

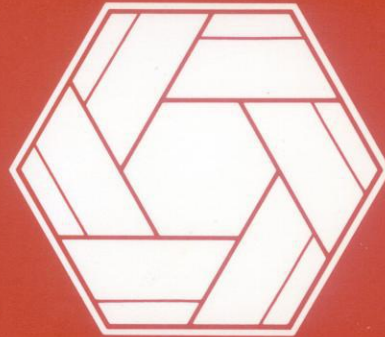
# VAX DATATRIEVE

---

## Guide to Writing Reports

Order No. AA-P862C-TE

Guide to Writing Reports



**digital**<sup>TM</sup>  
software

**VAX DATATRIEVE  
Guide to Writing Reports**

Order No. AA-P862C-TE

---

**November 1987**

This manual introduces the VAX DATATRIEVE Report Writer and contains the reference material for creating reports.

<b>OPERATING SYSTEM:</b>	<b>VMS</b>
	<b>MicroVMS</b>
<b>SOFTWARE VERSION:</b>	<b>VAX DATATRIEVE V4.1</b>

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 be used or copied only 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.

Copyright © 1984, 1985, 1987 by Digital Equipment Corporation. All Rights Reserved.

The postage-paid Reader's Comments forms at the end of this document request your critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

ACMS	PDP	VAX
CDD	RALLY	VAXcluster
DATATRIEVE	Rdb/ELN	VAXinfo
DEC	Rdb/VMS	VAX Information Architecture
DECnet	ReGIS	VAX/VMS
DECUS	TDMS	VIDA
MicroVAX	TEAMDATA	VMS
MicroVMS	UNIBUS	VT

**digital**™

IBM® is a registered trademark of International Business Machines Corporation.

**How to Use This Manual** vii

**1 Getting Started in Report Writing**

1.1 Using the DATATRIEVE Report Writer . . . . .	1-1
1.2 What the Report Writer Can Do . . . . .	1-2
1.3 Writing Simple Reports: Examples . . . . .	1-3
1.3.1 The PRINT Statement . . . . .	1-4
1.3.2 Summarizing Data with the SUM Statement . . . . .	1-4
1.3.3 Writing a Simple Report with the Report Writer . . . . .	1-5
1.3.4 Defining a Procedure to Produce a Report. . . . .	1-7
1.3.5 Printing Summary Statistics with the Report Writer . . . . .	1-9

**2 Designing a Report with the Report Writer**

2.1 Identifying the Data and Invoking the Report Writer . . . . .	2-2
2.2 Exiting from the Report Writer . . . . .	2-3
2.3 Using an Editor to Correct Mistakes . . . . .	2-4
2.4 Selecting the Output Medium . . . . .	2-4
2.4.1 Default: Output to Terminal . . . . .	2-5
2.4.2 Option 1: Output to a Line Printer . . . . .	2-5
2.4.3 Option 2: Output to a File. . . . .	2-5
2.4.4 Option 3: Prompt . . . . .	2-6
2.4.5 Using an ON Statement to Specify Multiple Output Media . . . . .	2-6
2.5 Formatting the Report Page . . . . .	2-7
2.5.1 Default Page Width and Length. . . . .	2-7
2.5.2 Option 1: Setting the Page Width. . . . .	2-7
2.5.3 Option 2: Setting the Page Length . . . . .	2-8
2.5.4 Option 3: Limiting the Total Lines or Pages in a Report. . . . .	2-9
2.6 Setting Up the Report Heading . . . . .	2-9
2.6.1 Default Format . . . . .	2-9
2.6.2 Option 1: Naming the Report . . . . .	2-10
2.6.3 Option 2: Assigning a Date. . . . .	2-11
2.6.4 Option 3: Suppressing a Date . . . . .	2-12
2.6.5 Option 4: Assigning a Page Number . . . . .	2-12
2.6.6 Option 5: Suppressing the Page Number . . . . .	2-12
2.6.7 Option 6: Suppressing the Date and Page Number. . . . .	2-13

2.7	Printing Detail Lines and Column Headers . . . . .	2-13
2.7.1	Content of the Detail Line . . . . .	2-14
2.7.1.1	Field Values . . . . .	2-14
2.7.1.2	Value Expressions . . . . .	2-14
2.7.2	Format of Fields in the Detail Lines . . . . .	2-16
2.7.2.1	Order of Print Items . . . . .	2-16
2.7.2.2	Column Position of Print Items . . . . .	2-17
2.7.2.3	Edit String Format of Print Items. . . . .	2-18
2.7.3	Column Headers for Print Items . . . . .	2-19
2.7.3.1	Option 1: Suppressing Column Headers . . . . .	2-20
2.7.3.2	Option 2: Specifying a Column Header . . . . .	2-20
2.8	Summarizing Data . . . . .	2-22
2.8.1	COUNT, AVERAGE, and TOTAL . . . . .	2-22
2.8.2	Maximum Value, Minimum Value, and Standard Deviation . . . . .	2-24

### 3 Mastering Report Writing Techniques

3.1	Dividing Data Records into Groups . . . . .	3-1
3.1.1	Developing Control Groups with a Sort Key . . . . .	3-2
3.1.2	Developing Levels of Control Groups Using Multiple Sort Keys. . . . .	3-5
3.1.3	Reporting Group Summaries Only . . . . .	3-8
3.1.3.1	Summaries Through the Report Writer . . . . .	3-8
3.1.3.2	Summaries by the SUM Statement Outside of the Report Writer . . . . .	3-9
3.2	Reporting Data Grouped by Date . . . . .	3-10
3.2.1	Solution 1: Control Groups Based on Dates . . . . .	3-11
3.2.2	Solution 2: Grouping Records with a Common Month and Year . . . . .	3-13
3.3	Reporting Data from More Than One Record Source. . . . .	3-15
3.3.1	Using CROSS to Report Data from Two Domains. . . . .	3-15
3.3.2	Using Inner Print Lists to Compare Budgets with Expenses . . . . .	3-17
3.3.3	Using Inner Print Lists with a Hierarchical View. . . . .	3-20
3.4	Printing a Title Page and Other Special Headings . . . . .	3-21
3.4.1	Printing a Title Page . . . . .	3-21
3.4.2	Printing Special Page Headings. . . . .	3-24
3.5	Printing Row Totals . . . . .	3-26
3.6	Reporting Hierarchical Records . . . . .	3-28
3.6.1	Using Nested FOR Statements to Flatten the Hierarchy. . . . .	3-29
3.6.2	Using CROSS to Flatten the Hierarchy . . . . .	3-30
3.6.3	Accessing List Items with the SET SEARCH Command. . . . .	3-31
3.6.4	Using the REPORT Statement to Report List Data. . . . .	3-32
3.7	Printing a Variety of Detail Lines in One Report. . . . .	3-33
3.7.1	CHOICE Value Expression in COMPUTED BY Fields. . . . .	3-34
3.7.2	CHOICE Value Expression Within a PRINT Statement . . . . .	3-38

## 4 Writing DBMS Reports

4.1	Accessing VAX DBMS Data with DATATRIEVE . . . . .	4-1
4.2	Writing a Simple Report with VAX DBMS Data . . . . .	4-2
4.3	Developing a Procedure for a Bill-of-Materials Report . . . . .	4-4
4.4	Using Control Groups with VAX DBMS Data . . . . .	4-6

## 5 Writing Relational Database Reports

5.1	Accessing VAX Relational Databases with DATATRIEVE . . . . .	5-1
5.2	Writing a Simple Report with Relational Data. . . . .	5-4
5.3	Combining Data from Many Relations in a Single Report . . . . .	5-6
5.3.1	Using View Domains to Combine Data from Many Relations . . . . .	5-7
5.3.2	Using the CROSS and OVER Clauses to Combine Data for a Report. . . . .	5-11
5.4	Developing a Procedure for a Report . . . . .	5-15

## 6 DATATRIEVE Report Writer Reference Section

6.1	AT BOTTOM Statement . . . . .	6-3
6.2	AT TOP Statement. . . . .	6-7
6.3	END_REPORT Statement. . . . .	6-12
6.4	PRINT Statement (RW). . . . .	6-13
6.5	REPORT Statement. . . . .	6-19
6.6	SET Statement (RW) . . . . .	6-21

## A Listing of Record Definitions

A.1	CATS_REC Record Definition. . . . .	A-1
A.2	CHECKS_REC Record Definition . . . . .	A-1
A.3	EMP_REC Record Definition . . . . .	A-2
A.4	EMP_REVIEW_REC Record Definition. . . . .	A-2
A.5	PAYABLES_REC Record Definition . . . . .	A-2
A.6	PERSONNEL_REC Record Definition . . . . .	A-3
A.7	SALES_REC Record Definitions . . . . .	A-3
A.7.1	SALES_REC Record Definition (Original) . . . . .	A-3
A.7.2	SALES_REC Record Definition (Expanded). . . . .	A-4
A.8	WAGES_REC Record Definition . . . . .	A-4
A.9	YACHT Record Definition . . . . .	A-5

## Index



## Figures

2-1	Field Structure of PAYABLES_REC . . . . .	2-15
2-2	Field Structure of EMP_REVIEW_REC . . . . .	2-18
2-3	Field Structure of SALES_REC . . . . .	2-22
3-1	Field Structure of PERSONNEL_REC. . . . .	3-2
3-2	Control Group Report Based on One Sort Key. . . . .	3-5
3-3	Control Group Report Based on Two Sort Keys . . . . .	3-7
3-4	Field Structure of PAYABLES_REC . . . . .	3-10
3-5	Report with Detail Lines Grouped by Date . . . . .	3-12
3-6	Accounts Payable Report by Month . . . . .	3-14
3-7	Sample Title Page for a Report . . . . .	3-23
3-8	Field Structure of WAGES_REC . . . . .	3-26
3-9	Field Structure of EMP_REC . . . . .	3-28
3-10	Field Structure of SALES_REC . . . . .	3-33
3-11	Revised Field Structure of SALES_REC . . . . .	3-35
3-12	Control Group Report with Variety of Detail Lines . . . . .	3-37
5-1	Sample Relational Database . . . . .	5-3

## Tables

3-1	Commission Schedule for the Sales Division. . . . .	3-33
6-1	Summary of Report Writer Statements . . . . .	6-2
6-2	AT BOTTOM Statement Summary Elements . . . . .	6-3
6-3	AT TOP Statement Header and Summary Elements . . . . .	6-8
6-4	Report Parameters Controlled by Print List Elements. . . . .	6-14
6-5	Report Parameters Affected by Print List Modifiers . . . . .	6-15
6-6	Output File Specification Defaults. . . . .	6-20
6-7	SET Statement Arguments . . . . .	6-23

## How to Use This Manual

This manual describes how to write reports with VAX DATATRIEVE. It also provides reference information for the statements you use in working with the DATATRIEVE Report Writer.

### Intended Audience

You should use this manual if you want to produce reports using data managed by VAX DATATRIEVE. To fully understand the contents of this manual, you should be familiar with the material covered in the *VAX DATATRIEVE Handbook*.

For more advanced topics in this manual (especially in the chapter on mastering report writing techniques), you should be familiar with the material in the *VAX DATATRIEVE User's Guide* about record definitions, tables, record selection expressions, the CROSS clause, hierarchical records, variables, procedures, and command files. For the material in the chapters on writing DBMS and relational database reports, you should be familiar with the material on using DBMS and relational databases in the *VAX DATATRIEVE User's Guide*. In addition, you need to refer to the section on the EDIT\_STRING clause in the *VAX DATATRIEVE Reference Manual*.

### Operating System Information

Information about the versions of the operating system and related software that are compatible with this version of VAX DATATRIEVE is included in the VAX DATATRIEVE media kit, in either the Installation Guide or the Before You Install letter.

Contact your DIGITAL representative if you have questions about the compatibility of other software products with this version of VAX DATATRIEVE. You can request the most recent copy of the *VAX System Software Order Table/Optional*

*Software Cross Reference Table*, SPD 28.98.xx, which will verify which versions of your operating system are compatible with this version of VAX DATATRIEVE.

## Structure

This manual consists of six chapters, including a reference section, an appendix, and an index:

- |            |   |
|------------|---|
| Chapter 1  | Introduces the Report Writer and provides examples of simple reports.   |
| Chapter 2  | Describes how to develop a report specification, which contains the instructions for the content and format of a report. The chapter presents the Report Writer's default and optional settings.  |
| Chapter 3  | Discusses more advanced techniques of report writing with VAX DATATRIEVE. It covers such topics as reporting groups of sorted records, reporting data from several domains, and including several types of detail lines in the same report. |
| Chapter 4  | Shows how to produce reports with data managed by VAX DBMS.   |
| Chapter 5  | Shows how to produce reports with data managed by one of the DIGITAL relational database products: Rdb/VMS, Rdb/ELN, or VIDA.   |
| Chapter 6  | Contains complete reference information about the DATATRIEVE statements used in the Report Writer.  |
| Appendix A | Contains listings of the record definitions used in the examples in this book.  |

## Related Documents

For further information on the topics covered in this manual, you can refer to:

- *VAX DATATRIEVE Release Notes*  
Includes specific information about the current DATATRIEVE release and contains material added too late for publication in the other DATATRIEVE documentation.

- *VAX DATATRIEVE Installation Guide*

Describes the installation procedure for VAX DATATRIEVE. The manual also explains how to run User Environment Test Packages (UETPs), which test DATATRIEVE product interfaces, such as the interface between DATATRIEVE and Rdb/VMS.

- *VAX DATATRIEVE Handbook*

Describes how to create DATATRIEVE applications. The manual includes some tutorial information on describing data and creating procedures.

- *VAX DATATRIEVE Guide to Using Graphics*

Introduces the use of DATATRIEVE graphics and contains the reference material for creating DATATRIEVE plots.

- *VAX DATATRIEVE User's Guide*

Describes how to use DATATRIEVE interactively. The manual includes information on using DATATRIEVE to manipulate data and on using DATATRIEVE with forms, relational databases, and database management systems. It also describes how to improve performance and how to work with remote data.

- *VAX DATATRIEVE Reference Manual*

Contains reference information for DATATRIEVE.

- *VAX DATATRIEVE Pocket Guide*

Contains quick-reference information about using DATATRIEVE.

- *VAX DATATRIEVE Guide to Programming and Customizing*

Explains how to use the DATATRIEVE Call Interface. The manual also describes how to create user-defined keywords and user-defined functions to customize DATATRIEVE and how to customize DATATRIEVE help and message texts.

