a list with a variable number of occurrences. See Figure 15–1. In FAMILY–REC, the field KIDS has multiple occurrences. The actual number of occurrences in any record depends on the value of the NUMBER–KIDS field. If the value is 0, there are no occurrences of the field; if it is 1, there is one occurrence, and so on. Each occurrence of the KIDS field contains two elementary field: KID–NAME and AGE. The group field EACH–KID allows you to refer to these two fields by a single field name.

Printing the domain shows the relationship between NUMBER–KIDS and the fields KID–NAME and AGE. Figure 15–2 shows the contents of all records in FAMILIES. Note that the values of KID–NAME and AGE appear in a list.

Figure 15–2: FAMILIES

```
DTR> PRINT FAMILIES

                          NUMBER    KID
  FATHER       MOTHER      KIDS     NAME      AGE

JIM            ANN           2    URSULA       7
                                  RALPH        3
JIM            LOUISE        5    ANNE        31
                                  JIM         29
                                  ELLEN       26
                                  DAVID       24
                                  ROBERT      16
JOHN           JULIE         2    ANN         29
                                  JEAN        26
JOHN           ELLEN         1    CHRISTOPHR   0
ARNIE          ANNE          2    SCOTT        2
                                  BRIAN        0
SHEARMAN       SARAH         1    DAVID        0
TOM            ANNE          2    PATRICK      4
                                  SUZIE        6
BASIL          MERIDETH      6    BEAU        28
                                  BROOKS      26
                                  ROBIN       24
                                  JAY         22
                                  WREN        17
                                  JILL        20
ROB            DIDI          0
JEROME         RUTH          4    ERIC        32
                                  CISSY       24
                                  NANCY       22
                                  MICHAEL     20
TOM            BETTY         2    MARTHA      30
                                  TOM         27
GEORGE         LOIS          3    JEFF        23
                                  FRED        26
                                  LAURA       21
HAROLD         SARAH         3    CHARLIE     31
                                  HAROLD      35
                                  SARAH       27
EDWIN          TRINITA       2    ERIC        16
                                  SCOTT       11
```

15.1.3 Referring to a List

You cannot refer to field values contained in a list as you would other fields. To refer to field values in a list, specify the list in a record selection expression. See Section 6.3 for the format of a record selection expression. You can use rse's that reference lists in the following ways:

- Nested within another rse
- Nested within a FOR statement
- Nested within a print list
- With a selected record

Nested in a Record Selection Expression

To restrict a record stream on the basis of fields contained in a list, nest one rse within another. You do this by using the relational operator ANY. The following examples show how to restrict record streams using nested rse's:

```
DTR> PRINT FAMILIES WITH ANY KIDS WITH AGE BT 13 AND 20

                          NUMBER    KID
  FATHER       MOTHER      KIDS     NAME      AGE

JIM            LOUISE         5    ANNE        31
                                    JIM         29
                                    ELLEN       26
                                    DAVID       24
                                    ROBERT      16
BASIL          MERIDETH       6    BEAU        28
                                    BROOKS      26
                                    ROBIN       24
                                    JAY         22
                                    WREN        17
                                    JILL        20
JEROME         RUTH           4    ERIC        32
                                    CISSY       24
                                    NANCY       22
                                    MICHAEL     20
EDWIN          TRINITA        2    ERIC        16
                                    SCOTT       11

DTR> FIND FAMILIES WITH ANY KIDS WITH AGE EQ 24
[3 records found]
DTR> PRINT FAMILIES WITH ANY KIDS WITH KID-NAME CONTAINING "Y"
                          NUMBER    KID
  FATHER       MOTHER      KIDS     NAME      AGE

BASIL          MERIDETH       6    BEAU        28
                                    BROOKS      26
                                    ROBIN       24
                                    JAY         22
                                    WREN        17
                                    JILL        20
JEROME         RUTH           4    ERIC        32
                                    CISSY       24
                                    NANCY       22
                                    MICHAEL     20

DTR>
```

Nested in a Print List

The simplest way to print a list is to print the entire record containing the list. To print anything other than the entire record, you must specify a print list in the PRINT statement. The print list consists of field names (or other value expressions) and modifiers. Print lists are explained in Section 5.25. You can also speciify lists within a print list. For example:

```
DTR> PRINT FATHER, ALL KIDS OF FIRST 1 FAMILIES
```

The format for specifying a list in a print list is:

```
ALL  [print-list OF]  rse
```

You must include the keyword ALL and the rse must refer to the name of the list. The optional inner print list can contain names of fields in the list and modifiers (such as column headers). The following examples show print lists that use the list KIDS:

```
DTR> PRINT FATHER, ALL FIRST 1 KIDS SORTED BY AGE OF FAMILIES

                KID
  FATHER        NAME     AGE

JIM          RALPH         3
JIM          ROBERT       16
JOHN         JEAN         26
JOHN         CHRISTOPHR    0
ARNIE        BRIAN         0
SHEARMAN     DAVID         0
TOM          PATRICK       4
BASIL        WREN         17
ROB
JEROME       MICHAEL      20
TOM          TOM          27
GEORGE       LAURA        21
HAROLD       SARAH        27
EDWIN        SCOTT        11

DTR> FIND FAMILIES
[14 records found]
DTR> PRINT ALL MOTHER, ALL FIRST 1 KIDS SORTED BY DESC AGE

                KID
  MOTHER        NAME     AGE

ANN          URSULA        7
LOUISE       ANNE         31
JULIE        ANN          29
ELLEN        CHRISTOPHR    0
ANNE         SCOTT         2
SARAH        DAVID         0
ANNE         SUZIE         6
MERIDETH     BEAU         28
DIDI
RUTH         ERIC         32
BETTY        MARTHA       30
LOIS         FRED         26
SARAH        HAROLD       35
TRINITA      ERIC         16
```

(continued on next page)

```
DTR> PRINT ALL ALL KIDS

   KID
   NAME     AGE

URSULA       7
RALPH        3
ANNE        31
JIM         29
ELLEN       26
       .
       .
       .
SARAH 27
ERIC        16
SCOTT 11
```

Nested in a FOR Statement

The FOR statement executes a statement on each record in a record stream. If the records contain lists, then you can apply statements to the fields contained in each list by nesting an rse in the FOR statement. For example:

```
DTR> FOR FAMILIES PRINT KIDS WITH AGE = 0

   KID
   NAME     AGE

CHRISTOPHR 0
BRIAN      0
DAVID      0

DTR> FOR FAMILIES
DTR> FOR KIDS WITH AGE = 0 MODIFY USING AGE = 1
DTR> FOR FAMILIES PRINT KIDS WITH AGE = 1

   KID
   NAME     AGE

CHRISTOPHR 1
BRIAN      1
DAVID      1

DTR> FOR FAMILIES FOR KIDS MODIFY USING AGE= AGE + 1
```

With a Selected Record

If you use the SELECT statement to select a record containing a list, then you can refer to the list very easily. All the records in a hierarchy contain lists that all have the same name. However, if there is a selected record, DATATRIEVE knows which list you refer to when you use a list name. In fact, you can even use the list as a collection. For example:

```
DTR> FIND A IN FAMILIES
[14 records found]
DTR> SELECT 8
DTR> PRINT
```

(continued on next page)

```
                          NUMBER       KID
  FATHER       MOTHER      KIDS       NAME      AGE

BASIL          MERIDETH       6     BEAU         28
                                     BROOKS       26
                                     ROBIN        24
                                     JAY          22
                                     WREN         17
                                     JILL         20

DTR> PRINT KID-NAME OF KIDS
   KID
   NAME
BEAU
BROOKS
ROBIN
JAY
WREN
JILL
DTR> FIND KIDS

[6 records found]
DTR> PRINT CURRENT WITH AGE BT 20 AND 24

   KID
   NAME      AGE
ROBIN        24
JAY          22
JILL         20

DTR> SELECT 5 A
DTR> PRINT KIDS

   KID
   NAME      AGE
SCOTT         2
BRIAN         0

DTR> SELECT 2
DTR> MODIFY AGE
Enter AGE: 1
DTR>
```

15.1.4 Changing the Length of a List

If you define a record with the OCCURS DEPENDING clause, you may be able to change the number of occurrences in a list. If you specify the MAX clause when defining the data file, all records have enough space for the maximum number of occurrences, and you can always change the number of occurrences. (You cannot exceed the maximum number of occurrences, of course.)

If you do not specify the MAX clause, and the file organization is sequential, the length of each record is determined when you store it. You can always decrease the number of occurrences, and you can then increase the number of occurrences back to the original number.

If you do not specify the MAX clause, and the file organization is indexed, you cannot always change the number of occurrences. On VMS systems you can always change the number of occurrences. On non VMS systems, if you allow

duplicate values for the primary key, you cannot change the number of occurrences. However, you can get around this by erasing the record and storing a new record with the desired number of occurrences.

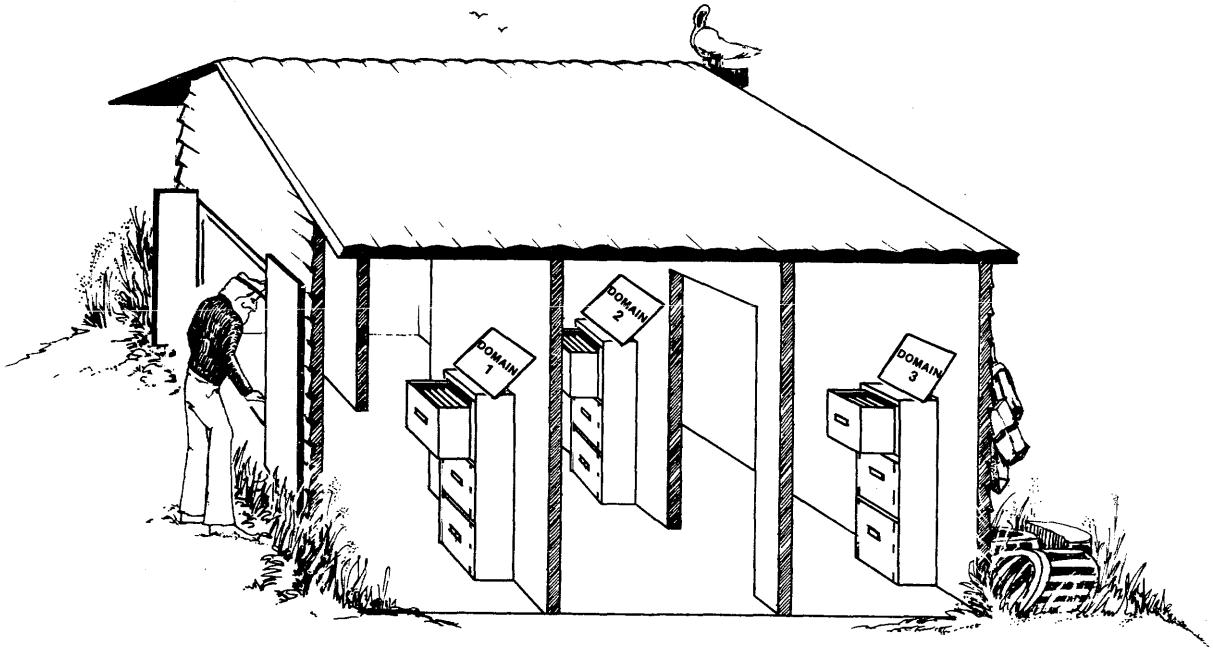
The following example shows how to add items to a list:

```
DTR> READY INDEXED-FAMILIES WRITE
DTR> FIND FIRST 1 INDEXED-FAMILIES
[1 Record found]
DTR> PRINT
No record selected, printing whole collection
                          NUMBER      KID
  FATHER      MOTHER      KIDS       NAME      AGE

JIM           ANN             2     URSULA       7
                                    RALPH        3


DTR> SELECT
DTR> MODIFY NUMBER-KIDS
Enter NUMBER-KIDS: 4
DTR> FIND KIDS
[4 records found]
DTR> SELECT 3
DTR> MODIFY
Enter KID-NAME: NICKY
Enter AGE: 2
DTR> SELECT 4
DTR> MODIFY
Enter KID-NAME: TAM
Enter AGE: 1
DTR> PRINT FIRST 1 INDEXED-FAMILIES


                          NUMBER      KID
  FATHER      MOTHER      KIDS       NAME      AGE

JIM           ANN             4     URSULA       7
                                    RALPH        3
                                    NICKY        2
                                    TAM          1
```

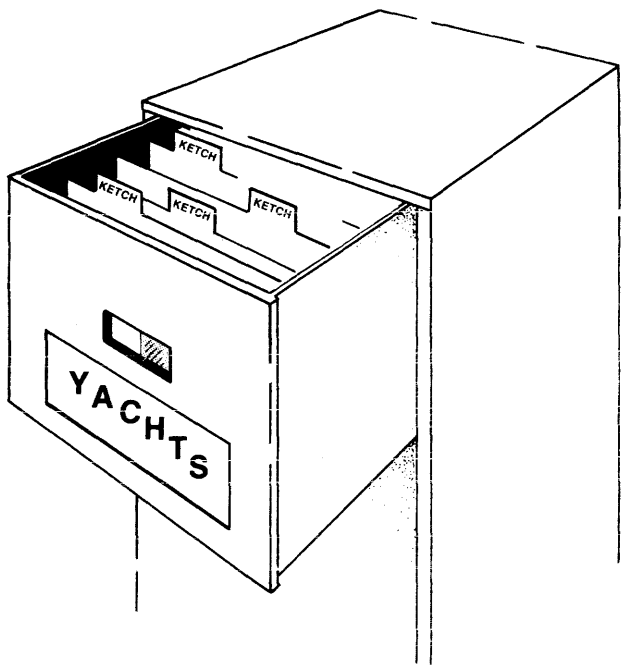
15.2 Views

A view is a special type of domain that is defined in terms of other domains. You can specify whether a view uses all the records in a domain or only some of them.

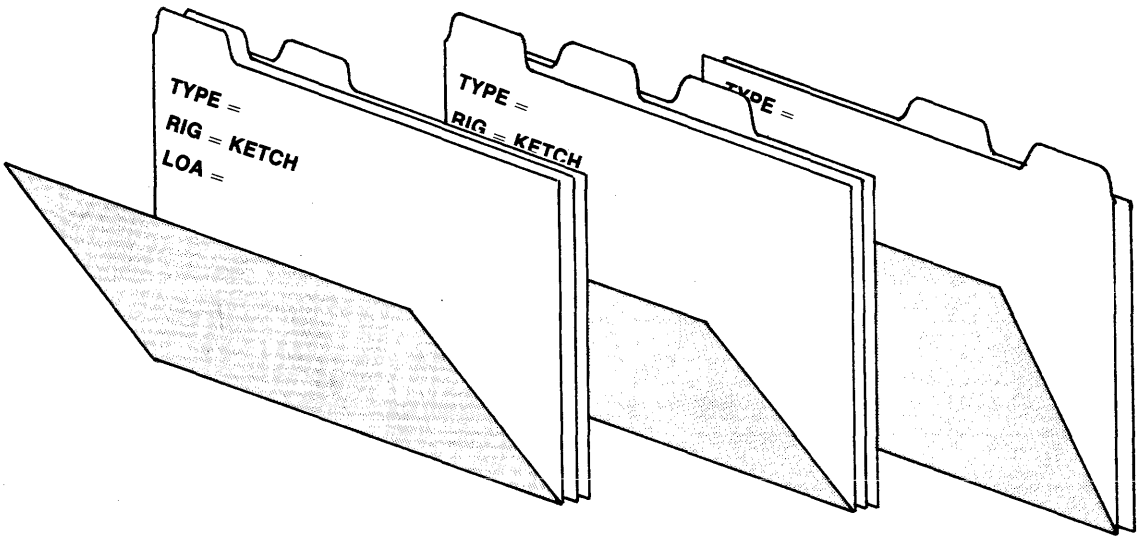
A view gets its name because it allows you to "see" data in a manner that is different from the way you see it in its original domain. It is analogous to a window that provides a view into a house. Depending on the location of the window, you can see into one or more rooms. Similarly, depending on how you define a view, you can see data in one or more domains.