## Conventions

This section explains the conventions for the syntax and symbols used in this manual:

### WORD

An uppercase word in a syntax format is a keyword. You must include it in the statement if the clause is used.



word	A lowercase word in a syntax format indicates a syntax element that you supply.
[ ]	Square brackets enclose optional clauses from which you can choose one or none.
{ }	Braces enclose clauses from which you must choose one alternative.
<RET>	This symbol indicates the RETURN key. Unless otherwise indicated, end all user input lines in examples by pressing the RETURN key.
<TAB>	This symbol indicates the TAB key.
<CTRL/x>	This symbol tells you to press the CTRL (control) key and hold it down while pressing a letter key. If you press CTRL/Z, the word Exit appears in reverse video; if you press CTRL/Y, the word Interrupt appears in reverse video. Examples of video output in this book do not include either word; instead the conventions ^Z and ^Y are used.
<GOLD-x>	This symbol indicates that you press the GOLD key and then a specified letter key consecutively.
" "	These are double quotation marks.
' '	These are single quotation marks.
...	A horizontal ellipsis in syntax formats means you can repeat the previous item.  A horizontal ellipsis in examples means that information not directly related to the example has been omitted.
.	
.	
.	A vertical ellipsis in syntax formats means you can repeat the syntax element from the preceding line.  A vertical ellipsis in examples means that information not directly related to the example has been omitted.
Color	Color in examples shows user input.

Since CDD Version 3.1, CDD path names include a leading underscore. For example:

```
DTR> SHOW DICTIONARY
The default directory is _CDD$TOP.DTR32.WEAGER
```

Examples of the output in DATATRIEVE manuals do not reflect this change. You do not need to enter CDD path names with the leading underscore.

## References to Products

VAX DATATRIEVE is a member of the VAX Information Architecture, a group of products that work with each other and with VAX languages conforming to the VAX calling standard to provide flexible solutions for information management problems.

VAX Information Architecture documentation explaining how these products interrelate is included with the VAX CDD documentation. VAX Information Architecture documentation is also available separately. Contact your DIGITAL representative.

The VAX DATATRIEVE documentation to which this manual belongs often refers to products that are part of the VAX Information Architecture by their abbreviated names:

- VAX CDD software is referred to as CDD.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX DBMS software is referred to as DBMS.
- VAX Rdb/ELN software is referred to as Rdb/ELN.
- VAX Rdb/VMS software is referred to as Rdb/VMS.
- VAX TDMS software is referred to as TDMS.
- VIDA software is referred to as VIDA.

This manual uses the terms relational database or relational source to refer to all three of these products:

- VAX Rdb/ELN
- VAX Rdb/VMS
- VIDA



# Getting Started in Report Writing 1

## 1.1 Using the DATATRIEVE Report Writer

VAX DATATRIEVE allows you to organize and maintain data conveniently. A major reason for keeping this data is to make it available to the people who need it. The DATATRIEVE Report Writer helps you present this information in attractive and comprehensive reports.

Managers, secretaries, and other people often need information immediately on a specific subject. To report this information, you must have rapid access to the data and reliable formatting techniques. With a few simple statements and commands, you can quickly display and accurately summarize data managed by VAX DATATRIEVE. You can also use the DATATRIEVE Report Writer to report on data stored in a VAX DBMS, VAX Rdb/VMS, VAX Rdb/ELN, or VIDA (VAX-IBM Data Access) database.

In addition to query reports, most organizations require detailed summary reports at regular intervals to compare current performance with past performance. These periodic reports are on subjects such as accounts receivable, inventory, and sales. You can use a number of statistical functions within the Report Writer to summarize the information. Then you can define DATATRIEVE procedures to produce these reports whenever they are needed.

The DIGITAL product line also includes VAX DECreporter, a menu-driven report generator. DECreporter offers many of the same features as the DATATRIEVE Report Writer to produce simple reports. You can use DECreporter to create DATATRIEVE procedures and store them in your CDD dictionary. You can later edit the DECreporter generated procedures in DATATRIEVE using the editor you use with DATATRIEVE.



This manual teaches you by example how to use the DATATRIEVE Report Writer to generate many types of reports, ranging from simple ad hoc queries to more complex summary reports. Many examples use the sample domains YACHTS, PERSONNEL, PAYABLES, and SALES, which are loaded during the installation procedure. The following command sets the CDD default to the directory that contains the domain definitions used in this book:

```
DTR> SET DICTIONARY CDD$TOP.DTR$LIB.DEMO
```

If you cannot access these domains, see the person responsible for VAX DATATRIEVE at your site. Note that definitions for DBMS or relational database examples are included in special subdictionaries of CDD\$TOP.DTR\$LIB.DEMO. For more information, see the chapters on using the Report Writer with DBMS and relational databases.

## 1.2 What the Report Writer Can Do

The Report Writer helps you organize your data in a clear, readable format. It can:

1. Center a report name at the top of the page
2. Print the current date at the upper right
3. Print page numbers at the upper right
4. Set up column headings
5. Print a detail line for each record
6. Print a summary line for:
  - a. A group of data (yachts by the same builder)
  - b. The entire report (yachts by several builders)

The following report was produced with the Report Writer. Each number corresponds to one of the features in the previous list:

(1) YACHTS BY ALBERG, ALBIN, AND AMERICAN				(2) 30-Apr-1984 Page 1 (3)		
(4) MANUFACTURER	MODEL	RIG	LENGTH OVER ALL	WEIGHT	BEAM	PRICE
ALBERG	37 MK II	KETCH	37	20,000	12	\$36,951 (5)
BOAT COUNT: 1				AVERAGE PRICE:		\$36,951 (6a)
*****						
ALBIN	79	SLOOP	26	4,200	09	\$17,900
ALBIN	VEGA	SLOOP	27	5,070	09	\$18,600
ALBIN	BALLAD	SLOOP	30	7,276	09	\$27,500
BOAT COUNT: 3				AVERAGE PRICE:		\$21,333
*****						
AMERICAN	26	SLOOP	26	4,000	08	\$9,895
AMERICAN	26-MS	MS	26	5,500	08	\$18,895
BOAT COUNT: 2				AVERAGE PRICE:		\$14,395
*****						
TOTAL BOAT COUNT: 6				OVERALL AVERAGE PRICE:		\$21,624 (6b)

### 1.3 Writing Simple Reports: Examples

DATATRIEVE enables you to produce reports in a number of ways:

- With the PRINT statement to display data from one group of records
- With the SUM statement to generate summary totals from groups of records
- With the Report Writer, which provides the most flexibility for designing the format and content of reports

The following sections illustrate these three methods of report writing in DATATRIEVE. You can use the examples as models for creating similar reports with your own data.

### 1.3.1 The PRINT Statement

When you display records with the PRINT statement outside of the Report Writer, you are creating a simple report. To display the YACHTS built by ALBIN, type:

```
DTR> READY YACHTS
DTR> PRINT YACHTS WITH BUILDER = "ALBIN"
```

MANUFACTURER	MODEL	RIG	LENGTH	WEIGHT	BEAM	PRICE
			OVER			
			ALL			
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

### 1.3.2 Summarizing Data with the SUM Statement

You may want to summarize information about selected groups within your report. The simplest way is to form a collection and use the DATATRIEVE SUM statement.

Assume that you have a collection of seven records from the YACHTS domain. When you use the number 1 as one of the print list items following the SUM statement, as in the following example, DATATRIEVE counts and totals the number of yachts for each builder. When a numeric field such as PRICE is used as a print list item after the SUM statement, DATATRIEVE totals and displays the price of all the yachts for each builder and for all the builders together. In the following example, DATATRIEVE calculated the price of all the yachts for each builder and for all the builders together. The expression ("NUMBER") causes the word NUMBER to be printed as a column header above the columns that represent the total number of yachts by builder and the overall total number of yachts.

BY BUILDER indicates a count of yachts according to builder. It is an implicit SORTED BY statement and sorts the seven records alphabetically by builder.

```
DTR> READY YACHTS; FIND FIRST 7 YACHTS
[7 records found]
DTR> SUM 1 ("NUMBER"),
[Looking for next element in list]
CON> PRICE USING $$$$,$$$ BY BUILDER
```

MANUFACTURER NUMBER		TOTAL PRICE	NUMBER
ALBERG	1	\$36,951	
ALBIN	3	\$64,000	
AMERICAN	2	\$28,790	
BAYFIELD	1	\$32,875	
			7
		\$162,616	

DTR)

The report displays four builders and summarizes data for each builder individually and for all the builders as a group.

### 1.3.3 Writing a Simple Report with the Report Writer

The PRINT and SUM statements give you some control over the display of your data. But the Report Writer can do more formatting for you than the PRINT statement alone can. The simplest reports can be produced with only a few statements:

- A REPORT statement to invoke the Report Writer and to identify the data that you wish to report.
- One or more SET statements to name the report and control the format. (SET statements are optional. If you prefer, the Report Writer can do all the formatting for you.)
- A PRINT statement to indicate which fields from the records are to be displayed.
- An END\_REPORT statement at the end.

Note that the Report Writer PRINT statement is different than the regular PRINT statement in DATARETRIEVE. In the Report Writer, you list the fields or value expressions that you want to display. You cannot say PRINT CURRENT or PRINT YACHTS, because CURRENT and YACHTS are not field names or value expressions.



To find out the field names and query names in the record definition, use the SHOW FIELDS command:

```
DTR> SHOW FIELDS FOR YACHTS
YACHTS
  BOAT
    TYPE      (Primary key)
    MANUFACTURER (BUILDER) (Character string, primary key)
    MODEL (Character string, alternate key)
  SPECIFICATIONS (SPECS)
    RIG (Character string)
    LENGTH_OVER_ALL (LOA) (Character string)
    DISPLACEMENT (DISP) (Number)
    BEAM (Number)
    PRICE (Number)
```

Note that BOAT is the top-level field for the records in the YACHTS domain. BOAT includes all of the elementary fields in YACHTS. If you want to report all the data on each yacht, specify BOAT in the PRINT statement.

The following example shows how to create a report on all the yachts manufactured by Albin. The name of the report is "YACHTS BY ALBIN." In this example, the date and page number are included as part of the report heading by default.

To produce the report, follow these steps:

1. Report each record in YACHTS where the builder is ALBIN. The REPORT statement identifies the data and invokes the Report Writer.
2. Name the report.
3. Print the top-level field BOAT for each record. Using the BOAT field lets you display all the data on each yacht.
4. End with an END\_REPORT statement.

The following DATATRIEVE session illustrates these steps. The RW > prompt shows that DATATRIEVE is ready to accept the Report Writer statements. The number after each statement corresponds to one of the steps:

```
DTR> REPORT YACHTS WITH BUILDER = "ALBIN" _____(1)
RW> SET REPORT_NAME = "YACHTS BY ALBIN" _____(2)
RW> PRINT BOAT _____(3)
RW> END_REPORT _____(4)
```

MANUFACTURER	MODEL	RIG	LENGTH	WEIGHT	BEAM	PRICE
			OVER ALL			
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

DTR)

Note the way the Report Writer sets up the page format. Each page has the date and page number in the upper right corner. In this example, the Report Writer automatically sets the page width at 80 columns per page, spacing the fields of each record for you.

### 1.3.4 Defining a Procedure to Produce a Report

When you need to produce the same report periodically, it is helpful to enclose the Report Writer statements within a procedure.

You can customize the report by using one or more prompting value expressions or prompts. The prompt consists of an asterisk and a period, followed by a character string enclosed in quotation marks. For example, you could include the statement:

```
SET REPORT_NAME = *."the report name enclosed in quotation marks"
```

After the user invokes the procedure, the terminal displays the following prompt:

```
Enter the report name enclosed in quotation marks:
```

DATATRIEVE does not finish processing the report until the user enters a report name.

You can also include prompts for page width and length, maximum report size, output device, and other features relating to the report. See the chapter on designing a report with the Report Writer for more details.

The following example shows how to define a procedure YACHT\_\_PER\_\_LB to produce a report on all yachts by a selected builder. The report includes columns for these fields: MANUFACTURER, MODEL, DISP ("WEIGHT"), and PRICE. In addition, there is a special column indicating the price per pound of each yacht. Note that the procedure is set up to allow the user to select at the time of execution: the builder, output device, name of the report, page width, page length, and the maximum number of pages in the report.

Follow these steps:

1. Use DEFINE PROCEDURE to define a procedure YACHT\_PER\_LB.
2. Ready the domain YACHTS.
3. Identify the data you wish to report within the REPORT statement. Include prompts so that the user can select the builder's name and the output device.
4. Allow the user to name the report by including a prompt with the SET REPORT\_NAME statement. Indicate that the report name must be enclosed with quotation marks.
5. Control the page width with a SET COLUMNS\_PAGE statement. Use the prompt option.
6. Control the page length with a SET LINES\_PAGE statement. Use the prompt option.
7. Limit the overall length of the report with a SET MAX\_PAGES statement. Use the prompt option.
8. Specify the items in each detail line with a PRINT statement. These become the columns for the report. Create a column for price per pound by including PRICE/DISP as one of the items. (The slash indicates division.) The USING \$\$99 clause indicates that you want the price per pound to be printed as a monetary value and that you expect the price per pound to be less than \$10.00.
9. End with an END\_REPORT statement.
10. Clear your workspace with a FINISH command.
11. End the procedure with an END\_PROCEDURE statement.
12. To invoke the procedure, type:

```
DTR> :YACHT_PER_LB
```

The following DATATRIEVE statements produce the report. The number after each statement corresponds to one of the steps:

```
DTR> DEFINE PROCEDURE YACHT_PER_LB_____ (1)
DFN> READY YACHTS_____ (2)
DFN> REPORT YACHTS WITH BUILDER = *."the builder" ON *."device"_____ (3)
DFN> SET REPORT_NAME = *."report name enclosed in quotation marks"_____ (4)
DFN> SET COLUMNS_PAGE = *."the columns per page"_____ (5)
DFN> SET LINES_PAGE = *."the lines per page"_____ (6)
```

```

DFN> SET MAX_PAGES = *."the maximum pages for the report"_____ (7)
DFN> PRINT TYPE, DISP, PRICE, _____ (8)
DFN> PRICE/DISP ("PRICE/LB") USING $$ .99
DFN> END_REPORT _____ (9)
DFN> FINISH YACHTS _____ (10)
DFN> END_PROCEDURE _____ (11)
DTR> :YACHT_PER_LB _____ (12)
Enter the columns per page: 60
Enter the lines per page: 55
Enter the maximum pages for the report: 10
Enter report name enclosed in quotation marks: "YACHTS BY AMERICAN"
Enter the builder: AMERICAN
Enter device: TT:

```

```

                YACHTS BY AMERICAN                17-Apr-1984
                                                Page 1

```

MANUFACTURER	MODEL	WEIGHT	PRICE	PRICE/LB
AMERICAN	26	4,000	\$9,895	\$2.47
AMERICAN	26-MS	5,500	\$18,895	\$3.44

### 1.3.5 Printing Summary Statistics with the Report Writer

If you want summary information on the data records in the report, use the Report Writer's statistical functions to compute values for a summary line. The functions are:

- AVERAGE
- COUNT
- Maximum value (MAX)
- Minimum value (MIN)
- RUNNING COUNT
- RUNNING TOTAL
- Standard deviation (STD\_\_DEV)
- TOTAL

The following example shows a report on yachts built by Albin. The total number of yachts and the average price of a yacht are displayed at the bottom of the report. The page width is limited to 70 columns.