Just as a window's size lets you see everything in a room or only a few things, a view allows you to see all records or just selected records in the domain(s).



The construction of the view can let you see some fields or all fields in a record.



You can also change the apparent order of fields in a record to suit your needs.

A view allows you to read and modify selected field values. There is no data stored in a view, so you cannot store or erase records. A view is only a means of working with the data in other domains.

15.2.1 Two Sample Views: KETCHES and SAILBOATS

A view lets you refer to a subset of fields from the records of another domain. For example, the record definition for YACHTS contains seven elementary fields and three group fields:

```
RECORD YACHT
USING
01 BOAT.
   03 TYPE.
      06 MANUFACTURER PIC C(10)
         QUERY-NAME IS BUILDER.
      06 MODEL PIC X(10).
   03 SPECIFICATIONS.
      QUERY-NAME SPECS.
      06 RIG PIC X(6).
         VALID IF RIG EQ "SLOOP", "KETCH", "MS", "YAWL".
      06 LENGTH-OVER-ALL PIC XXX
         VALID IF LOA BETWEEN 15 AND 50
         QUERY-NAME IS LOA.
      06 DISPLACEMENT PIC 99999
         QUERY-HEADER IS "WEIGHT"
         EDIT-STRING IS ZZZ,ZZ9
         QUERY-NAME IS DISP.
      06 BEAM PIC 99.
         PRICE PIC 99999
         VALID IF PRICE DISP*1.3 OR PRICE EQ 0
         EDIT-STRING IS $$$,$$$.
   ;
```

However, you may want to work with only a few fields of the record (such as TYPE, LOA, and PRICE). You could create a record definition for those fields, then create a domain and data file for the records. You would also have to store one record in the data file for each record in YACHTS. The result is a data file that duplicates some field values in an existing data file (YACHT.DAT). Maintaining these two files so that they always contain the same field values would be difficult.

On the other hand, you can define a view that allows you to look at just the fields in YACHTS that you want without duplicating field values and incurring the additional time and overhead involved in creating another data file and record definition.

A view also lets you work with a specified subset of records from another domain. For instance, you may want to work with only the records for ketches and no other rig type. The following example shows a view definition that allows you to work with four fields of the yachts that are ketches:

```
DTR> DEFINE DOMAIN KETCHES OF YACHTS USING
DFN> 01 KETCH OCCURS FOR YACHTS WITH RIG EQ "KETCH".
DFN>    03 TYPE FROM YACHTS.
DFN>    03 LOA FROM YACHTS.
DFN>    03 PRICE FROM YACHTS.
DFN> ;
DTR>
DTR> READY KETCHES
DTR> PRINT FIRST 4 KETCHES

                          LENGTH
                           OVER
MANUFACTURER      MODEL     ALL     PRICE

 ALBERG         37 MK II    37     $36,951
 CHALLENGER     41          41     $51,228
 FISHER         30          30
 FISHER         37          37
```

You can also use field values from more than one domain. The sample DATATRIEVE data contains the domain OWNERS. OWNERS contains records of people who own yachts. Each record contains the owner's name and the name, builder, and model of the owner's yacht:

```
DTR> SHOW OWNER-RECORD
RECORD OWNER-RECORD
01 OWNER.
   03 NAME PIC X(10) QUERY-HEADER IS "OWNER"/"NAME"
      EDIT-STRING IS X(5).
   03 BOAT-NAME PIC X(17) QUERY-HEADER IS "BOAT NAME".
   03 TYPE.
      06 BUILDER PIC X(10).
      06 MODEL PIC X(10).
;
DTR> READY OWNERS
DTR> PRINT FIRST 1 OWNERS

OWNER
NAME       BOAT NAME       BUILDER      MODEL

SHERM MILLENNIUM FALCON ALBERG      35
```

If you define a view that refers to both OWNERS and YACHTS, you can use any field or any record in OWNERS and any field or record in YACHTS. For example, you can look at just the owner's name and yacht type from the OWNERS domain and yacht's price from the YACHTS domain.

The following example shows the definition of a view that uses field values from both YACHTS and OWNERS. The view, SAILBOATS, uses each record in YACHTS (OCCURS FOR YACHTS), and it uses each field in these records (BOAT FROM YACHTS). But, it uses only a subset of records from OWNERS: Only those records with a boat type that is the same as a boat type in YACHTS (OWNERS WITH TYPE EQ BOAT.TYPE) are included in the view. SAILBOATS uses only the NAME field from records in the OWNERS domain.

```
DTR> SHOW SAILBOATS
DOMAIN SAILBOATS
 OF YACHTS, OWNERS USING
01 SAILBOAT OCCURS FOR YACHTS.
   03 BOAT FROM YACHTS.
   03 SKIPPERS OCCURS FOR OWNERS WITH TYPE EQ BOAT.TYPE.
      05 NAME FROM OWNERS.
;
DTR> READY SAILBOATS
DTR> PRINT FIRST 4 SAILBOATS

                                  LENGTH
                                   OVER                              OWNER
MANUFACTURER    MODEL      RIG     ALL    WEIGHT  BEAM   PRICE      NAME

 ALBERG       37 MK II    KETCH    37     20,000   12   $36,951
 ALBIN        79          SLOOP    26     4,200    10   $17,900
 ALBIN        BALLAD      SLOOP    30     7,276    10   $27,500
 ALBIN        VEGA        SLOOP    27     5,070    08   $18,600    STEVE
                                                                    HUGH

DTR>
```

The sample views KETCHES and SAILBOATS are both used in Section 15.2.3.

15.2.2 Defining a View

You define a view with the DEFINE DOMAIN command. The format of the command for defining a view is:

```
DEFINE DOMAIN view-name OF domain-name [,domain-name-2] ... USING

01      fld-name-1 OCCURS FOR rse-1.

[ lev-no-m fld-name-n { OCCURS FOR rse-n  } ]
                      { FROM domain-name-n }
                .
                .
                .

  ;
```

After the keyword OF you must list each domain that the view uses. The domains that you list cannot be views. You may specify them in any order. If you list more than one domain, separate the domains with commas. You must end each field definition with a period and end the view definition with a semicolon.

You can use only two clauses to define the fields in a view: OCCURS FOR and FROM. The top level field must be defined with an OCCURS FOR clause. The record selection expression in the first OCCURS FOR clause determines the number of records in the view. Each subsequent OCCURS FOR clause creates a list within the view, so a view that contains more than one OCCURS FOR clause is always a hierarchy.

Each FROM clause specifies which domain contains that field. The domain must be the same domain specified in the last previous OCCURS FOR clause. The field name must be either a field name or a query name from that domain.

In general, if you define a view that uses only one domain, use an OCCURS FOR clause to define the top level field and then use FROM clauses to specify which fields are included in the view. If you define a view using more than one domain, you must group the the field definitions by the domain which they refer to, putting the field definition with an OCCURS FOR clause first in each group.

If two or more fields have the same name then you may have to qualify the field name using the as explained in Section 6.1.2. For example, in defining SAILBOATS, it is necessary to qualify the field name TYPE:

```
DTR> DEFINE SAILBOATS
DFN> OF YACHTS, OWNERS USING
DFN> 01 SAILBOAT OCCURS FOR YACHTS.
DFN>    03 BOAT FROM YACHTS.
DFN>    03 SKIPPERS OCCURS FOR OWNERS WITH TYPE EQ BOAT.TYPE.
DFN>       05 NAME FROM OWNERS.
DFN> ;
```

15.2.3 Using a View

To use a view, you must ready it as you would any other domain. Do not ready the domains used by the view. To ready a view, you must have the proper access privilege to the view, and you must also have the same access privilege to the domains used by the view.

You cannot store or erase records in a view. Otherwise, you can use a view just as you would any other domain. For example:

```
DTR> READY KETCHES MODIFY
DTR> FIND KETCHES WITH PRICE EQ 0
[4 records found]
DTR> PRINT ALL

                          LENGTH
                           OVER
MANUFACTURER    MODEL      ALL     PRICE

 FISHER         30          30
 FISHER         37          37
 PEARSON        365         36
 PEARSON        419         42

DTR> FOR CURRENT PRINT THEN MODIFY PRICE

                          LENGTH
                           OVER
MANUFACTURER    MODEL      ALL     PRICE

 FISHER         30          30
Enter PRICE:$30,000
 FISHER         37          37
Enter PRICE: 45,000
 PEARSON        365         36
Enter PRICE: 32000
 PEARSON        419         42
Enter PRICE: 54000

DTR> PRINT ALL

                          LENGTH
                           OVER
MANUFACTURER    MODEL      ALL     PRICE

 FISHER         30          30     $30,000
 FISHER         37          37     $45,000
 PEARSON        365         36     $32,000
 PEARSON        419         42     $54,000

DTR> FINISH
```

All views that use more than one domain are hierarchies. To refer to field values contained in a list, you must use one of the methods explained in Section 15.1.3. For example:

```
DTR> READY SAILBOATS WRITE
DTR> FIND A IN SAILBOATS WITH ANY SKIPPERS
[6 records found]
DTR> PRINT ALL
```

(continued on next page)

```
                                LENGTH
                                 OVER                                  OWNER
MANUFACTURER    MODEL      RIG    ALL    WEIGHT BEAM   PRICE   NAME

 ALBIN          VEGA       SLOOP  27      5,070  08   $18,600 STEVE
                                                               HUGH
 C&C            CORVETTE   SLOOP  31      8,650  09           JIM
                                                               ANN
 ISLANDER       BAHAMA     SLOOP  24      4,200  08    $6,500 JIM
                                                               ANN
                                                               STEVE
                                                               HARVE
 PEARSON        10M        SLOOP  33     12,441  11           TOM
 PEARSON        26         SLOOP  26      5,400  08           DICK
 RHODES         SWIFTSURE  SLOOP  33     14,000  10           JOHN

DTR> SELECT 3
DTR> FIND SKIPPERS
[4 records found]
DTR> PRINT ALL

OWNER
NAME

JIM
ANN
STEVE
HARVE

DTR> SELECT 2
DTR> MODIFY NAME
Enter NAME: ANNE
DTR> PRINT BOAT, ALL SKIPPERS SORTED BY NAME OF A

                                LENGTH
                                 OVER                                  OWNER
MANUFACTURER    MODEL      RIG    ALL    WEIGHT BEAM   PRICE   NAME

 ALBIN          VEGA       SLOOP  27      5,070  08   $18,600 HUGH
                                                               STEVE
 C&C            CORVETTE   SLOOP  31      8,650  09           ANN
                                                               JIM
 ISLANDER       BAHAMA     SLOOP  24      4,200  08    $6,500 ANNE
                                                               HARVE
                                                               JIM
                                                               STEVE
 PEARSON        10M        SLOOP  33     12,441  11           TOM
 PEARSON        26         SLOOP  26      5,400  08           DICK
 RHODES         SWIFTSURE  SLOOP  33     14,000  10           JOHN

DTR>
```

Chapter 16
Creating New Domains from Old

This chapter describes how you can create new domains with data from existing ones. You might do this to:

- Describe new fields for a domain
- Change field descriptions
- Restructure the fields
- Combine data from two or more domains into one
- Create a copy of a domain for testing
- Change the file organization
- Change the index structure (key fields)
- Select subsets of records

To create the new domain, you do the following:

1. Define the new domain, its record, and its file.
2. Ready the old and new domains.
3. Use a FOR statement to transfer the data.

If the old domain name appears in procedures, you can still use those procedures if you do one of the following:

1. Delete the old domain definition; then create a new domain definition using the new record definition and the new file.

or:

2. Edit the procedures.

16.1 The Current Domain

Consider this domain, PROJECTS:

```
RECORD PROJECTS-REC
01 PROJECT.
   03  PROJ-CODE        PIC 9(3) QUERY-NAME IS NUM.
   03  PROJ-NAME        PIC X(10) QUERY-NAME IS NAME.
   03  MANAGER-NUM      PIC 9(5).
;
```

The domain contains three fields:

```
DTR> PRINT PROJECTS

PROJ     PROJ      MANAGER
CODE     NAME        NUM

002   GROUNDS       00006
005   BUILDING 2    00003
008   SHED          00002
018   RESEARCH      00006
037   PUB REL       00008
073   MATERIALS     00002

DTR>
```

16.2 Defining the New Domain

If you needed a domain with space for two more data items, you might define a new domain with this description:

```
DTR> DEFINE DOMAIN NEW-PROJECTS
DFN>  USING NEW-PROJECTS-REC ON NEWPROJ.DAT;
DTR> DEFINE RECORD NEW-PROJECTS-REC
DFN> 01 PROJECT.
DFN>     03  PROJ-CODE        PIC 9(3) QUERY-NAME IS NUM.
DFN>     03  PROJ-NAME        PIC X(10) QUERY-NAME IS NAME.
DFN>     03  PROJ-COST        PIC 9(5)V99 EDIT-STRING IS $$$,$$9.99.
DFN>     03  MANAGER-NUM      PIC 9(5).
DFN>     03  MGR-NAME         PIC X(15).
DFN> ;
[Record NEW-PROJECTS-REC is 41 bytes long]
DTR> DEFINE FILE NEW-PROJECTS KEY=PROJ-CODE
DTR>
```

16.3 Creating the New Domain

To create the domain using the data from PROJECTS, first ready both domains. Ready NEW-PROJECTS with WRITE or EXTEND, because that's where the new records will be stored. Then, combine the FOR and STORE statements to transfer the data.

The FOR statement loops through the entire domain (processes each record); the STORE statement then stores a record in the new domain for each record in the old.

```
DTR> READY PROJECTS
DTR> READY NEW-PROJECTS WRITE
DTR> FOR PROJECTS
DTR>   STORE NEW-PROJECTS USING
DTR>   PROJECT = PROJECT
DTR>
```

For each matching field name, the STORE statement transfers the data from PROJECTS to NEW-PROJECTS. In this example, the group item name PROJECT implies all the elementary items in it. In a later example, you will see that you can also transfer fields individually.