Follow these steps:

1. Identify the data you wish to report within the REPORT statement.
2. Name the report.
3. Set the page width.
4. Print the top-level field BOAT in each detail line.
5. Summarize the report with an AT BOTTOM OF REPORT statement. Use COUNT for the total number of boats. To suppress the header "COUNT," include a hyphen in parentheses after COUNT. Then apply AVERAGE to the field PRICE for the average price.
6. End with an END\_REPORT statement.

The following group of statements produces the desired summary report. The number after each statement corresponds to one of the steps:

```
DTR> REPORT YACHTS WITH BUILDER = "ALBIN" _____(1)
RW> SET REPORT_NAME = "YACHTS BY ALBIN" _____(2)
RW> SET COLUMNS_PAGE = 70 _____(3)
RW> PRINT BOAT _____(4)
RW> AT BOTTOM OF REPORT PRINT SKIP, COL 10, _____(5)
RW>   "BOAT COUNT:", SPACE, COUNT (-) USING Z9,
RW>   COL 45, "AVERAGE PRICE:", AVERAGE PRICE
RW> END_REPORT _____(6)
```

```
                YACHTS BY ALBIN                16-Apr-1984
                                                Page 1
```

MANUFACTURER	MODEL	RIG	LENGTH OVER ALL	WEIGHT	BEAM	PRICE
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
BOAT COUNT: 3			AVERAGE PRICE:			\$21,333

DTR>

Note that the Report Writer has lined up the value for the average price in the PRICE column. However, you had to specify the placement of the count, its text, and the average price text.

These sample reports demonstrate only a few applications of the DATATRIEVE Report Writer. The next two chapters discuss and illustrate other features to control the format and content of reports.

## Designing a Report with the Report Writer **2**

You create a DATATRIEVE report with a series of Report Writer statements called a **report specification**. A report specification controls the format and determines the content of a report. Some types of statements are required for a valid report specification, and others are optional:

- A report specification must begin with a REPORT statement, in which you can specify the data for the report and the file or device to which DATATRIEVE writes the report.
- A report specification may contain SET statements to control page format and assign column headings. The Report Writer uses its built-in default assignments for the SET statements you do not include.
- A report specification can include a single PRINT statement in a report specification as well as several AT statements. The PRINT statement indicates detail lines to be printed in the report. AT statements indicate summary or header lines.
- A report specification must end with an END\_REPORT statement.

Each major section of this chapter discusses a part of the report specification. In each section, the default format built into the Report Writer is described first, followed by descriptions of your options to change the format. If you are satisfied with the default format, skip the sections labeled “option” that immediately follow. Otherwise, you can incorporate one or more of the options into your report specification.

For example, the Report Writer automatically sets the page width at 80 columns per page. This is the default format. If this page width is satisfactory, you need not take any action to change it. However, you can include an optional statement (SET COLUMNS\_PAGE) to specify a different page width.

Take a piece of paper to make a rough sketch of a report that you would like to produce based on data in DATATRIEVE domains. Indicate what information you would like the report to contain and the basic format for each page. Then read each section of this chapter to develop a report specification that produces such a report, using the instructions and examples provided as aids.

## 2.1 Identifying the Data and Invoking the Report Writer

Reports usually highlight only a portion of the available information. To report specific data, you must identify it to DATATRIEVE and invoke the Report Writer. Follow these steps:

1. Ready the domain(s) containing the data you wish to report:

```
DTR> READY YACHTS
DTR>
```

2. Identify the data within the domain on which you will report. This enables you to limit the number of records in the report and to sort the records if you desire. You can identify the data in one of two ways: with a FIND statement or with a record selection expression (RSE) in a REPORT statement. As a general rule, use a FIND statement when you need the collection of data for other purposes during a DATATRIEVE session. However, if you need the set of data only for the report, identify the data with an RSE in the REPORT statement.

This example shows a FIND statement that forms a sorted collection from the YACHTS domain:

```
DTR> FIND YACHTS WITH LOA > 40 SORTED BY BEAM
[8 records found]
DTR>
```

3. Enter the REPORT statement to invoke the Report Writer. The format of the REPORT statement is:

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

The following are valid REPORT statements:

- REPORT – When you omit the RSE from the REPORT statement, the Report Writer reports on the records in the CURRENT collection and writes the report on your terminal:

```
DTR> REPORT
RW>
```

- REPORT ON file-spec – The following REPORT statement asks the Report Writer to report on the CURRENT collection and to write the report to a file named BIGYAT.LIS:

```
DTR> REPORT ON BIGYAT
RW>
```

- REPORT rse – If you did not form a collection with a FIND statement, you must identify a record stream with the REPORT statement, for example:

```
DTR> REPORT YACHTS WITH LOA > 40 SORTED BY BEAM
RW>
```

## 2.2 Exiting from the Report Writer

You invoke the Report Writer with a REPORT statement, and you normally exit with an END\_REPORT statement. The following example represents such a report:

```
DTR> REPORT YACHTS WITH BUILDER = "ALBIN"
RW> PRINT BOAT
RW> END_REPORT
```

12-Apr-1987  
Page 1

MANUFACTURER	MODEL	RIG	LENGTH		WEIGHT	BEAM	PRICE
			ALL	OVER			
ALBIN	79	SLOOP	26		4,200	09	\$15,000
ALBIN	BALLAD	SLOOP	30		7,276	09	\$27,500
ALBIN	VEGA	SLOOP	27		5,070	09	\$18,600

DATATRIEVE processes the report specification and produces the report if there are no errors. If the report specification contains any prompts, the Report Writer prompts you for the needed information before producing the report.

If DATATRIEVE detects an error in your Report Writer statements, it displays an error message on your terminal and returns you to DATATRIEVE command level.



To force an exit from the Report Writer and return to DATATRIEVE command level, you can enter CTRL/C as a response to a RW> prompt or in the middle of an input line:

```
DTR> READY YACHTS
DTR> REPORT YACHTS
RW> <CTRL/C>
Execution terminated by operator
DTR>
```

## 2.3 Using an Editor to Correct Mistakes

If you make a typing error or a syntax error, DATATRIEVE may display an error message on your terminal and return you to DATATRIEVE command level, indicated by the DTR> prompt. Type EDIT and press RETURN. DATATRIEVE copies all of the REPORT statement into the main buffer of the editor. Now you can edit and modify your report specification with the editor. The DTR> and RW> prompts do not appear on your terminal when you use an editor.

When you finish modifying the report specification, exit from the editor by pressing CTRL/Z, typing EXIT, and then pressing RETURN. DATATRIEVE displays on your terminal either the report or an error message. In the case of another error message, enter the EDIT command and begin the editing cycle again.

If DATATRIEVE displays the report but you want to change the specification again, enter the EDIT command to resume editing. For more information on editing, see the *VAX DATATRIEVE Handbook*, as well as the documentation for the VMS editor you are using.

When DATATRIEVE displays the report you want, you can save the report specification in either of two ways. Call the editor and do one of the following:

- Enter DEFINE PROCEDURE before the REPORT statement and enter END\_\_PROCEDURE after the END\_\_REPORT statement to form a procedure stored in the CDD.
- Use the editor's WRITE command to copy the specification into a file.

## 2.4 Selecting the Output Medium

The REPORT statement enables you to select the device or file that is to receive the report. If you use the prompt option, you can choose the output medium immediately before the report is produced.

### 2.4.1 Default: Output to Terminal

If you begin your report specification with a REPORT RSE statement, the report is displayed on your terminal. Such a statement is equal to entering REPORT [rse] ON TT: explicitly.

### 2.4.2 Option 1: Output to a Line Printer

To get a copy of your report from a line printer, type the system name of the line printer at the end of the REPORT statement. Note that for the following example to work properly, LP: must be defined as the system printer:

```
DTR> REPORT ON LP:
```

If the REPORT command contains an RSE, put the output specification after the RSE:

```
DTR> REPORT YACHTS WITH BUILDER CONT "PEARSON" ON LP:
```

### 2.4.3 Option 2: Output to a File

If you want to store your report in a file, end the REPORT statement with ON followed by a file specification. This example creates a file called REPORT.TXT in your default VMS directory:

```
DTR> REPORT BIG_ONES ON REPORT.TXT
```

You can also write a report to a file in any VMS directory to which you have W (write) access, for example:

```
DTR> REPORT YACHTS WITH RIG = "MS" ON DB0:[MORRISON.RPTS]SEP3.REP
```

If you send your report to a file, you can use DCL PRINT commands to print a hard copy at your convenience. You can also make the report available for display on other terminals or send it across a network link to people on other computers at other sites.

### 2.4.4 Option 3: Prompt

If you want to choose the output medium at the time the report is processed, use the prompt option. DATATRIEVE prompts you for the output medium when it is processing the report:

```
DTR> READY PAYABLES
DTR> REPORT FIRST 2 PAYABLES ON *."the file specification"
RW> PRINT PAYABLES_REC
RW> END_REPORT
Enter the file specification: PAY.LIS
DTR>
```

In this case, the report is placed in the file PAY.LIS in the default VMS directory.

### 2.4.5 Using an ON Statement to Specify Multiple Output Media

If you want to write the report to your terminal and to a file, you can use the ON statement in conjunction with the Report Writer, for example:

```
DTR> READY PAYABLES
DTR> ON PAY.LIS
CON> REPORT PAYABLES ON TT:
RW> PRINT INVOICE_DUE
RW> END_REPORT
```

The report appears on your terminal and also is written to the file DB0:[MORRISON.RPTS]PAY.LIS;6.

If you want the report written to more than two output media, you can include a nested ON statement before you invoke the Report Writer, for example:

```
DTR> READY PAYABLES
DTR> ON [MORRISON]PAY1.LIS
CON> ON [MORRISON.RPTS]PAY1.LIS
CON> REPORT PAYABLES ON TT:
RW> PRINT INVOICE_DUE
RW> END_REPORT
```

The report appears on your terminal and also is written to the two files DB0:[MORRISON]PAY1.LIS;1 and DB0:[MORRISON.RPTS]PAY1.LIS;1.

Note, also, that you must use the ON statement instead of the ON clause if you wish to produce multiple reports as a result of multiple iterations of a loop. If you use the ON clause within a FOR or WHILE loop, the output file is not closed until the loop is completed. In such a case only a single report containing all the data will be produced.



## 2.5 Formatting the Report Page

One of the main advantages of the Report Writer is its ease of formatting. You can let the Report Writer use the default format. Or you can include SET statements to specify the number of columns and lines per page and the maximum number of lines or pages in a report.

### 2.5.1 Default Page Width and Length

The default format is:

- Page width: 80 columns
- Page length: 60 lines

### 2.5.2 Option 1: Setting the Page Width

If you do not want the default format of 80 columns per page, you can specify the number of columns by including a SET COLUMNS\_PAGE statement within your report specification. The statement has the following form:

$$\text{SET COLUMNS\_PAGE} = \left\{ \begin{array}{l} n \\ *.prompt \end{array} \right\}$$

The value *n* is the number of columns per page. The maximum value for *n* is 255.

You can use this command at DTR command level or in the Report Writer to set the page width for reports. Entering SET COLUMNS\_PAGE in the Report Writer does not affect the page width when you return to the DTR > prompt.

If you make the width too small, the Report Writer breaks lines and wraps the fields that do not fit to the next line of the display. In considering the setting for page width, take into account the number of columns used to display the same fields outside the Report Writer with the regular PRINT statement. If you request totals and other special value expressions on the detail line, you must add in the additional columns required for these print objects. See the section on the PRINT statement (RW) in the reference chapter of this manual for more information.

To set the page width at 60 columns, enter:

```
RW> SET COLUMNS_PAGE = 60
```

If you specify a width that is greater than 80, it is advisable to adjust the column width of your terminal screen. For VT100- or VT200-family terminals, use `FN$WIDTH(n)` at DTR command level to specify the number of columns that can be shown across the terminal screen. For this function, `n` can be as large as 132. For example:

```
DTR> FN$WIDTH(132)
```

You must also include the following, either at DTR command level or within the Report Writer:

```
DTR> SET COLUMNS_PAGE = 132
```

If you use the prompt option, you can choose the page width when the report is processed.

### 2.5.3 Option 2: Setting the Page Length

If you do not want the default format of 60 lines per page, you can include a `SET LINES_PAGE` statement within your report specification. The statement has the following form, where `n` is an integer representing the number of detail lines per page:

$$\text{SET LINES\_PAGE} = \left\{ \begin{array}{l} n \\ *.prompt \end{array} \right\}$$

To set the page length at 55 lines, enter:

```
RW> SET LINES_PAGE = 55
```

If you use the prompt option, you can choose the page length at the time the report is processed.

If your report contains a list, each item in the list counts as a separate detail line.

If you want a report without any page breaks, give `n` a value larger than the number of lines in the report.

### 2.5.4 Option 3: Limiting the Total Lines or Pages in a Report

The Report Writer sets no automatic limit on either the total lines or total pages in a report. The SET MAX\_LINES or SET MAX\_PAGES statement can prevent an infinite loop from tying up your system's resources. These statements specify the maximum number of lines or pages that your report can contain. For SET MAX\_LINES, DATATRIEVE counts header lines, blank lines, detail lines, and summary lines. The two formats are similar:

```
SET MAX_LINES = { n  
                 *.prompt }
```

```
SET MAX_PAGES = { n  
                 *.prompt }
```

The Report Writer counts all the lines or pages of the report. When the limit you have specified is reached, the Report Writer stops producing the report and prints one of the following error messages:

```
Maximum line count exceeded - report terminated.
```

```
Maximum report pages exceeded - report terminated.
```

If you use the prompt option, you can select the maximum number of pages or lines at the time the report is processed. For example:

```
DTR> READY YACHTS  
DTR> REPORT YACHTS  
RW> PRINT BOAT  
RW> SET MAX_PAGES = *."the maximum number of pages"  
RW> END_REPORT  
Enter the maximum number of pages:
```

## 2.6 Setting Up the Report Heading

The report heading consists of the report name, date, and page number. You can let the Report Writer use the default format, or you can specify a report heading with SET statements.

### 2.6.1 Default Format

This is the default format for the report heading:

- Report Name – The Report Writer produces a report without a name unless you include a SET REPORT\_NAME statement.
- Date – The Report Writer prints the current system date in the upper right corner of the page. It uses the format DD-Mmm-YYYY (for example, 21-Jan-1987).