You now have a new domain. Printing its contents, you see the two new fields, PROJ-COST and MGR-NAME. The STORE statement did not transfer values to these fields from the old domain, because PROJECTS does not contain fields with the same names. However, the new fields do contain values. DATATRIEVE initializes numeric fields, like PROJ-COST, to zero. Character fields contain spaces.

```
DTR> PRINT NEW-PROJECTS

PROJ     PROJ        PROJ     MANAGER       MGR
NUM      NAME        COST       NUM         NAME

002  GROUNDS          $0.00  00006
005  BUILDING 2       $0.00  00003
008  SHED             $0.00  00002
018  RESEARCH         $0.00  00006
037  PUB REL          $0.00  00008
073  MATERIALS        $0.00  00002

DTR>
```

16.4 Using a Record Subset

You can create the new domain from a subset of the old domain's records. For example, you might limit a domain to two managers' projects. Use an rse in the FOR statement:

```
DTR> READY PROJECTS
DTR> READY NEW-PROJECTS WRITE
DTR> FOR PROJECTS WITH MANAGER-NUM EQ 2, 6
DTR>   STORE NEW-PROJECTS USING
DTR>   PROJECT = PROJECT
DTR> PRINT NEW-PROJECTS

PROJ     PROJ        PROJ     MANAGER       MGR
NUM      NAME        COST       NUM         NAME

002  GROUNDS          $0.00  00006
008  SHED             $0.00  00002
018  RESEARCH         $0.00  00006
073  MATERIALS        $0.00  00002

DTR>
```

If the record definition contains a VALID IF clause, you can use an rse to avoid validation errors. If you do not do this, DATATRIEVE stops storing records when a validation error occurs. For example, if you included the clause VALID IF PROJ-CODE IN PROJ-TABLE you could avoid validation errors with the following FOR statement:

```
DTR> FOR PROJECTS WITH PROJ-CODE IN PROJ-TABLE
DTR>  STORE NEW-PROJECTS USING
DTR>  PROJECT = PROJECT
DTR>
```

You can also use an rse to print those records which do not satisfy the VALID IF clause. You can then make the necessary changes and store those records.

16.5 Combining Data from Two or More Domains

One reason for creating a new domain is to combine the data from two or more existing domains. For example, you might want to include the managers' names from another domain, MANAGERS. MANAGERS contains two fields:

```
DTR> SHOW MANAGERS-REC
RECORD MANAGERS-REC
01 MANAGER.
   03  MANAGER-NUM          PIC 9(5).
   03  MGR-NAME             PIC X(8).
;
DTR>
```

Printing MANAGERS shows that MANAGER-NUM corresponds to the field MANAGER-NUM in the domain PROJECTS:

```
DTR> PRINT MANAGERS

MANAGER      MGR
  NUM        NAME

 00002  BLOUNT
 00003  GERBLE
 00005  GORFF
 00006  PUFFNER
 00008  FEBNELL

DTR>
```

Using a FOR statement to match PROJECTS and MANAGERS records by manager number, you can combine the data into NEW-PROJECTS. The data comes from two existing domains. Therefore, you must first ready both:

```
DTR> READY PROJECTS
DTR> READY MANAGERS
DTR> READY NEW-PROJECTS WRITE
DTR> FOR PROJECTS
DTR>   FOR MANAGERS WITH MANAGER-NUM EQ PROJECT.MANAGER-NUM
DTR>    STORE NEW-PROJECTS USING
DTR>    BEGIN
DTR>     PROJECT = PROJECT
DTR>     MGR-NAME = MGR-NAME
DTR>    END
DTR>
```

Printing NEW-PROJECTS shows the results. Notice that the value of PROJ-COST in each record is zero: The field did not exist in either of the source domains.

```
DTR> PRINT NEW-PROJECTS

PROJ     PROJ         PROJ     MANAGER       MGR
NUM      NAME         COST       NUM         NAME

002  GROUNDS           $0.00   00006   PUFFNER
005  BUILDING 2        $0.00   00003   GERBLE
008  SHED              $0.00   00002   BLOUNT
018  RESEARCH          $0.00   00006   PUFFNER
037  PUB REL           $0.00   00008   FEBNELL
073  MATERIALS         $0.00   00002   BLOUNT

DTR>
```

This example also shows that you can change field descriptions as you create a new domain. In MANAGERS, the field MGR-NAME is eight characters long. In NEW-PROJECTS, the size of the corresponding field is 15 characters.

Appendix A
DATATRIEVE Error Messages

DATATRIEVE error messages fall into two categories: common and severe. Common error messages are straightforward informational statements indicating that you have made an error in a command or statement. Severe error messages, which occur rarely, include the name of a module and subroutine.

A.1 Common Error Messages

If any command or statement causes a common error, DATATRIEVE attempts to recover from the error and displays an error message. If DATATRIEVE successfully recovers, the DTR> prompt is issued, and all data items remain the same as before the command or statement. Most of these messages simply describe the condition that caused the error. For example, if you used an undefined name with a FIND command, DATATRIEVE would respond with a concise informational message:

```
DTR>FIND ZILCH
"ZILCH" is neither a collection nor a readied domain
DTR>
```

Notice that DATATRIEVE has successfully recovered from the error and returned you to DATATRIEVE prompt level. All data items remain unchanged. At this point you should retype the command including a valid collection or readied domain name.

Other common error messages are prompting messages that allow you to continue by entering the missing part of the command or statement. Such messages are enclosed in brackets. For example, if you enter the FIND command with no associated name at all, DATATRIEVE responds in the following way:

```
DTR>FIND RET
[Looking for "FIRST," domain name or collection name]
DTR>
```

Here you merely supply the missing name in order to continue.

```
DTR>YACHTS
```

NOTE

If you enter the SET NO PROMPT command, you will not receive prompting messages.

A.2 Severe Error Messages

A message including an informational remark preceded by a module and subroutine name indicates a severe error. For example:

```
(AL)ALGC: Bad block encountered during garbage collection
```

The module name in parentheses (AL) is followed by the subroutine name, ALGC. DATATRIEVE may recover from a severe error, displaying the interim message:

```
DATATRIEVE RECOVERING
```

In this case you are returned to DATATRIEVE prompt level and proceed as with a common error. On occasion, however, you can incur a severe error that will cause your session to abort. Such an error will return you to system command level.

All severe error messages, whether recoverable or not, should be reported to your system manager and to your support facility.

A.3 Reporting Severe Errors

If you encounter any severe error submit a Software Performance Report (SPR). Along with this report include a trace file of the error condition along with copies of associated record definitions, reports and data dictionaries.

A.3.1 Making a Trace File

To make a trace file, invoke DATATRIEVE and give the command OPEN. When DATATRIEVE asks you for a file name, respond with an output file specification.

Everything displayed on your terminal from this point on will be captured in this file. This includes prompts and messages as well as your typed input. After giving the OPEN command, recreate the situation that resulted in the severe error. Your trace file will document the error condition. To stop copying into the trace file, exit from DATATRIEVE or issue the command CLOSE.

Here is a sample OPEN/CLOSE trace file session:

```
>DTR
DTR>OPEN
Please supply output file name: ERRORS.LOG
DTR>(RECREATE YOUR ERROR CONDITION)
DTR>CLOSE
DTR>
```

A.3.2 Copying Definitions

Use the EXTRACT command to copy relevant domain and record definitions, as well as procedures to to an indirect command file. See Section 5.19 for information on how to use EXTRACT.

A.3.3 Error Submission

After you have copied errors, definitions and procedures, print a hard copy of the trace file and of the indirect command file holding the definitions and procedures. Then submit the files, along with appropriate SPR's. If these files are lengthy, copy them to tape and send them in that form. In this case, include a tape file of your data; use the RMSBCK utility to create this file.

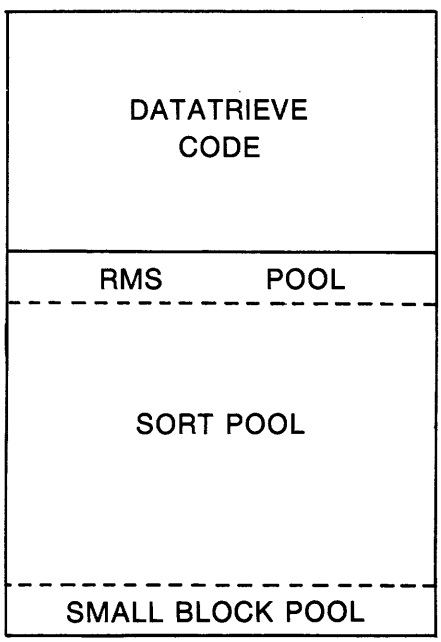
Appendix B
Optimization Techniques

Two considerations in using DATATRIEVE are speed and pool space. Section B.1 explains what DATATRIEVE pool space is, and Sections B.2 and B.3 describe how to optimize DATATRIEVE performance.