- Page number – The Report Writer prints the page number of the report directly under the date. Regardless of the lengths of the date string and the page number, the Report Writer aligns the first character of the page number under the first character of the date. It uses the format Page n. For example:

21-Jan-1987  
Page 2

## 2.6.2 Option 1: Naming the Report

Use the SET REPORT\_NAME statement to name your report. Be sure to enclose the name in quotation marks. The Report Writer centers this name on the first printed line at the top of the page. For example:

```
RW> SET REPORT_NAME =
RW> "ACCOUNTS PAYABLE FOR BOCK'S YACHTS"
RW> END_REPORT
DTR>
```

ACCOUNTS PAYABLE FOR BOCK'S YACHTS      21-Jan-1987  
Page 1

The SET REPORT\_NAME statement has the following syntax:

```
SET      REPORT_NAME =      { "string" [...] }
                                 { *.prompt      }
```

You are not limited to report names of one line. To produce a report name of two or more lines, enclose each segment of the report name in quotation marks and separate each segment from the next with a slash (/):

```
RW> SET REPORT_NAME =
RW> "ACCOUNTS PAYABLE"/"FOR BOCK'S YACHTS"
```

This statement produces the following report heading:

ACCOUNTS PAYABLE      18-Apr-1987  
FOR BOCK'S YACHTS      Page 1

If you specify a prompting value expression in the SET REPORT\_NAME statement, the Report Writer prompts you for a name when it is processing the report. In response to the prompt, enclose each segment with quotation marks, and separate each segment from the next with a slash:

```
DTR> REPORT PAYABLES
RW> SET REPORT_NAME = *."a Report Name"
RW> PRINT PAYABLE
RW> END_REPORT
Enter a Report Name: "ACCOUNTS PAYABLE"/"FOR BOCK'S YACHTS"
```

This statement produces the following report heading:

```
ACCOUNTS PAYABLE                21-Jan-1987
FOR BOCK'S YACHTS              Page 1
```

### 2.6.3 Option 2: Assigning a Date

The SET DATE statement enables you to replace the default date in the top right corner of the page with another date or any other character string:

```
RW> SET DATE = "22 MAR 1987"
```

This statement produces:

```
22 MAR 1987
Page 1
```

```
RW> SET DATE = "TUESDAY, MARCH 14TH"
```

This statement produces:

```
TUESDAY, MARCH 14TH
Page 1
```

```
RW> SET DATE = "COMPANY CONFIDENTIAL"
```

This statement produces:

```
COMPANY CONFIDENTIAL
Page 1
```



### 2.6.4 Option 3: Suppressing a Date

If you do not want a date on your report, include the SET NO DATE statement as part of your report specification. If you have included a one-line name for the report, the report name is printed on the same line as the page number.

### 2.6.5 Option 4: Assigning a Page Number

If you want the page number of your report to begin with a number other than the default number 1, use the SET NUMBER statement. The SET NUMBER statement has the following syntax:

```
SET NUMBER = { n  
              *.prompt }
```

For example:

```
RW> SET NUMBER = 3
```

This statement displays:

```
1-Nov-1986  
Page 3
```

When you use the prompt option, you choose the page number when your report is processed:

```
DTR> REPORT PAYABLES  
RW> SET NUMBER = *."the page number"  
RW> PRINT PAYABLES  
RW> END_REPORT  
Enter the page number: 4
```

This report specification produces the following heading:

```
1-Nov-1986  
Page 4
```

### 2.6.6 Option 5: Suppressing the Page Number

If you do not want any page numbers on your report, include the SET NO NUMBER statement as part of your report specification. Otherwise, the Report Writer automatically prints the page number in the upper right corner of the page.

## 2.6.7 Option 6: Suppressing the Date and Page Number

Use the `SET NO REPORT_HEADER` if you do not want either the date or the page number as part of your report.

## 2.7 Printing Detail Lines and Column Headers

Detail lines are the formatted lines of data in a report. Each detail line contains information about an individual record from a domain. The Report Writer arranges the data in columns, and the column headers indicate what the data items represent.

The Report Writer `PRINT` statement produces a detail line in the report for every record in the current collection or in the specified record stream. A detail line can cover several lines on the report page, depending on the content and format you specify. You can include only one `PRINT` statement in a report specification.

The `PRINT` statement has the following syntax:

```
PRINT print-list-element [...]
```

Print-list elements include field names, numeric and character-string literals, and arithmetic and statistical expressions. You can also specify horizontal and vertical spacing (`SPACE`, `TAB n`, `COL n`, `SKIP`, `NEW_PAGE`), as well as edit strings and column headers.

With the Report Writer `PRINT` statement, you can specify three characteristics of the detail lines:

- The content of the detail lines:
  - Values of fields from records identified either by a `FIND` or by a record selection expression in the `REPORT` statement
  - Value expressions
- The format of the print items in the detail lines:
  - Order of each print item
  - Column position of each print item
  - Edit string format for each print item
- The column headings for each print item

## 2.7.1 Content of the Detail Line

A detail line can have two types of print items. The first type is the value of a field from the record. One example is the value of the PRICE field from YACHTS. The second type is a value expression. Value expressions may be derived from field values, or they may be literals or variables. Some examples of value expressions derived from field values are PRICE/DISP (price per pound) or PRICE \* 1.1 (10% markup on price).

**2.7.1.1 Field Values** — You determine the content of the detail line by indicating which fields from the record should be printed. You can specify either elementary fields or group fields. In the case of a group field, each of its elementary fields is printed in a separate column.

The following example shows how to print the TYPE and PRICE fields of the YACHTS domain in the default format. The report displays the group field TYPE as two columns, one for each elementary field (MANUFACTURER and MODEL):

```
DTR> REPORT FIRST 2 YACHTS
RW> PRINT TYPE, PRICE
RW> SET COLUMNS_PAGE = 60
RW> END_REPORT
```

19-Apr-1987  
Page 1

MANUFACTURER	MODEL	PRICE
ALBERG	37 MK II	\$36.951
ALBIN	79	\$17.900

**2.7.1.2 Value Expressions** — You can create additional detail line items computed from other field values with arithmetic or statistical operators. In addition, you can include other value expressions such as literals or variables.

The following example displays the model, current price, and a new price 10% higher than the current price for the first five records in the YACHTS domain. It includes the literal "BARGAIN" on each detail line. It also specifies appropriate column headers:

```
DTR> REPORT FIRST 5 YACHTS
RW> SET COLUMNS_PAGE = 60
RW> SET REPORT_NAME = "BOCK'S YACHTS"/"PRICE LIST"
RW> PRINT TYPE, PRICE ("CURRENT"/"PRICE"),
RW>     PRICE * 1.1 ("SUGGESTED"/"PRICE") USING $$$,$$$,
RW>     "BARGAIN" ("COMMENT")
RW> END_REPORT
```

MANUFACTURER	MODEL	CURRENT PRICE	SUGGESTED PRICE	COMMENT
ALBERG	37 MK II	\$36,951	\$40,646	BARGAIN
ALBIN	79	\$17,900	\$19,690	BARGAIN
ALBIN	BALLAD	\$27,500	\$30,250	BARGAIN
ALBIN	VEGA	\$18,600	\$20,460	BARGAIN
AMERICAN	26	\$9,895	\$10,885	BARGAIN

The edit string clause, USING \$\$\$, specifies the output format for the PRICE \* 1.1 field. You need to use one more dollar sign than the maximum number of digits for the field value. See the section on edit string format for more information.

You can also maintain running statistics. It is possible to keep a running count of the detail lines or a running total of the values of selected fields.

Figure 2-1 shows the logical structure of the record PAYABLES\_REC for the PAYABLES domain. PAYABLES is used in the example that follows. (See Appendix A for the complete record definition.)

01 PAYABLE						
05 ORDER_ NUM	05 TYPE		05 ITEMS_ RECEIVED	05 INVOICE_ DUE	05 BILL_ PAID	05 WHSLE_ PRICE
	10 MANUFACTURER	10 MODEL				

MK-01120-00

**Figure 2-1: Field Structure of PAYABLES\_REC**

In the following example, Bock's Yachts has an accounts payable domain to collect data on unpaid bills. When the company orders goods, it stores a record indicating the manufacturer, the wholesale price of the goods, and the order number. It leaves the fields for INVOICE\_DUE and BILL\_PAID blank. (A MISSING VALUE has been defined in the record for these fields, along with a string that is printed when the value is missing.) When an invoice for the goods is received, the company modifies the record to indicate the invoice date. After Bock's Yachts pays the bill, it modifies the record to indicate the payment date.

Report all unpaid bills for which invoices have been received, sorted by the invoice date. Keep a running count of bills and a running total of the amount owed:

```
DTR> REPORT PAYABLES WITH BILL_PAID MISSING AND
RW>     ITEMS_RECEIVED NOT MISSING AND
RW>     INVOICE_DUE NOT MISSING SORTED BY INVOICE_DUE
RW> SET REPORT_NAME = "BOCK'S YACHTS"/"ACCOUNTS PAYABLE"
RW> SET COLUMNS_PAGE = 65
RW> PRINT RUNNING COUNT ("COUNT"), MANUFACTURER,
RW>     ITEMS_RECEIVED, INVOICE_DUE, BILL_PAID, WHSLE_PRICE,
RW>     COL 55, RUNNING TOTAL WHSLE_PRICE ("TOTAL"/"OWED") USING $$$,$$$
RW> END_REPORT
```

BOCK'S YACHTS  
ACCOUNTS PAYABLE

19-Apr-1983  
Page 1

COUNT	VENDOR	ITEMS RECEIVED	INVOICE DUE	BILL PAID	WHSLE PRICE	TOTAL OWED
1	CAPE DORY	2/15/83	12/30/82	NOT PAID	\$3,150	\$3,150
2	SALT	3/01/83	1/25/83	NOT PAID	\$4,850	\$8,000
3	AMERICAN	11/05/82	2/14/83	NOT PAID	\$9,000	\$17,000
4	ALBIN	6/21/82	2/15/83	NOT PAID	\$13,500	\$30,500
5	ALBIN	8/22/82	3/14/83	NOT PAID	\$23,850	\$54,350
6	BAYFIELD	8/04/82	4/01/83	NOT PAID	\$13,000	\$67,350
7	IRWIN	3/01/83	4/01/83	NOT PAID	\$29,999	\$97,349

Bock's Yachts finds the last column very informative. The controller allocates a fixed amount of cash in a given period to pay bills, paying the earliest bills first. If the firm wants to spend no more than \$60,000 to pay bills, the controller orders payment of only the first five bills.

## 2.7.2 Format of Fields in the Detail Lines

The Report Writer determines a default format for each print item based on the edit string or picture clauses in the record definition or variable declaration. However, you can, at your option, control the format within the PRINT statement.

**2.7.2.1 Order of Print Items** — The order of the field names in the PRINT statement determines the left to right printing order of the detail line items. For example, you should use the following statement to display the fields from PAYABLES in the following order: WHSLE\_PRICE, ITEMS\_RECEIVED, INVOICE\_DUE, and BILL\_PAID:

```
RW> PRINT WHSLE_PRICE, ITEMS_RECEIVED, INVOICE_DUE, BILL_PAID
RW>
```



**2.7.2.2 Column Position of Print Items** — The Report Writer automatically sets up the column spacing, based on field, header, and page widths. If you want to change the default spacing, you can specify the print position of any or all of the print items. In either case, if you do not leave enough room for the column headers and data items, the Report Writer “wraps” the detail line. That is, it prints some items on a second line, including column headers as space permits.

If you choose to specify print positions, you can:

- Specify the column number where the Report Writer begins to print each item
- Require spacing between columns by including a SPACE n element in the PRINT statement

The following example uses the first option to display the same fields from PAYABLES; however, each field begins at 20 space intervals because the column number is specified:

```
RW> PRINT COL 1, WHSLE_PRICE, COL 21, ITEMS_RECEIVED,
RW>     COL 41, INVOICE_DUE, COL 61, BILL_PAID
```

WHSLE PRICE	ITEMS RECEIVED	INVOICE DUE	BILL PAID
\$40,000	NO GOODS	NO INVCE	NOT PAID
\$28,500	5/24/82	1/02/83	6/15/82
\$13,500	6/21/82	2/15/83	NOT PAID

The following example displays the same fields but uses a SPACE n element to specify only five spaces between columns:

```
RW> PRINT WHSLE_PRICE , SPACE 5, ITEMS_RECEIVED,
RW>     SPACE 5, INVOICE_DUE, SPACE 5, BILL_PAID
```

WHSLE PRICE	ITEMS RECEIVED	INVOICE DUE	BILL PAID
\$40,000	NO GOODS	NO INVCE	NOT PAID
\$28,500	5/24/82	1/02/83	6/15/82
\$13,500	6/21/82	2/15/83	NOT PAID

You do not have to specify a column number or spacing option for every field in the PRINT statement. If you do not, the Report Writer automatically sets up the space between the remaining columns.

**2.7.2.3 Edit String Format of Print Items** — If you declare an edit string for a field in the record definition, the Report Writer uses that edit string to format the print item. If you are setting up print items derived from field values, the Report Writer sets up its own edit string. However, you can override the Report Writer format for a print item by specifying an edit string.

The full range of edit strings for record definitions is available to you in PRINT or AT statements. With a USING clause, you can indicate how the values of alphabetic or numeric fields should be displayed. The USING clause can contain commas, hyphens, percent signs, dollar signs, or decimal points. See the *VAX DATATRIEVE Reference Manual* for more information on edit strings.

If, for example, you want a numeric field to represent money and if all values for the field are less than \$10,000, you could choose the following edit string:

```
RW> PRINT PRICE/DISP USING $$,###.99
```

Some typical values formatted with this edit string are \$1,016.97 and \$995.38.

If you have negative values, you can indicate whether the sign should precede or follow the number. Alternatively, you can have “DB” follow the number, or you can enclose the number in parentheses when the value is negative. For example, if you want a field to represent the balance due in an account, you could choose the following edit string:

```
RW> PRINT BALANCE USING ((###.99))
```

If the balance is -\$567.88, DATATRIEVE displays it as (\$567.88). Note that double parentheses are required when specifying the edit string. If a field value contains a large amount of text, specify a T edit string. The format of a T edit string is T(n), where n is the column width. This edit string instructs the Report Writer to print the text over several lines, with a maximum of n characters per line. No words are divided across lines, and the other print items are printed on the same line as the first line of text.

Figure 2-2 shows the structure of the record EMP\_REVIEW\_REC for the EMP\_REVIEW domain. EMP\_REVIEW is used in the example that follows. (See Appendix A for the complete record definition.)

01 EMP_REVIEW_REC				
05 ID	05 NAME	05 JOB	05 EVALUATION	05 EVAL_DATE

MK-01121-00

**Figure 2-2: Field Structure of EMP\_REVIEW\_REC**

The following example shows a report of the job evaluations of employees from the EMP\_REVIEW domain. It specifies that the EVALUATION field be printed with a column width of 15 characters:

```
DTR> REPORT EMP_REVIEW
RW> SET COLUMNS_PAGE = 70
RW> SET REPORT_NAME = "EMPLOYEE REVIEW"
RW> PRINT ID, NAME, JOB,
RW>     EVALUATION USING T(15), EVAL_DATE, SKIP
RW> END_REPORT
```

EMPLOYEE REVIEW

5-May-1987  
Page 1

ID	NAME	JOB	EVALUATION	EVAL DATE
11111	BRAD H.	DEVELOPER	Brad did a fine job in implementing staged output.	29-Apr-87
22222	DAVID D.	CONSULTANT	David is a master in developing report specifications.	12-Mar-87
88888	TERRY C.	DEVELOPER	Terry is an exceptional system manager and developer.	21-Mar-87

DTR>

The edit string T(15) sets up a maximum of 15 characters per line in the EVALUATION column. In addition, T edit strings ensure that no words are divided across lines. However, if a word is bigger than the T format width, it is split.