B.1 DATATRIEVE Pool Space

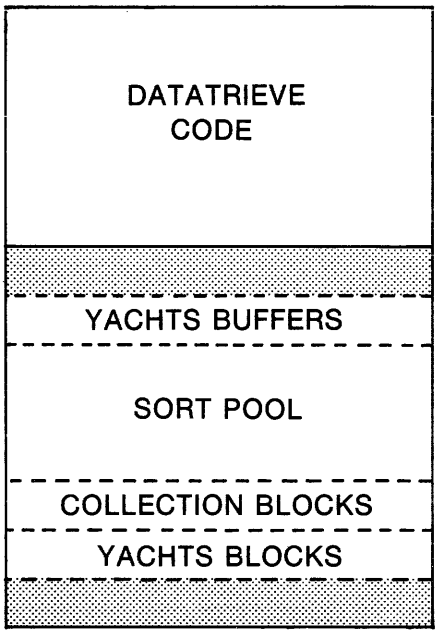
DATATRIEVE pool space is part of your allotted DATATRIEVE task image. Pool space is used for RMS buffers, DATATRIEVE internal blocks, and DATATRIEVE sort. Because the DATATRIEVE pool is a part of the task, not space on a disk, the pool cannot be enlarged when the task is at its maximum size. Figure B–1 shows the DATATRIEVE task and the three parts of DATATRIEVE pool.

Figure B–1: DATATRIEVE Task



The dashed lines indicate temporary boundaries: DATATRIEVE takes space from the sort pool and allocates it to the RMS pool and the small block pool when they need more space. A readied domain uses space in both the RMS pool and the small block pool. Collections, variables, and tables all use space in the small block pool. Figure B–2 shows the task after the commands READY YACHTS and FIND YACHTS.

Figure B–2: DATATRIEVE Task



The shaded regions represent space in the RMS pool and small block pool that DATATRIEVE always uses. DATATRIEVE uses RMS pool to connect you to the message file and data dictionary, and some small block pool to process your commands.

By issuing the command SHOW SPACE, you can see:

- The amount of space allocated to each pool
- How much space is in use
- The number of fragments

Figure B–3 shows the output of a SHOW SPACE command.

Figure B–3: Output of SHOW SPACE Command

	Current Memory Usage			
	Allocated	Used	Free	# of Fragments
RMS pool	7120	6792	328	3
Small block pool	3260	3104	156	4
Sort pool	10144	0	10144	0
Total	20524	9896	10628	7

Finishing domains and releasing collections, variables, and tables frees the space that they take up. Free space is returned to the sort pool only if it is adjacent to the sort pool. If space is freed in the interior of the RMS pool or small block pool, then it remains in that pool as a fragment.

The sort pool is used each time DATATRIEVE sort is invoked, whether within an rse or by the sort command. DATATRIEVE sort immediately seizes space from the sort pool, and releases it when the sort is finished. DATATRIEVE sort is slower when it has less space to work with, so keeping the sort pool as large as possible saves computing time. If the sort pool is too small, DATATRIEVE returns the message:

```
Sort workspace exhausted
Execution failed
```

As noted earlier, readied domains, collections, variables, and tables all use pool space until you release or finish them. Every command and statement uses pool space, but they release the space when they finish executing. If any command or statement requires more pool space than is available, DATATRIEVE returns a message that there is not enough pool space, and does not execute the command or statement.

B.2 Space Saving Techniques

The following steps increase effective workspace usage.

- Use a small file bucket size.

This applies only to RMS files created outside DATATRIEVE, whether by the RMSDEF utility or by another language. The file bucket size determines the size of the RMS buffers needed. A bucketsize of 8 would use approximately half of DATATRIEVE pool, and could cause you to run out of space repeatedly. A bucket size of 1 or 2 reduces the requirements for DATATRIEVE pool.

- Reduce the number of open files.

Finishing domains releases space in both the RMS pool and the small block pool. To decrease fragmentation, finish all domains by giving the command FINISH with no argument, then ready the domains you need.

- Release collections, tables, and variables.

Finishing a domain automatically releases all collections that came from it, but tables used by the domain must be released explicitly. Global variables must also be released explicitly. All of these data items take up space in the small block pool, and you should release them when they are no longer needed.

- Use tables instead of large IF-THEN-ELSE blocks.

It is more efficient to use a table than to use a large block of IF-THEN-ELSE statements. Tables can be used for verifying data and for printing descriptions. For example:

```
DTR> STORE PART VERIFY USING IF PART-NUMBER IN PART-TABLE THEN
DTR> PRINT "PART STORED" ELSE ABORT "BAD PART NUMBER"
```

- Control loading of tables.

If you know that a report uses a table, then load the table first by making a dummy reference to it. This decreases fragmentation and protects you against running out of pool in the report writer. Without this precaution, a report with a sort-key can seize all of the sort pool and not leave enough space to bring in the table. For example:

```
DTR> BEGIN
DTR> DECLARE A PIC 9.
DTR> A = 1 VIA PART-TABLE
DTR> END
```

- Use minimum record definitions.

Don't put unnecessary clauses and fields in record definitions. Pool space is used to store query-headers, edit-strings, query-names, VALID IF clauses, field names, group fields, and computed-by fields. Using short field names and eliminating nonessentials decreases the pool space each record takes up. The query-headers and edit-strings can be put into PRINT statements directly. Figure B-4 shows an efficient record definition for YACHTS.

Figure B-4: Alternate Definition for YACHTS

```
DTR> SHOW YACHT
RECORD YACHT
 USING
01 BOAT.
  03 TYPE.
     06 BUILDER PIC X(10).
     06 MODEL PIC X(10).
  03 SPECS.
     06 RIG PIC X(6).
     06 LOA PIC XXX.
     06 DISP PIC 99999.
     06 BEAM PIC 99.
     06 PRICE PIC 99999.
;
DTR>
```

B.3 Time Saving Techniques

DATATRIEVE does arithmetic computations more quickly if the numbers are stored as the COMP or INTEGER data type. To speed up arithmetic operations on fields, use the COMP data type when defining the fields involved.

When performing the same operation on several records, it is more efficient to use a FOR loop than to use many SELECT statements. For example:

```
DTR> FOR YACHTS WITH BEAM EQ 0 BEGIN PRINT; MODIFY BEAM END
```

The space saving recommendations in Section B.2 do not involve time trade-offs. In fact, they can save time by giving DATATRIEVE sort more space to work with. There is a trade-off involved with printing sorted groups of records. It is fastest to do a PRINT statement with an rse to select the desired records. This uses more dynamic pool space than doing a FIND followed by a SORT and a PRINT ALL. So, it is usually best to do a PRINT rse; but when there is some danger of running out of pool space, it is better to use a FIND and a SORT.

The time spent accessing files is affected by the types of files used for domains. If the records of a domain are usually accessed by the contents of one field then consider using an indexed file with that field as a key. Suppose that you wish to find the record with a certain value in that field. If the file organization is sequential or the field referenced is not a key, then each record in the domain must be checked. For a file with thousands of records, this means thousands of file accesses, rather than one. When defining an indexed file, remember that the primary key cannot be modified. You should not define more than 1 or 2 alternate keys, because with each alternate key you define, you increase the time needed to store and update records.

Appendix C
Keywords

ABORT
ADT
ADVANCED
ALL
ALLOCATION
AND
ANY
ASC
ASCENDING
AT
AVERAGE
BEGIN
BETWEEN
BOTTOM
BT
BUT
BY
CHANGE
CHARACTER
CLOSE
COL
COLLECTIONS
COLUMN
COLUMN-HEADER
COLUMNS-PAGE
COMP
COMP-1
COMP-2
COMP-3
COMP-5
COMP-6
COMPUTED
CONT
CONTAINING
COUNT
CURRENT
DATE
DECLARE
DECREASING
DEFINE
DEFINEP
DELETE
DELETEP
DEPENDING
DESC
DESCENDING
DICTIONARY
DISPLAY
DO
DOMAIN
DOMAINS
DOUBLE
DUP
EDIT
EDIT-STRING
ELSE
END
END-PROCEDURE
END-REPORT
END-TABLE
EQ
EQUAL
ERASE
EXCLUSIVE
EXIT
EXTEND
EXTRACT
FIELDS
FILE
FILLER
FIND
FINISH
FIRST
FOR
FROM
GE
GREATER-EQUAL
GREATER-THAN
GT
GUIDE
HELP
IF
IN
INCREASING
INTEGER
IS
KEY
LAST
LE
LEADING
LEFT-RIGHT
LESS-EQUAL
LESS-THAN
LINES-PAGE
LT
MAJOR-MINOR

MAX
MAX-LINES
MAX-PAGES
MEDIAN
MEMBERS
MIN
MODIFY
NE
NEW-PAGE
NEW-SECTION
NEXT
NO
NO-DATE
NO-NUMBER
NOT
NOT-EQUAL
NUMBER
OCCURS
OF
ON
OPEN
OR
OWNER
PACKED
PAGE
PIC
PICTURE
PRINT
PROCEDURE
PROCEDURES
PROMPT
PROTECTED
PW
QUERY-HEADER
QUERY-NAME
QUIT
READ
READY
REAL
RECORD
RECORDS
REDEFINES
RELEASE
REPEAT
REPORT
REPORT-HEADER
REPORT-NAME
SELECT
SEPARATE
SET
SETS
SHARED
SHOW
SHOWP
SIGN
SKIP
SORT
SORTED
SPACE
STORE
SUBSCHEMA
SUM
SUPERCEDE
SUPERSEDE
SYNC
TAB
TABLE
TABLES
THE
THEN
TIMES
TO
TOP
TOTAL
TRAILING
UIC
USAGE
USING
VALID
VERIFY
VIA
WITH
WRITE
ZONED

Appendix D
Differences from Version 1.1

In addition to having many new features, DATATRIEVE Version 2.0 allows the user more freedom than did Version 1.1, and contains some language changes. The DATATRIEVE compatibility policy is to support both the new and old ways of doing an operation, if possible, for one full release. After a full release the old way will not be supported. Changes in the DATATRIEVE language are listed in the following sections.

D.1 Changes in the Report Writer

- In Version 1.1 reports were ended with the statement "REPORT END". In Version 2.0 the proper statement is "END-REPORT". Both statements are supported in Version 2.0.
- The report options "NO-DATE" and "NO-NUMBER" should be given as "NO DATE" and "NO NUMBER". Both forms are supported in Version 2.0.
- The report writer uses a new formatter. The new formatter is better at aligning columns and wrapping report lines. This alters the appearance of the report produced. In the new format, the first field is not automatically printed starting in column 1; instead, all fields are spaced evenly across the page.
- You are no longer restricted to a maximum of 16 TOTALs within a report.

D.2 Changes in Data Types

- DATATRIEVE Version 1.1 allocated a three word binary data type for a COMP field between 10 and 14 digits long. The correct length for fields in this range is four words. DATATRIEVE Version 2.0 allocates four words for COMP fields in this range, and cannot read files containing the three word data types. (This does not apply to COMP-1, COMP-2, COMP-3,

COMP-5, or COMP-6.) If you have files containing these three word data types you should use DATATRIEVE Version 1.1 to restructure the fields to decimal length 15 or greater.

- Floating point numbers (COMP-1 and COMP-2) now have scaling. This means that PICTURE clauses containing a decimal position will now cause scaling of your data. If you have fields that are COMP-1 or COMP-2, and you don't want automatic scaling, you must take any decimal positions out of their PICTURE clauses.

D.3 Other Changes

- The precedence of statistical functions has changed. In DATATRIEVE Version 2.0 the statistical function is the high precedence operator. To take the total, average, max, or min of an expression you must put the expression in parentheses.
- The access control for records, procedures, and tables has been enhanced. In DATATRIEVE Version 2.0, you need R (read) access to SHOW or EXTRACT a record, procedure, or table. In order to use an object, you need the privilege E (execute). This permits a user to use but not see a procedure or table.
- The value "columns per page" (SET COLUMNS-PAGE) is now installation dependent. In Version 1.1 the DATATRIEVE default for this value was 132.
- In Version 2.0, the maximum input string from a terminal is 132 characters. In Version 1.1 the size was 255 characters. This means that the number of continuation lines permitted may decrease, since continuation lines are considered as part of one input string.

Appendix E
Alignment of Fields

DATATRIEVE has an optional clause that you can use in defining a record. The allocation clause controls the internal storage format for a record. Use the following format to specify an allocation clause:

```
DEFINE RECORD record-name USING [ALLOCATION IS {MAJOR-MINOR}]
                                               {LEFT-RIGHT }
fld-def-1, [fld-def-2],...
```

The allocation clause specifies whether DATATRIEVE uses LEFT-RIGHT alignment or MAJOR-MINOR alignment when storing records in the domain. It also tells DATATRIEVE how to read records from the data file. If you do not specify an alignment clause, then DATATRIEVE uses LEFT-RIGHT alignment.

Certain elementary fields have to be stored on two-byte, four-byte, or eight-byte boundaries. Fields that require a specific alignment are those defined with the following USAGE clauses:

```
DATE

COMP

COMP-1

COMP-2
```

Aligning fields on specific byte boundaries creates empty bytes, known as fill bytes. The exact location of these fill bytes depends on the allocation clause. For files which are used only by DATATRIEVE-11, it does not matter whether you use MAJOR-MINOR alignment or LEFT-RIGHT alignment.

If your files are created or read by VAX-11 COBOL, you should include an allocation clause. For more information on alignment and record allocation, see the *VAX-11 COBOL Language Reference Manual*.

Appendix F
DATATRIEVE Sorting Order

When you use the SORT command, DATATRIEVE arranges records in a collection in ASCENDING or DESCENDING order of the value of a key field. Generally, DATATRIEVE sorts records according to the ASCII value of the character in the key field. However, the DATATRIEVE sort order departs from ASCII order in one significant way: unlike the ASCII sequence, which places all upper case letters before all lower case letters ("Z" would sort before "a"), DATATRIEVE treats all lower case letters as if they were upper case: "a" or "A" would sort before "Z." See Chapter 5 for more information on the SORT command.

Table F–1 contains all the characters sorted in ascending order. The "space" (SP) has the lowest value, tilde (~) has the highest.

Table F-1: DATATRIEVE Sorting Sequence

Character	Character
SP	A,a
!	B,b
"	C,c
#	D,d
$	E,e
%	F,f
&	G,g
'	H,h
(	I,i
)	J,j
*	K,k
+	L,l
,	M,m
-	N,n
	O,o
/	P,p
0	Q,q
1	R,r
2	S,s
3	T,t
4	U,u
5	V,v
6	W,w
7	X,x
8	Y,y
9	Z,z
:	[
;	\
<	]
=	^
>	_
?	`
@	{
	|
	}
	~

Glossary

Access mode

The type of access you specify when readying a domain. Access modes are read, write, modify, and extend. See Section 5.26.

Access privilege

The types of access allowed to a class of users, determined by the password table. Access privileges are R (read), E (execute or extend), M (modify), W (write), and C (control).

ADT

See Application Design Tool.

Allow type

This determines the type of access allowed to other users. The allow types are shared, protected, and exclusive.

Application Design Tool

Part of DATATRIEVE that creates an indirect command file containing a domain definition, its associated record definition, and the appropriate DEFINE FILE command, by prompting the user for a definition of his data. See Chapter 9.

Boolean expression

An expression that DATATRIEVE evaluates as "True" or "False".

Central data dictionary

The dictionary to which you are connected when you invoke DATATRIEVE.