### 2.7.3 Column Headers for Print Items

DATATRIEVE displays column headers at the top of the report and at the top of each page. If a query header has been defined for a field or variable, the Report Writer uses the query header as the default column header; otherwise, the Report Writer uses the field or variable name as the default.

DATATRIEVE allows you to override this default in one of two ways: by suppressing the column header or by specifying a new column header. These options are explained in the following sections.



**2.7.3.1 Option 1: Suppressing Column Headers** — If you do not want a column header for an individual field, follow the print-list item with a hyphen in parentheses (-). For example:

```
DTR> REPORT FIRST 1 YACHTS
RW> SET COLUMNS_PAGE = 60
RW> PRINT MANUFACTURER, MODEL (-), LOA, RIG, PRICE (-)
RW> END_REPORT
```

9-Apr-1987  
Page 1

MANUFACTURER		LENGTH OVER ALL	RIG	
ALBERG	37 MK II	37	KETCH	\$36,951

DTR>

If you do not want any column headers in your report, use the SET NO COLUMN\_HEADER statement in your report specification:

```
DTR> REPORT FIRST 2 YACHTS
RW> SET NO COLUMN_HEADER
RW> PRINT MANUFACTURER, MODEL, LOA, RIG, PRICE
RW> END_REPORT
```

2-Nov-1986  
Page 1

ALBERG	37 MK II	37	KETCH	\$36,951
ALBIN	79	26	SLOOP	\$17,900

DTR>

**2.7.3.2 Option 2: Specifying a Column Header** — You can choose the column header by following a print-list item with the column header enclosed in parentheses and quotation marks. For example, you could change the name of the first column header in the previous example from MANUFACTURER to VENDOR by using this PRINT statement:

```
RW> PRINT MANUFACTURER ("VENDOR"), MODEL (-), LOA, RIG, PRICE (-)
```

This produces the following report:

5-May-1987  
Page 1

VENDOR		LENGTH OVER ALL	RIG	
ALBERG	37 MK II	37	KETCH	\$36,951

DTR>

To include a two-line column header, use the slash (/) between quoted strings.  
For example:

```
DTR> REPORT FIRST 1 YACHTS
RW> PRINT MANUFACTURER("BOAT"/"BUILDER"), MODEL (-), LOA,
RW>     RIG, PRICE (-)
RW> END_REPORT
DTR>
```

This statement produces:

5-May-1987  
Page 1

BOAT BUILDER		LENGTH OVER ALL	RIG	
ALBERG	37 MK II	37	KETCH	\$36,951

For fields with one or two character values, you can compress the headers. The following PRINT statement specifies a three-line header for the LOA field:

```
RW> PRINT MANUFACTURER, LOA ("L"/"O"/"A"), RIG, PRICE
```

This produces the following report:

28-May-1987  
Page 1

MANUFACTURER	L O A	RIG	PRICE
ALBERG	37	KETCH	\$36,951

## 2.8 Summarizing Data

You can instruct the Report Writer to calculate summary statistics for any column in the report. Use the AT BOTTOM OF PAGE or the AT BOTTOM OF REPORT statement for end-of-page or end-of-report summaries. (An AT BOTTOM OF field-name statement can generate summaries on groups within the report. See the chapter on mastering report writing techniques for the discussion on dividing data records into groups.)

As a general rule, it is best not to use AT TOP statement for summarizing data. These statements are for printing special headings in the report.

The following functions are available for summary lines:

- AVERAGE
- COUNT
- Maximum value (MAX)
- Minimum value (MIN)
- Standard deviation (STD\_DEV)
- TOTAL

Two other functions, RUNNING COUNT and RUNNING TOTAL, provide running summaries within each detail line. They can be included in a PRINT statement. See the section on value expressions in this chapter. They can also be used to compute running summaries of the groups within the report in an AT BOTTOM of field-name statement. See the chapter on mastering report writing techniques for more information.

### 2.8.1 COUNT, AVERAGE, and TOTAL

Figure 2-3 shows the logical structure of the record SALES\_REC for the SALES domain. SALES is used in the example that follows. (See Appendix A for the complete record definition.)

01 SALESREC				
05 ID	05 SALES_NAME	05 START_DATE	05 MONTHS_EMP	05 AMOUNT

MK-01122-00

**Figure 2-3: Field Structure of SALES\_REC**

The field definition for MONTHS\_EMP is:

```
05 MONTHS_EMP COMPUTED BY ("TODAY" - START_DATE)/30
   EDIT_STRING IS Z29.
```

MONTHS\_EMP is a COMPUTED BY field based on the value of START\_DATE and the date value expression "TODAY", representing the current system date. See the *VAX DATATRIEVE Reference Manual* for more information on the date value expression "TODAY".

---

**Note**

---

Because MONTHS\_EMP varies according to the current system date, the value of MONTHS\_EMP depends on the day that you use the SALES domain. As a result, you probably will not duplicate the output shown in the examples using SALES. The same principle applies to examples using the AGE field from the PAYABLES domain.

---

In the following example, the Acme Computer Company has a SALES domain that stores data about the members of its sales force. The report shows the name, starting date, months employed, and amount sold for each salesperson at Acme Computer. The bottom of the report indicates the number of salespeople, the total amount sold, and the average amount sold:

```
DTR> REPORT SALES
RW> SET COLUMNS_PAGE = 60
RW> SET REPORT_NAME = "ACME COMPUTER"/"SALES REPORT"
RW> PRINT SALES_NAME, START_DATE, MONTHS_EMP, AMOUNT
RW> AT BOTTOM OF REPORT PRINT SKIP 2,
RW>   COL 10, "SALES FORCE:", SPACE, COUNT (-) USING Z9,
RW>   COL 38, "TOTAL:", TOTAL AMOUNT USING $$$,$$.99,
RW>   COL 38, "AVERAGE:", AVERAGE AMOUNT USING $$,$$.99
RW> END_REPORT
```

(continued on next page)

SALES NAME	START DATE	MONTHS EMP	AMOUNT
ANNE DINNAN	1-Apr-1983	3	\$2,389.90
NANCY ROTHBLATT	1-May-1983	2	\$6,325.88
LINDA REINE	15-Dec-1982	7	\$8,532.22
WAYNE SMITH	1-Feb-1983	5	\$9,853.52
JAMES STORER	15-May-1982	14	\$25,876.02
SANDY LEVINE	15-Nov-1982	8	\$10,000.01
SEYMOUR KIMMELMAN	15-Feb-1983	5	\$7,325.67
JOSEPH FREDERICK	1-Mar-1983	4	\$5,000.00
RICK LANGHART	15-Mar-1983	4	\$4,999.99
WILLIAM SULLIVAN	15-Oct-1982	9	\$8,672.99
DAN DERRICK	16-Nov-1982	8	\$11,456.87
LYDIA BARNET	1-Jun-1983	1	\$2,598.79
HENRY MAILER	15-Dec-1982	7	\$9,999.99
DENNIS MCADOO	1-Aug-1982	11	\$12,345.62

SALES FORCE: 14

TOTAL: \$125,377.47  
AVERAGE: \$8,955.53

DTR)

Summary reports using DATATRIEVE statistical functions are very useful for financial analysis. The AT BOTTOM statement lets you produce summary lines with the total number of salespeople (COUNT), and the AVERAGE and TOTAL amounts sold (AVERAGE and TOTAL). The AVERAGE and TOTAL of AMOUNT automatically print in the AMOUNT column with the edit string you indicated.

### 2.8.2 Maximum Value, Minimum Value, and Standard Deviation

To show the maximum and minimum values of selected fields, use the statistical operators MAX and MIN on the field names. You can also print the standard deviation of numeric fields. Use the operator STD\_DEV with an AT BOTTOM statement.

The following example expands the sales report to indicate the maximum amount sold, the minimum amount sold, and the standard deviation of the amount sold. The example is written as a VMS command file (SALESTAT.COM) so that you can invoke the report at DCL command level.



Here is the file SALESTAT.COM:

```
READY SALES
REPORT SALES
SET REPORT_NAME = "ACME COMPUTER"/
  "DETAILED SALES REPORT"
SET COLUMNS_PAGE = 60
PRINT SALES_NAME, START_DATE, MONTHS_EMP, AMOUNT
AT BOTTOM OF REPORT PRINT SKIP 2,
  COL 10, "SALES FORCE:", SPACE, COUNT(-) USING Z9,
  COL 38, "TOTAL:", TOTAL AMOUNT USING $$, $$$, $$$ .99,
  COL 38, "AVERAGE:", AVERAGE AMOUNT USING $$, $$$ .99,
  COL 38, "MAXIMUM:", MAX AMOUNT USING $$$, $$$ .99,
  COL 38, "MINIMUM:", MIN AMOUNT USING $$, $$$ .99,
  COL 38, "STD DEV:", STD_DEV AMOUNT USING $$, $$$ .99
END_REPORT
```

You do not need to enter DATATRIEVE to produce the report. Use your global symbol for invoking DATATRIEVE, followed by the at sign character (@) and the command file name. The following example uses DTR32 as the DATATRIEVE symbol:

```
$ DTR32 @SALESTAT
```

After a listing of the statements in SALESTAT.COM, the terminal displays the following report:

```
                ACME COMPUTER                2-Jul-1983
              DETAILED SALES REPORT          Page 1
```

SALES NAME	START DATE	MONTHS EMP	AMOUNT
ANNE DINNAN	1-Apr-1982	3	\$2,389.90
NANCY ROTHBLATT	1-May-1982	2	\$6,325.88
LINDA REINE	15-Dec-1981	7	\$8,532.22
WAYNE SMITH	1-Feb-1982	5	\$9,853.52
JAMES STORER	15-May-1981	14	\$25,876.02
SANDY LEVINE	15-Nov-1981	8	\$10,000.01
SEYMOUR KIMMELMAN	15-Feb-1982	5	\$7,325.67
JOSEPH FREDERICK	1-Mar-1982	4	\$5,000.00
RICK LANGHART	15-Mar-1982	4	\$4,999.99
WILLIAM SULLIVAN	15-Oct-1983	9	\$8,672.99
DAN DERRICK	16-Nov-1981	8	\$11,456.87
LYDIA BARNET	1-Jun-1982	1	\$2,598.79
HENRY MAILER	15-Dec-1981	7	\$9,999.99
DENNIS MCADOO	1-Aug-1981	11	\$12,345.62

SALES FORCE: 14	TOTAL:	\$125,377.47
	AVERAGE:	\$8,955.53
	MAXIMUM:	\$25,876.02
	MINIMUM:	\$2,389.90
	STD DEV:	\$5,153.85

\$

Another way to produce the report is to define a procedure within DATATRIEVE. To invoke the procedure at DCL level, type your symbol to enter DATATRIEVE, followed by EXECUTE and the procedure name. For example, assume that the procedure name is SALES\_RPT and your symbol is DTR32. Type:

```
$ DTR32 EXECUTE SALES_RPT
```

DATATRIEVE then displays the report on your terminal. Of course, you can also invoke the procedure at DTR command level by typing a colon and the procedure name.

These last two examples use AT BOTTOM statements to produce overall page and report summaries. AT BOTTOM statements can also be used to divide your data records into groups. You can then compile statistics about groups of records, as well as the entire report. The chapter on mastering report writing techniques shows how to report summary data on groups and illustrates other advanced report writing techniques.

## Mastering Report Writing Techniques **3**

Many report requirements are quite complex and present special problems. You may want to:

- Summarize information on key groups within the report
- Use dates as a basis of organizing data for the report
- Report data from two domains that share a common field
- Compare a budget with actual expenses
- Print a title page and special page headings
- Produce a column of totals for each detail line (cross tabulation)
- Report data from hierarchical records
- Print a variety of detail lines in one report, depending on conditional tests

To help you meet these requirements, this chapter describes useful techniques and illustrates them with sample reports.

### **3.1 Dividing Data Records into Groups**

You may often need to report not only on a body of data but also on the groups within it. For example, you could report on employees sorted by department, with summary totals for each department as well as for all employees. Groups of sorted records are called **control groups**. A control group is a series of sorted data records that have the same value in one or more fields.



### 3.1.1 Developing Control Groups with a Sort Key

When you sort a group of records, you choose at least one field as the basis for the sort. That field is called the **sort key**. Sometimes every record in a record stream has a unique value for the sort key. For example, every employee's ID number is unique. When you sort employee records by the ID number, there are no records with duplicate values in that field. You are not grouping similar records.

In other cases, the number of unique values for a sort key may be small compared to the number of sorted records. For example, a company of 500 employees may have only 10 departments. That is, there are only 10 unique values for DEPT in the collection of 500 employee records. When you sort the employee records by the department code (DEPT), you create 10 control groups. DEPT would be a meaningful sort key for grouping employee records because it enables you to group records that share a common field value.

Figure 3-1 shows the logical structure of the record PERSONNEL\_REC for the PERSONNEL domain. PERSONNEL is used in the example that follows. (See Appendix A for the complete record definition.)

01 PERSON							
05 ID	05 STATUS	05 NAME		05 DEPT	05 START _ DATE	05 SALARY	05 SUP _ ID
		10 FIRST _ NAME	10 LAST _ NAME				

MK-01123-00

**Figure 3-1: Field Structure of PERSONNEL\_REC**

The following REPORT statement identifies all records from three specific departments and establishes DEPT as the sort key for the records.

DATATRIEVE then sorts the records according to the department code:

```
DTR> REPORT PERSONNEL WITH DEPT = "D98", "E46", "T32" SORTED BY DEPT
RW> SET COLUMNS_PAGE = 70
RW> PRINT ID, DEPT, FIRST_NAME, LAST_NAME, SALARY
RW> END_REPORT
```

ID	DEPT	FIRST NAME	LAST NAME	SALARY
02943	D98	CASS	TERRY	\$29,908
39485	D98	DEE	TERRICK	\$55,829
49843	D98	BART	HAMMER	\$26,392
84375	D98	MARY	NALEVO	\$56,847
38465	E46	JOANNE	FREIBURG	\$23,908
48475	E46	GAIL	CASSIDY	\$55,407
34456	T32	HANK	MORRISON	\$30,000
38462	T32	BILL	SWAY	\$54,000
48573	T32	SY	KELLER	\$31,546
83764	T32	JIM	MEADER	\$41,029

DTR)

If the desired field is the primary key for the records or if you have formed and sorted a collection, you do not need to sort the records again within the REPORT statement. If the records are not already sorted, you can sort them within the REPORT statement. Once the records are sorted, you can use an AT TOP OF field-name or an AT BOTTOM OF field-name statement to create control groups based on the values of the field specified.

Remember that to form control groups based on a sort key, you must sort the records by a field and use the same field name in the AT TOP OF or AT BOTTOM OF statement. The field name you use must be either a field name specified in the record description or a variable name created in a DECLARE statement. See the *VAX DATATRIEVE User's Guide* for more information on variables.

In the following example, an AT BOTTOM OF DEPT statement groups records according to department. Within this statement, you can use one or more of the DATATRIEVE statistical operators to summarize data about the employees in each department control group.

The example shows how Bock's Yachts keeps salary records for employees in various departments with the PERSONNEL domain. The firm groups the data according to departments so that it can make financial comparisons.

The report displays information on all employees in departments D98, E46, and T32 within the company. It indicates the total salary for each department and keeps a running total for the salary for the three departments combined.

Follow these steps:

1. Sort the records according to the sort key DEPT.
2. Use an AT TOP OF DEPT statement to print the value for DEPT at the beginning of each DEPT control group.
3. Indicate the values to be printed in each detail line. Use a concatenation expression, indicated by three vertical lines, to allow exactly one space between the values of FIRST\_NAME and LAST\_NAME. Specify a header ("NAME").
4. Summarize salary information on each department with an AT BOTTOM OF DEPT statement. TOTAL SALARY gives the total salary for a department. RUNNING TOTAL (TOTAL SALARY) gives the total thus far for all the departments. Specify a column position so that this value is displayed in the SALARY column rather than in a separate RUNNING TOTAL column.

```
DTR> SHOW SALARY_REPORT
```

```
PROCEDURE SALARY_REPORT
REPORT PERSONNEL WITH DEPT = "D98","E46","T32" SORTED BY DEPT—————(1)
SET REPORT_NAME = "SALARY REPORT"
SET COLUMNS_PAGE = 60
AT TOP OF DEPT PRINT DEPT—————(2)
PRINT ID, FIRST_NAME!!!LAST_NAME ("NAME"), SALARY—————(3)
AT BOTTOM OF DEPT PRINT SKIP,—————(4)
    COL 36, DEPT!!!"TOTAL:",
    TOTAL SALARY USING $$$,$$$, SKIP, COL 13,
    "*****",
    SKIP, COL 32, "OVERALL TOTAL:", COL 50,
    RUNNING TOTAL (TOTAL SALARY) USING $$$,$$$, SKIP
END_REPORT
END_PROCEDURE
```

Figure 3-2 shows the report produced by this specification.

## SALARY REPORT

14-May-1984

Page 1

DEPT	ID	NAME	SALARY
D98			
	02943	CASS TERRY	\$29,908
	39485	DEE TERRICK	\$55,829
	49843	BART HAMMER	\$26,392
	84375	MARY NALEVO	\$56,847
		D98 TOTAL:	\$168,976
		*****	
		OVERALL TOTAL:	\$168,976
E46			
	38465	JOE FREIBURG	\$23,908
	48475	GAIL CASSIDY	\$55,407
		E46 TOTAL:	\$79,315
		*****	
		OVERALL TOTAL:	\$248,291
T32			
	34456	HANK MORRISON	\$30,000
	38462	BILL SWAY	\$54,000
	48573	SY KELLER	\$31,546
	83764	JIM MEADER	\$41,029
		T32 TOTAL:	\$156,575
		*****	
		OVERALL TOTAL:	\$404,866

**Figure 3-2: Control Group Report Based on One Sort Key**

### 3.1.2 Developing Levels of Control Groups Using Multiple Sort Keys

The last example used one sort key to sort the records and an AT statement to establish control groups. However, there may also be times when you wish to identify subgroups within these control groups. It is possible for one control group to contain other control groups based on the values of other sort keys. For example, you could sort a personnel file by department and by type of employee as specified in the STATUS field. Each department group contains several control groups for the types of employees within that department.



The following example also uses the PERSONNEL domain. It shows the two types of employees that work at Bock's Yachts: experienced workers and trainees. The STATUS field takes one of two values: EXPERIENCED or TRAINEE. The report shows salaries for each department and for each type of employee within a given department.

Follow these steps:

1. Sort the records according to two sort keys, DEPT and STATUS.
2. Use AT TOP OF DEPT and AT TOP OF STATUS statements to set up special headings for the control groups.
3. Print the field values of the detail line. Use a concatenation expression (|||) to allow exactly one space between FIRST\_NAME and LAST\_NAME. Specify a header for the full name ("NAME").
4. Use AT BOTTOM OF DEPT and AT BOTTOM OF STATUS statements to print summary lines for each control group.
5. Summarize the entire report with an AT BOTTOM OF REPORT statement.

The report specification follows these basic steps to produce a double control break report. (For brevity, this report is limited to employees in departments D98 and T32.)

This is the procedure SALARY\_REPORT2:

```

DTR) SHOW SALARY_REPORT2
PROCEDURE SALARY_REPORT2
REPORT PERSONNEL WITH DEPT = "D98","T32" SORTED BY DEPT, STATUS_____ (1)
SET REPORT_NAME = "DETAILED SALARY REPORT"
SET COLUMNS_PAGE = 70
AT TOP OF DEPT PRINT DEPT_____ (2)
AT TOP OF STATUS PRINT STATUS_____ (2)
PRINT ID, FIRST_NAME|||LAST_NAME ("NAME"), SALARY_____ (3)
AT BOTTOM OF STATUS PRINT SKIP 2, COL 38,_____ (4)
    STATUS|||"TOTAL:", TOTAL SALARY USING
    $, $$$, $$$, SKIP
AT BOTTOM OF DEPT PRINT COL 36,_____ (4)
    "DEPARTMENT"|||DEPT|||"TOTAL:",
    TOTAL SALARY USING $, $$$, $$$,
    COL 30, "-----", SKIP
AT BOTTOM OF REPORT PRINT COL 15,_____ (5)
    "*****",
    SKIP 2, COL 38, "GRAND TOTAL SALARY:",
    TOTAL SALARY USING $, $$$, $$$
END_REPORT
END_PROCEDURE

DTR) :SALARY_REPORT2
  
```



DETAILED SALARY REPORT

14-May-1984

Page 1

DEPT	STATUS	ID	NAME	SALARY	
D98	EXPERIENCED	02943	CASS TERRY	\$29,908	
		39485	DEE TERRICK	\$55,829	
		84375	MARY NALEVO	\$56,847	
			EXPERIENCED TOTAL:	\$142,584	
	TRAINEE	49843	BART HAMMER	\$26,392	
				TRAINEE TOTAL:	\$26,392
			DEPARTMENT D98 TOTAL:	\$168,976	
	-----				
	T32	EXPERIENCED	38462	BILL SWAY	\$54,000
			83764	JIM MEADER	\$41,029
			EXPERIENCED TOTAL:	\$95,029	
TRAINEE		34456	HANK MORRISON	\$30,000	
		48573	SY KELLER	\$31,546	
			TRAINEE TOTAL:	\$61,546	
			DEPARTMENT T32 TOTAL:	\$156,575	
-----					
*****					
		GRAND TOTAL SALARY:	\$325,551		

Figure 3-3: Control Group Report Based on Two Sort Keys

### 3.1.3 Reporting Group Summaries Only

Control groups allow you to separate groups of detail lines and to print group summaries. However, sometimes you may want only the summary information for the groups. You can produce a report with summary lines and no detail lines using either the Report Writer or the SUM statement,

**3.1.3.1 Summaries Through the Report Writer** — You can produce a report consisting only of summary lines for control groups with the AT BOTTOM OF field-name statement.

The following example creates a report showing salary information for each department in Bock's Yachts. It includes the number of employees in each department, the total salary, and the average salary. Finally, for the entire company, it indicates total number of employees, total salary, and average salary.

Follow these steps:

1. Use AT BOTTOM OF DEPT to print each line of the body of the report. Each line summarizes a different department. Do not use the PRINT statement. (You are reporting on group totals, not on the individual members of the group.)
2. Use COUNT as a print item for the number of employees.
3. Use AT BOTTOM OF REPORT for the aggregate summaries. COUNT provides the total of all employees because it represents the total of all records processed.

The following report specification is enclosed in the procedure SALARY\_TOTALS. (To create this procedure, use a DEFINE PROCEDURE command.)

```
DTR> SHOW SALARY_TOTALS
PROCEDURE SALARY_TOTALS
READY PERSONNEL
REPORT PERSONNEL SORTED BY DEPT
SET REPORT_NAME = *. "a report name"
SET COLUMNS_PAGE = 60
AT BOTTOM OF DEPT PRINT COL 10, DEPT, _____(1)
    COL 20, COUNT ("NUMBER"/"EMPLOYEES"), _____(2)
    COL 35, TOTAL SALARY ("TOTAL"/"SALARY") USING $, $$$, $$$,
    COL 50, AVERAGE SALARY ("AVERAGE"/"SALARY") USING $$$, $$$
AT BOTTOM OF REPORT PRINT SKIP 2, COL 10, _____(3)
    "*****",
    SKIP 2, COL 10, "CORPORATE:", COL 20, COUNT,
    COL 35, TOTAL SALARY USING $, $$$, $$$,
    COL 50, AVERAGE SALARY USING $$$, $$$
END_REPORT
END_PROCEDURE
```

DTR) :SALARY\_TOTALS  
Enter a report name: "SALARY REPORT BY DEPARTMENT"

SALARY REPORT BY DEPARTMENT 14-May-1984  
Page 1

DEPT	NUMBER EMPLOYEES	TOTAL SALARY	AVERAGE SALARY
C82	5	\$202,465	\$40,493
D98	4	\$168,976	\$42,244
E46	2	\$79,315	\$39,658
F11	4	\$151,566	\$37,892
G20	3	\$117,554	\$39,185
T32	4	\$156,575	\$39,144
TOP	1	\$75,902	\$75,902

\*\*\*\*\*

CORPORATE: 23 \$952,353 \$41,407

DTR)

**3.1.3.2 Summaries by the SUM Statement Outside of the Report Writer** — You can generate a simple report that produces control group totals outside of the Report Writer by using the SUM statement. The SUM statement operates on the CURRENT collection. The collection need not be presorted because DATATRIEVE sorts the data while processing the SUM statement. You cannot use the SUM statement for other control group statistics, such as averages.

The following report shows salary information for each department. It includes the number of employees in each department and the total salary:

DTR) READY PERSONNEL  
DTR) FIND PERSONNEL  
[23 records found]  
DTR) SUM 1 ("Number"/"Employees"),  
[Looking for next element in list]  
CON) SALARY USING \$\$\$,\$\$\$ BY DEPT

(continued on next page)

DEPT	Number Employees	TOTAL SALARY	Number Employees
C82	5	\$202,465	
D98	4	\$168,976	
E46	2	\$79,315	
F11	4	\$151,566	
G20	3	\$117,554	
T32	4	\$156,575	
TOP	1	\$75,892	
		\$952,343	23

DTR)

### 3.2 Reporting Data Grouped by Date

When you have date fields in records, you may want to group the records according to similar dates. Sorting records by date is a common practice in accounting, for example. Financial managers and controllers want the data on accounts receivable or accounts payable broken down into groups based on dates. They need to know the dollar value of accounts that are one month old, two months old, and so on. These divisions are often called aging categories, and they are a valuable tool of financial analysis.

To produce this kind of report, set up groups based on common ages. Then report summary information on all accounts in a given aging category.

Figure 3-4 shows the logical structure of the record PAYABLES\_REC for the PAYABLES domain. PAYABLES is used in the example that follows. (See Appendix A for the complete record definition.)

01 PAYABLE						
05 ORDER_ NUM	05 TYPE		05 ITEMS_ RECEIVED	05 INVOICE_ DUE	05 BILL_ PAID	05 WHSLE_ PRICE
	10 MANUFACTURER	10 MODEL				

MK-01120-00

**Figure 3-4: Field Structure of PAYABLES\_REC**

There are two ways to develop the accounts payable report:

- Edit the record definition to form a new field AGE giving the age of an account in months. Then produce a control group report sorted by AGE.
- Use the FORMAT value expression to identify the month and year component of the value for INVOICE\_DUE. Then group records with the same month and year for INVOICE\_DUE.

The following sections discuss each of these in more detail. Each section shows how Bock's Yachts uses a PAYABLES domain to keep track of accounts payable. The record PAYABLES\_REC has a field called INVOICE\_DUE for storing the invoice due date. The report in each section shows the accounts sorted by the age in months and summarizes the total accounts payable in each category.

### 3.2.1 Solution 1: Control Groups Based on Dates

One way to produce the accounts payable report is to form control groups based on the account's age in months. Edit the record definition to add a COMPUTED BY field called AGE. To determine the account's age, subtract the account's date from today's date and divide the result by 30. The account date is the value of the field INVOICE\_DUE. "TODAY" is a date value expression that has the value of the current system date. The expression ("TODAY" - INVOICE\_DUE)/30 gives the approximate age in months. Because you want to group accounts with a similar age together, retain only the integer part of the expression. The function FN\$FLOOR returns integer values for the field AGE.

Give AGE this field definition:

```
05 AGE          COMPUTED BY FN$FLOOR(("TODAY" - INVOICE_DUE)/30)
                  EDIT_STRING IS 29.
```

Enter a FINISH command to clear the old definition; then reedit PAYABLES again to use the new definition.

Now you can develop a procedure to produce the accounts payable report (Figure 3-5). Follow these steps:

1. Report the domain PAYABLES sorted by AGE.
2. Use an AT BOTTOM OF AGE statement to print totals for each monthly aging category.
3. Summarize data on all accounts with an AT BOTTOM OF REPORT statement.