Character string literal

A value expression consisting of a string of characters enclosed in quotation marks.

Code

An entry to the left of a colon in a description table.

Collection

A temporary grouping of related records, established with the FIND command. A collection can include some or all of the records in a domain.

Column header

The characters that DATATRIEVE prints over a column of data. You can specify a column header in the PRINT statement.

Command

A direction to DATATRIEVE to perform a function. A command begins with a keyword and terminates with a carriage return or semicolon. A command cannot be combined with other commands or statements.

Command file

See indirect command file.

Command level

Command level means that DATATRIEVE is not in the middle of a command or statement. Command level is indicated by the prompt DTR>, but DATATRIEVE also gives this prompt when waiting for certain command or statement elements.

Comment

A user-specified character string that is preceded by an exclamation point and is not processed in any way by DATATRIEVE.

Compound statement

Two DATATRIEVE statements connected by the keyword THEN, or a BEGIN-END block.

Context record

The record which DATATRIEVE acts upon or uses the field values of.

Continuation character

A language element that allows a command or statement to be continued on the next input line: a hyphen in commands, literals, or names; a hyphen or carriage return in statements.

Control privilege

An access privilege that allows the user to issue the following commands: EDIT, DELETE, DELETEP, DEFINEP, and SHOWP.

Count

A statistical function whose value is equal to the number of records in the record stream.

Current collection

The collection established by the most recent FIND command.

Current dictionary

The dictionary to which the user is currently connected. Only one dictionary is current at any time.

Data dictionary

An indexed file used to store DATATRIEVE definitions and access privilege information.

Data Dictionary Definition

A data dictionary entry that contains a complete description of a record, domain, procedure, or table. It contains the structure of the data, though not the data itself.

Description table

A set of code and description pairs, stored in the data dictionary.

Detail line

The line or lines printed by the Report Writer for each record in the report.

Dictionary

See data dictionary.

Dictionary Object

A description table, procedure, record definition, or domain definition.

Domain

A group of records containing fields that all have the same structure.

Domain definition

An entry in a data dictionary that links a record definition with a data file.

Edit Character

A character used in an edit string. The edit characters are listed in Table 11–3.

Edit command level

Edit command level means that you can enter any of the editor commands. This command level is indicated by the prompt QED>.

Edit Mode

Same as edit command level.

Edit String

A string of characters that determines the format of printed data. You can specify an edit string in a field definition or in a PRINT statement.

Elementary field

The smallest accessible unit of data in a record. An elementary field contains a value.

Execute privilege

A type of privilege that allows the user to access a record, procedure, or table.

Expression

A command or statement element that represents a value or specifies a record.

Extend privilege

A privilege that allows a user to ready a domain for EXTEND access.

Field

A unit of data in a record. A field is defined in a record definition and contains a value or sub-fields.

File

A group of logically related records.

File spec

See file specification.

File Specification

Specifies a unique file. It has the format dev:[UFD]filename.ext;ver , where dev: is the device containing the file, UFD is the UFD of the directory containing the file, ext is the three letter file extension, and ver is the version number.

Global variable

A variable defined at DATATRIEVE command level. You must define it as an elementary field. A global variable may be referred to anywhere until you release it or exit from DATATRIEVE.

Group code

The first three digits of UIC or PPN.

Group field

A field which contains other fields, rather than containing a value.

Hierarchy

A grouping of related data items in a multi-level structure.

Indirect command file

A text file containing DATATRIEVE commands and statements. You can create indirect command files with ADT, the EXTRACT command, or a text editor.

Index key

A field that is used to order records in a file. The primary index key determines where a record is stored and cannot be changed.

Indexed File

A file that uses one or more fields as index keys.

Insert Mode

Insert mode means that the DATATRIEVE editor interprets all your input as new text to be entered into the dictionary object. Insert mode is indicated by the prompt IN>.

Keyword

A language element reserved for specific use in DATATRIEVE commands and statements. Keywords cannot be used in user-specified elements such as the names of fields, records, domains, or procedures.

Line pointer

The line pointer keeps track of your position in text that you are editing. The line pointer points to the entire current line.

List

A repetition of a field or a group of fields within a single record. You create a list within a record by using the OCCURS clause.

Literal value expression

A user-specified expression whose value is a character string literal or a numeric literal.

Local variable

A variable defined within a BEGIN–END block. It is not kept after the BEGIN–END block has been executed.

Lock type

The part of a password table entry that identifies the kind of key needed to access a dictionary item.

Modify privilege

A type of access privilege that allows the user to ready a domain for read or modify access.

Modify access

An access mode that allows the user to retrieve and modify records in a domain.

Name

A character string that identifies a domain, record, field, description table, collection, procedure, variable, or view.

Numeric Literal

A string of digits that DATATRIEVE interprets as a decimal number.

Password

A character string of up to 10 characters used to determine access privileges.

Password table

A collection of entries that regulates the type of access users have to an item in the data dictionary.

PICTURE clause

Part of a field definition that specifies the internal format of a field.

PPN

Project programmer number. Each user on RSTS/E has a two-number identification code enclosed in brackets which is used for logging in and determining privileges. The number is in the form [g,m], with g giving the user's group number and m giving the user's member number.

Print list

A list of one or more elements describing the data to be printed and the format.

Private data dictionary

A data dictionary created by the user.

Privileged user

A user whose group code is 1.

Procedure

A sequence of DATATRIEVE commands, statements, clauses, or arguments stored in the data dictionary under a procedure name.

Procedure definition

An entry in a data dictionary that describes a procedure.

Query header

An optional part of a field definition that specifies the default column header.

Query name

Part of the record definition which specifies an alternate name for a field.

Read privilege

This access privilege allows the user to show the associated dictionary item. For a domain the user can ready the domain for read access.

Record

A structural part of a file, recognized as a unit by DATATRIEVE. A record describes one instance of the items described by a domain. For example, in the domain YACHTS, each record describes one YACHT.

Record definition

An entry in a data dictionary that describes the fields in a record.

Record selection expression

A user-specified expression that indicates a record stream to be used in a DATATRIEVE command or statement.

Record stream

The group of records you specify with a record selection expression. You can only use the record stream in the command or statement which contains the rse.

Report writer

Part of DATATRIEVE that allows the user to create and print reports in a variety of formats.

RMS

Record Management System. DATATRIEVE uses this software to manage all files.

rse

Record selection expression.

RWMEC

R (read), E (execute or extend), M (modify), W (write), and C (control) privileges. Also called full access privilege.

Sort key

The name of the field used to sort the records in a collection or record stream and the direction of the sort (INCREASING or DECREASING).

Statement

A user-specified direction to DATATRIEVE to perform a function. A statement can be issued by itself or in combination with other statements.

Terminator

A language element that signals the end of a command or statement.

UFD

All files are contained in User File Directories, or UFD's. The UFD is a file listing all the files included in the Directory. The UFD is a two-number code in the form [g,m], and is usually the same as the UIC or PPN which you use to log in. On VMS you can specify a UFD in the form [directory-name].

UIC

Each user has a two-number identification code enclosed in brackets which is used for logging in and determining privileges. The number is in the form [g,m], with g giving the user's group number and m giving the user's member number.

User privilege

See access privilege.

Value expression

A language element that specifies a value.

Variable

A user defined name which represents a value.

View

A domain that allows you to use selected fields in records in one or more domains without duplicating the records or their data.

Write privilege

The access privilege that allows the user to ready a domain for read, extend, modify, or write access.

Index

F

G

H

I

P

Q

R

S

T

U

V

Reader's Comments

Note: This form is for document comments only. Digital will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number. _____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

—Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Commercial Engineering Publications MK1-2/ H3
DIGITAL EQUIPMENT CORPORATION
CONTINENTAL BOULEVARD
MERRIMACK N.H. 03054

— — Do Not Tear - Fold Here and Tape

Cut Along Dotted Line