The following is the procedure AGING\_REPORT:

```

DTR) SHOW AGING_REPORT
PROCEDURE AGING_REPORT
READY PAYABLES
REPORT PAYABLES WITH INVOICE_DUE NOT MISSING SORTED BY AGE-----(1)
PRINT AGE ("A"/"G"/"E"), INVOICE_DUE, TYPE, WHSLE_PRICE
SET COLUMNS_PAGE = 70
SET REPORT_NAME = "ACCOUNTS PAYABLE REPORT"
AT BOTTOM OF AGE PRINT SKIP 2, COL 20, "NUMBER OF ACCOUNTS:",-----(2)
    SPACE, COUNT (-) USING Z9, COL 53, "TOTAL:",
    TOTAL WHSLE_PRICE USING $$$$, $$$, SKIP
AT BOTTOM OF REPORT PRINT COL 15, "TOTAL NUMBER OF ACCOUNTS:",-----(3)
    SPACE, COUNT (-) USING Z9,
    COL 53, "TOTAL:", TOTAL WHSLE_PRICE USING $$$$, $$$
END_REPORT
END_PROCEDURE

```

DTR) :AGING\_REPORT

ACCOUNTS PAYABLE				28-May-1984
AGING REPORT				Page 1
A	INVOICE	VENDOR	ITEM_TYPE	WHSLE
E	DUE			PRICE
1	4/15/83	ALBIN	VEGA	\$14,250
1	4/01/83	BAYFIELD	30/32	\$13,000
1	4/01/83	IRWIN	37 MARK II	\$29,999
NUMBER OF ACCOUNTS: 3				TOTAL: \$57,249
2	3/14/83	ALBIN	BALLAD	\$23,850
2	3/25/83	ALBIN	FLAGPOLES	\$48
2	3/14/83	BOMBAY	CLIPPER	\$18,150
2	3/02/83	ISLANDER	BAHAMA	\$4,950
NUMBER OF ACCOUNTS: 4				TOTAL: \$46,998
3	2/15/83	ALBIN	79	\$13,500
3	2/14/83	AMERICAN	26	\$9,000
3	1/31/83	AMERICAN	26-MS	\$15,150
3	2/12/83	WINDPOWER	IMPULSE	\$1,500
NUMBER OF ACCOUNTS: 4				TOTAL: \$39,150

(continued on next page)

Figure 3-5: Report with Detail Lines Grouped by Date

4	1/02/83	ALBERG	37 MK II	\$28,500
4	12/30/82	CAPE DORY	TYPHOON	\$3,150
4	1/25/83	SALT	19	\$4,850
NUMBER OF ACCOUNTS: 3			TOTAL:	\$36,500
5	12/01/82	GRAMPIAN	34	\$25,250
NUMBER OF ACCOUNTS: 1			TOTAL:	\$25,250
TOTAL NUMBER OF ACCOUNTS: 15			TOTAL:	\$205,147

**Figure 3-5: Report with Detail Lines Grouped by Date (Cont.)**

You could generate a similar report based on records from a RECEIVABLES domain. Financial administrators may want data on how efficiently their organization is collecting accounts receivable. If there are many outstanding accounts with an age of five or greater, it is probably time to look at the operation of the credit department.

### 3.2.2 Solution 2: Grouping Records with a Common Month and Year

There is another way to produce a report that groups accounts that have a similar age. You could group all the records where INVOICE\_DUE has a common month and year. For example, you want information on all accounts due in April of 1984, in May of 1984, and so on.

Producing this report requires a different technique. You identify the month and year portion from each value of INVOICE\_DUE with the FORMAT value expression. Then, group records with the same value for the month and year of INVOICE\_DUE. Follow these steps:

1. Declare a variable (ACCT\_MONTH) to compute the month and year for each value of INVOICE\_DUE with a FORMAT value expression.
2. Identify the data with a REPORT rse statement, sorting the records according to the value for INVOICE\_DUE.
3. Begin each month's report on a new page numbered 1 with the NEW\_SECTION element in the print list of the AT TOP statement.
4. Indicate the content of the detail lines with a PRINT statement.
5. Summarize each month's accounts with an AT BOTTOM OF ACCT\_MONTH statement.

The procedure MONTHLY\_ACCT\_RPT generates one report for each month's accounts payable:

```
DTR> SHOW MONTHLY_ACCT_RPT
PROCEDURE MONTHLY_ACCT_RPT
DECLARE ACCT_MONTH COMPUTED BY_____ (1)
    FORMAT INVOICE_DUE USING YYMM.
REPORT PAYABLES WITH INVOICE_DUE NOT MISSING SORTED BY_____ (2)
    INVOICE_DUE
SET COLUMNS_PAGE = 70
SET REPORT_NAME = "ACCOUNTS PAYABLE"
AT TOP OF ACCT_MONTH PRINT NEW_SECTION_____ (3)
PRINT INVOICE_DUE, TYPE, WHSLE_PRICE_____ (4)
AT BOTTOM OF ACCT_MONTH PRINT SKIP 2, COL 15,_____ (5)
    "NUMBER OF ACCOUNTS:", SPACE,
    COUNT(-) USING 29, COL 53, "TOTAL:",
    TOTAL WHSLE_PRICE USING $$$,$$$
END_REPORT
END_PROCEDURE
```

Running the procedure produces a multipage report. Each month's accounts payable begins on a new "Page 1". Figure 3-6 shows three pages from the report. To save space, the pages are printed together here.

ACCOUNTS PAYABLE			3-Nov-1983
			Page 1
INVOICE DUE	VENDOR	ITEM_TYPE	WHSLE PRICE
12/01/83	GRAMPIAN	34	\$25,250
12/30/83	CAPE DORY	TYPHOON	\$3,150
NUMBER OF ACCOUNTS: 2		TOTAL: \$28,400	

(continued on next page)

**Figure 3-6: Accounts Payable Report by Month**

## ACCOUNTS PAYABLE

3-Nov-1984  
Page 1

INVOICE DUE	VENDOR	ITEM_TYPE	WHSLE PRICE
1/02/84	ALBERG	37 MK II	\$28,500
1/25/84	SALT	19	\$4,850
1/31/84	AMERICAN	26-MS	\$15,150

NUMBER OF ACCOUNTS: 3

TOTAL: \$48,500

## ACCOUNTS PAYABLE

3-Nov-1984  
Page 1

INVOICE DUE	VENDOR	ITEM_TYPE	WHSLE PRICE
4/01/84	BAYFIELD	30/32	\$13,000
4/01/84	IRWIN	37 MARK II	\$29,999
4/15/84	ALBIN	VEGA	\$14,250

NUMBER OF ACCOUNTS: 3

TOTAL: \$57,249

**Figure 3-6: Accounts Payable Report by Month (Cont.)****3.3 Reporting Data from More Than One Record Source**

By using the CROSS clause, you can report on records from two domains that share a common field. You can report on records that do not share a common field by using an inner print list.

**3.3.1 Using CROSS to Report Data from Two Domains**

An effective way to organize your data is to set up two or more domains that share a common field. For example, all records from the PAYABLES domain and the YACHTS domain contain the group field TYPE. If you use the CROSS clause in specifying an RSE, you can report the combined information from both domains.

The following example uses the PAYABLES domain of Bock's Yachts. The PAYABLES domain is set up so that the records share the group field TYPE with YACHTS. Because TYPE is the primary key of YACHTS with no duplication permitted, each value for TYPE is unique. The PAYABLES domain contains information on the wholesale price for each yacht, and the YACHTS domain stores data on the retail price for each yacht. By crossing the two domains, Bock's Yachts can report both the retail and the wholesale price for each yacht. With a simple computation, the markup on each yacht can be indicated.

The example uses a procedure to report the markup on Bock's Yachts. The wholesale price for a yacht comes from PAYABLES, and the retail price for a yacht comes from YACHTS.

Follow these steps:

1. In the REPORT statement, specify an RSE encompassing two domains by crossing PAYABLES with YACHTS over TYPE.
2. Print ORDR\_NUM, TYPE, WHSLE\_PRICE (the wholesale price from PAYABLES), PRICE (the retail price from YACHTS), and the markup percentage.
3. Use this formula for the markup:

$$(\text{PRICE} - \text{WHSLE\_PRICE}) * 100 / \text{WHSLE\_PRICE}$$

Specify an appropriate edit string with a USING clause.

```
DTR> SHOW MARKUP_REPORT
PROCEDURE MARKUP_REPORT
READY PAYABLES, YACHTS
REPORT PAYABLES CROSS YACHTS OVER TYPE_____ (1)
SET COLUMNS_PAGE = *."columns per page"
SET REPORT_NAME = *."the report name"
PRINT ORDER_NUM, TYPE, _____ (2)
  PRICE ("RETAIL"/"PRICE"), WHSLE_PRICE,
  (PRICE - WHSLE_PRICE) * 100 / WHSLE_PRICE ("MARKUP") USING _____ (3)
  ZZ9.9Z
END_REPORT
END_PROCEDURE
```



```
DTR) :MARKUP_REPORT
Enter columns per page: 70
Enter the report name: "MARKUP ON BOCK'S YACHTS"
```

MARKUP ON BOCK'S YACHTS

11-Jan-1984  
Page 1

ORDER NUM	VENDOR	ITEM_TYPE	RETAIL PRICE	WHSLE PRICE	MARKUP
8101123	CHALLENGER	41	\$51,228	\$40,000	28.1%
8102158	ALBERG	37 MK II	\$36,951	\$28,500	29.7%
8103158	ALBIN	79	\$17,900	\$13,500	32.6%
8103159	ALBIN	BALLAD	\$27,500	\$23,850	15.3%
8103161	ALBIN	VEGA	\$18,600	\$14,250	30.5%
8103162	AMERICAN	26	\$9,895	\$9,000	9.9%
8104101	AMERICAN	26-MS	\$18,895	\$15,150	24.7%
8105161	BAYFIELD	30/32	\$32,875	\$13,000	152.9%
8106184	BOMBAY	CLIPPER	\$23,950	\$18,150	32.0%
8107147	WINDPOWER	IMPULSE	\$3,500	\$1,500	133.3%
8201245	CAPE DORY	TYPHOON	\$4,295	\$3,150	36.3%
8201247	GRAMPIAN	34	\$29,675	\$25,250	17.5%
8202168	SALT	19	\$6,590	\$4,850	35.9%
8202521	IRWIN	37 MARK II	\$36,950	\$29,999	23.2%
8202634	ISLANDER	BAHAMA	\$6,500	\$4,950	31.3%

DTR)

### 3.3.2 Using Inner Print Lists to Compare Budgets with Expenses

It is common practice in both business and personal finance to develop a budget. When you incur an expense, you charge it to a particular category of the budget. At periodic intervals, you may wish to compare each category of the budget with the actual expenditures. A report of the budget should indicate all the expenses for each category and whether you are over budget or under budget in each category.

To compare budgeted expenses with actual expenses, you need to relate two separate record sources. One domain contains a record for each category of the budget. A second domain stores records every time an expenditure is made.

In this example, the budget domain is CATS and the expenses domain is CHECKS. The record definition, CATS\_REC, is:

```
DTR) SHOW CATS_REC
RECORD CATS_REC USING
01 CATS_REC.
  05 CATEGORY      PIC 9.
  05 DESC          PIC X(10).
  05 BUDGET        PIC 99999V99
                  EDIT_STRING IS $$$,$$$,99.
```

The definition for the expense records is:

```
DTR> SHOW CHECKS_REC
RECORD CHECKS_REC USING
01 CHECKS_REC.
  05 CAT PIC 9.
  05 NUM      PIC 999
             EDIT_STRING ZZ9.
  05 CHECK_DATE USAGE DATE.
  05 AMOUNT PIC 999V99
             EDIT_STRING IS $$$$.99.
```

Although you would normally use the CROSS clause to join two record sources, it does not apply here. You need to have budgetary information for all categories, even for those to which no checks were written. If you join the domains over the category field, you form a record stream that leaves out the records from CATS on the categories with no checks. But you need this information for summary totals to determine whether overall you were over budget or under budget.

This type of report requires two record streams. You can bring two record streams into the same DATATRIEVE report by including an inner print list within the PRINT statement. To be sure that all budget categories are included, report on the CATS data. Then, for each budget category, include data on every check that matches on the category field. The check records are identified in an inner print list of the PRINT statement.

Follow these steps:

1. Ready the two domains, CATS and CHECKS.
2. Form a collection of the CHECKS records that fall within a specified period.
3. Report data on each category of the budget.
4. For each budget category, form an inner print list to display data on the relevant checks. The inner print list is:  

```
ALL CHECK_DATE, AMOUNT OF CURRENT WITH CAT = CATEGORY
```
5. Total the amount spent within a category and compute the difference from the budgeted amount. Use an edit string that prints parentheses around a negative value.
6. Total the overall amount spent and amount budgeted and compute the difference.

Here is the expense and budget report:

```

DTR> SHOW CHECK_RPT
PROCEDURE CHECK_RPT
READY CATS, CHECKS-----(1)
FIND CHECKS WITH CHECK_DATE AFTER "01-JAN-82"-----(2)
REPORT CATS-----(3)
PRINT COL 1, CATEGORY, COL 10, DESC, COL 25, ALL CHECK_DATE,-----(4)
      COL 40, AMOUNT OF CURRENT WITH CAT = CATEGORY
AT BOTTOM OF CATEGORY PRINT COL 20,-----(5)
  "Category"!!!CATEGORY!!!"Totals:", COL 42,
  (TOTAL AMOUNT OF CURRENT WITH CAT = CATEGORY) USING
  $$,$$$.$99, COL 54, BUDGET, COL 70,
  (BUDGET - (TOTAL AMOUNT OF CURRENT WITH
  CAT = CATEGORY)) ("DIFFERENCE") USING (($$$.$99)), SKIP
AT BOTTOM OF REPORT PRINT SKIP, COL 28, "TOTALS:",-----(6)
  (TOTAL AMOUNT OF CURRENT) USING $$$,$$$.$99,
  COL 53, TOTAL BUDGET USING $$$,$$$.$99, COL 70,
  (TOTAL BUDGET - TOTAL AMOUNT OF CURRENT) USING (($$$.$99))
END_REPORT
END_PROCEDURE
  
```

Running the procedure produces the report:

DTR> :CHECK\_RPT

8-Nov-1982  
Page 1

CATEGORY	DESC	CHECK DATE	AMOUNT	BUDGET	DIFFERENCE
1	EDUCATION	15-Jun-1982	\$123.00		
		Category 1 Totals:	\$123.00	\$500.00	\$377.00
2	RECREATION	15-May-1982	\$134.00		
		Category 2 Totals:	\$134.00	\$25.00	(\$109.00)
3	CLOTHING	15-Oct-1982	\$243.00		
		Category 3 Totals:	\$243.00	\$250.00	\$7.00
4	FOOD	15-Jan-1982	\$126.00		
		15-Aug-1982	\$234.00		
		17-May-1982	\$239.00		
		16-May-1982	\$543.00		
		Category 4 Totals:	\$1,142.00	\$150.00	(\$992.00)
5	MORTGAGE				
		Category 5 Totals:	\$ .00	\$550.00	\$550.00
		TOTALS:	\$1,642.00	\$1,475.00	(\$167.00)

DTR>



Inner print lists provide a convenient way to bring an additional record source into the report. DATATRIEVE formats this data as it would a list field in a hierarchical record. You can use this technique when an inner join, or CROSS, does not bring in all the necessary records.

You can also use inner print lists to retrieve list items from a hierarchical view. The next section illustrates this by showing another way to produce the budget and expense report.

### 3.3.3 Using Inner Print Lists with a Hierarchical View

The budget and expense report described in the last section brought two separate record streams together in the PRINT statement. Another approach would be to define a hierarchical view domain that specified the two record streams. You still need to use inner print lists to provide a context for the fields of the dependent stream.

Here is the view domain you could define:

```
DTR> SHOW CHECK_VIEW
DOMAIN CHECK_VIEW OF CATS, CHECKS USING
01 CATS_INFO OCCURS FOR CATS.
  03 CATEGORY FROM CATS.
  03 DESC FROM CATS.
  03 BUDGET FROM CATS.
  03 CHECK_INFO OCCURS FOR CHECKS WITH CAT = CATEGORY AND
      CHECK_DATE AFTER "01-JAN-82".
    05 NUM FROM CHECKS.
    05 CHECK_DATE FROM CHECKS.
    05 AMOUNT FROM CHECKS.
```

The parent record stream consists of each record of the CATS domain. The child, or dependent, stream consists of the CHECKS records that match on the category. This stream is very much like a list in a hierarchical record. The name for the list field is CHECK\_INFO. Here is the report procedure:

```
DTR> SHOW CHECK_VIEW_RPT
PROCEDURE CHECK_VIEW_RPT
READY CHECK_VIEW
DECLARE TOT_AMT COMPUTED BY TOTAL AMOUNT OF CHECK_INFO
  EDIT_STRING IS $$,$$$$.99.
REPORT CHECK_VIEW
PRINT COL 1, CATEGORY, COL 10, DESC, COL 25, ALL CHECK_DATE,
  COL 42, AMOUNT OF CHECK_INFO
AT BOTTOM OF CATEGORY PRINT COL 20,
  "Category"!!!CATEGORY!!!"Totals:", COL 40, TOT_AMT,
  COL 54, BUDGET, COL 70,
  (BUDGET - TOT_AMT) ("DIFFERENCE") USING (($$$$.99)), SKIP
```

```
AT BOTTOM OF REPORT PRINT SKIP, COL 28, "TOTALS:",  
  COL 40, TOTAL TOT_AMT USING $$,$$$ .99,  
  COL 53, TOTAL BUDGET USING $$$$,$$$ .99,  
  COL 70, (TOTAL BUDGET - TOTAL TOT_AMT) USING (($$$ .99))  
END_REPORT  
END_PROCEDURE
```

The procedure is quite similar to the previous one but note several differences.

First, the inner print list refers to the list field in the view, CHECK\_INFO. This inner print list is used to display data on the relevant checks:

```
ALL CHECK_DATE, AMOUNT OF CHECK_INFO
```

Second, this procedure declares a variable, TOT\_AMT, to compute the TOTAL of the AMOUNT field for a list. This statistical expression ranges over all list items within the current occurrence of the list:

```
TOTAL AMOUNT OF CHECK_INFO
```

This variable is used in several ways in the procedure:

- In the AT BOTTOM OF CATEGORY statement, TOT\_AMT gives the total of the AMOUNT field for the *current* occurrence of the list.
- In the AT BOTTOM OF REPORT statement, TOTAL TOT\_AMT gives the total of the AMOUNT field for *all* occurrences of the list. (It is a total of list totals.) The value expression, TOTAL TOT\_AMT, is also used in the computation of the overall difference between expenses and the budget.

The output from this procedure is identical to that shown in the previous section. See the section on reporting hierarchical records for more examples of reporting data from hierarchies.

### 3.4 Printing a Title Page and Other Special Headings

You can enhance your reports with attractive title pages and headings. Use AT TOP OF REPORT and AT TOP OF PAGE statements to produce these effects.

#### 3.4.1 Printing a Title Page

The AT TOP OF REPORT statement allows you to produce a title page before any detail lines are printed. When you use this statement, you suppress the default report heading on the first page of the report. This enables you to design the first page as a special title page without detail lines. The next page includes the report and column headers and is numbered Page 1.



The following example shows how to produce a title page for a report as well as create the report that shows the salaries of employees at Bock's Yachts from departments D98, E46, and T32. The title page includes the company name, the warning CONFIDENTIAL: FOR OFFICIAL EYES ONLY, and the company motto. In the example, the date and page number do not appear until the next page of the report.

Follow these steps:

1. Identify the data for the report in the REPORT statement.
2. Include NEW\_PAGE as the last print item of your AT TOP OF REPORT statement. This ensures that the first detail lines do not appear until the next page.

```

DTR) SET NO PROMPT
DTR) REPORT PERSONNEL WITH DEPT = "D98", "E46", "T32"-----(1)
RW) AT TOP OF REPORT PRINT SKIP 15, COL 20,
RW)      "*****", SKIP,
RW)      COL 20, "*", COL 56, "*", SKIP,
RW)      COL 20, "*", COL 56, "*", SKIP,
RW)      COL 20, "*", COL 32, "SALARY REPORT", COL 56, "*", SKIP,
RW)      COL 20, "*", COL 56, "*", SKIP,
RW)      COL 20, "*", COL 30, "FOR BOCK'S YACHTS", COL 56, "*", SKIP,
RW)      COL 20, "*", COL 56, "*", SKIP,
RW)      COL 20, "*", COL 56, "*", SKIP, COL 20,
RW)      "*****", SKIP 3,
RW)      COL 32, "CONFIDENTIAL:", SKIP,
RW)      COL 32, "FOR OFFICIAL", SKIP,
RW)      COL 33, "EYES ONLY", SKIP 5,
RW)      COL 20, "-----", SKIP 3,
RW)      COL 22, "OUR MOTTO: A YACHT FOR EVERY WORKER", SKIP 3,
RW)      COL 20, "-----", NEW_PAGE-----(2)
RW) SET COLUMNS_PAGE = 70
RW) PRINT ID, STATUS, FIRST_NAME LAST_NAME ("NAME"),
RW)      DEPT, SALARY
RW) END_REPORT

```

Figure 3-7 illustrates the title page. The body of the report follows the title page.

\*\*\*\*\*  
\*  
\*  
\* SALARY REPORT \*  
\* FOR BOCK'S YACHTS \*  
\*  
\*  
\*\*\*\*\*

CONFIDENTIAL:  
FOR OFFICIAL  
EYES ONLY

-----  
OUR MOTTO: A YACHT FOR EVERY WORKER  
-----

**Figure 3-7: Sample Title Page for a Report**

ID	STATUS	NAME	DEPT	SALARY
02943	EXPERIENCED	CASS TERRY	D98	\$29,908
34456	TRAINEE	HANK MORRISON	T32	\$30,000
38462	EXPERIENCED	BILL SWAY	T32	\$54,000
38465	EXPERIENCED	JOE FREIBURG	E46	\$23,908
39485	EXPERIENCED	DEE TERRICK	D98	\$55,829
48475	EXPERIENCED	GAIL CASSIDY	E46	\$55,407
48573	TRAINEE	SY KELLER	T32	\$31,546
49843	TRAINEE	BART HAMMER	D98	\$26,392
83764	EXPERIENCED	JIM MEADER	T32	\$41,029
84375	EXPERIENCED	MARY NALEVO	D98	\$56,847

DTR>

### 3.4.2 Printing Special Page Headings

The AT TOP OF PAGE statement lets you print special headings for your report. You are not limited to the two line heading at the top of the page. But when you use this statement, you are suppressing the default report and column headings on every page. If you want either of these headings on the page, you must include REPORT\_HEADER or COLUMN\_HEADER in the print list for the AT TOP OF PAGE statement.

The following example produces the salary report for Bock's yachts without a title page but with the report heading: "SALARY REPORT FOR BOCK'S YACHTS", surrounded by asterisks. It includes the date and page number, as well as the appropriate column headings:

```
DTR> REPORT PERSONNEL WITH DEPT = "D98", "E46", "T32"
RW> SET COLUMNS_PAGE = 70
RW> AT TOP OF PAGE PRINT REPORT_HEADER, SKIP 2,
RW> COL 20, "*****",SKIP,
RW> COL 20, "", COL 56, "",SKIP,
RW> COL 20, "", COL 56, "",SKIP,
RW> COL 20, "", COL 32, "SALARY REPORT", COL 56, "",
RW> COL 20, "", COL 56, "", SKIP,
RW> COL 20, "", COL 30, "FOR BOCK'S YACHTS",COL 56, "",
RW> COL 20, "", COL 56, "", SKIP,
RW> COL 20, "", COL 56, "", SKIP,
RW> COL 20, "*****",
RW> SKIP 3, COLUMN_HEADER
RW> PRINT ID, STATUS, FIRST_NAME!!!LAST_NAME ("NAME"),
RW> DEPT, SALARY
RW> END_REPORT
```

```

*****
*
*
*           SALARY REPORT
*
*           FOR BOCK'S YACHTS
*
*
*****

```

ID	STATUS	NAME	DEPT	SALARY
02943	EXPERIENCED	CASS TERRY	D98	\$29,908
34456	TRAINEE	HANK MORRISON	T32	\$30,000
38462	EXPERIENCED	BILL SWAY	T32	\$54,000
38465	EXPERIENCED	JOE FREIBURG	E46	\$23,908
39485	EXPERIENCED	DEE TERRICK	D98	\$55,829
48475	EXPERIENCED	GAIL CASSIDY	E46	\$55,407
48573	TRAINEE	SY KELLER	T32	\$31,546
49843	TRAINEE	BART HAMMER	D98	\$26,392
83764	EXPERIENCED	JIM MEADER	T32	\$41,029
84375	EXPERIENCED	MARY NALEVO	D98	\$56,847

DTR)

Note that the `REPORT_HEADER` print item produces both the date and page number. There is no report name on the same line as the date because the `SET REPORT_NAME` statement is not used. Then, after skipping two lines, the Report Writer prints the special heading as specified in the `AT TOP OF PAGE` statement. If this were a multipage report, each successive page would have this heading.

Using this technique, you can design a special billing form with the company heading, with labels for charges and charge descriptions. If the data for each bill is in a single record, you can include `NEW_PAGE` as the last print item in the `PRINT` statement. Then each page contains one customer's bill.

You can also apply this technique to replicate an existing standard form. Specify the appropriate character strings, column numbers, and field values as print list items. Finally, include `NEW_PAGE` as the last print-list item.

### 3.5 Printing Row Totals

Sometimes reports require totals across the fields of a detail line. For example, you might have a payroll record indicating gross salary and deductions. You want to design a payroll report to print a detail line for each employee, indicating gross salary, deductions, and net salary. To compute net salary, you need to subtract the deductions from the gross salary for each detail line.

Though DATATRIEVE does not have an operator to total rows, as it does for columns, it is often possible to write a report specification that generates row totals. Specify an additional print item for the detail line by indicating the formula for net salary. The Report Writer then produces a column of totals for each row.

Figure 3-8 shows the structure of the record WAGES\_REC for the WAGES domain. WAGES is used in the example that follows. (See Appendix A for the complete record definition.)

01 WAGE				
05 LAST_NAME	05 GROSS_PAY	05 FICA	05 STATE_TAX	05 FEDERAL_TAX

MK-01126-00

**Figure 3-8: Field Structure of WAGES\_REC**

The following example shows how the Famous Tech Writers' School uses the WAGES domain to display a report showing each employees weekly wages, deductions, and net pay. The totals of each of the following are displayed at the bottom of the report: gross pay, FICA, state tax, federal tax, and net pay.

To solve this problem, follow these steps:

1. Use this formula for the net pay column:

$GROSS\_PAY - (FICA + FEDERAL\_TAX + STATE\_TAX)$

2. Compute the totals of the field values with an AT BOTTOM OF REPORT statement.



```

DTR> SET NO PROMPT
DTR> REPORT WAGES
RW> SET REPORT_NAME = "FAMOUS TECH WRITERS' SCHOOL"/
RW>   "WEEKLY WAGE REPORT"
RW> SET COLUMNS_PAGE = 70
RW> PRINT LAST_NAME, GROSS_PAY, FICA,
RW> FEDERAL_TAX, STATE_TAX,
RW> GROSS_PAY - (FICA + FEDERAL_TAX + STATE_TAX) ("NET PAY") USING _____(1)
RW>   $$,$$$ .99
RW> AT BOTTOM OF REPORT PRINT SKIP 2, COL 1, "TOTAL:", _____(2)
RW>   TOTAL GROSS_PAY USING $$$,$$$ .99,
RW>   TOTAL FICA USING $$$,$$$ .99,
RW>   TOTAL FEDERAL_TAX USING $$$,$$$ .99,
RW>   TOTAL STATE_TAX USING $$$,$$$ .99,
RW>   TOTAL (GROSS_PAY - (FICA + FEDERAL_TAX + STATE_TAX)) USING
RW>   $$$,$$$ .99
RW> END_REPORT

```

FAMOUS TECH WRITERS' SCHOOL  
WEEKLY WAGE REPORT

19-Apr-1984  
Page 1

LAST NAME	GROSS PAY	FICA	FEDERAL TAX	STATE TAX	NET PAY
BLAKESLEY	\$1,000.00	\$103.86	\$204.77	\$.01	\$691.36
JAMES	\$1,500.00	\$145.87	\$297.98	\$54.32	\$1,001.83
HILLS	\$500.00	\$52.93	\$79.75	\$32.98	\$334.34
JONES	\$999.99	\$103.85	\$204.76	\$57.90	\$633.48
MEADE	\$1,900.98	\$145.87	\$375.98	\$75.90	\$1,303.23
NAPRAVA	\$9,500.00	\$145.87	\$999.84	\$106.90	\$8,247.39
TOTAL:	\$15,400.97	\$698.25	\$2,163.08	\$328.01	\$12,211.63

DTR>

An alternate solution is to edit the record definition to define a new COMPUTED BY field, NET\_PAY. Then include NET\_PAY as one of the print items in the PRINT statement. The following is a sample field definition:

```

10 NET_PAY      COMPUTED BY
  (GROSS_PAY - (FICA + FEDERAL_TAX + STATE_TAX))
  EDIT_STRING IS $$$,$$$ .99.

```

This approach saves typing if you need the value for the net salary in several different reports. Then if the formula changes, you have to change it in only one place—the record definition.

### 3.6 Reporting Hierarchical Records

Hierarchical records are records that contain a list. The list field specifies the number of items in the list with an OCCURS clause. Each list item is subordinate to the list field. The list items are on a lower logical level than the other fields of the record.

Figure 3-9 illustrates the structure of the hierarchical record EMP\_REC. Each record contains data on the previous jobs held by a particular employee. Data on each job is stored as an item of the JOB\_HISTORY list. (See Appendix A for the complete record definition.)

01 EMP_REC			
03 NAME		03 NUMBER_JOBS	03 JOB_HISTORY OCCURS 0 TO 9 TIMES DEPENDING ON NUMBER_JOBS
05 FIRST_NAME	05 LAST_NAME		
		05 OLD_JOB 05 OLD_DATE 05 OLD_JOB 05 OLD_DATE . . . . 05 OLD_JOB 05 OLD_DATE	

MK-01127-00

**Figure 3-9: Field Structure of EMP\_REC**

The OCCURS clause of a hierarchical record designates either a fixed-length or a variable-length list. For variable-length lists, the list field's definition includes an OCCURS DEPENDING clause. This indicates that the number of items in the list depends on the value of another field. For example, the length of the JOB\_HISTORY list in EMP\_REC depends on the value stored in NUMBER\_JOBS.

If you have a domain like EMPLOYEE with hierarchical records, you may want access to individual items from the list to compare their values or to find associated values in a table. In this example, each list item has two components: the code for the old job (OLD\_JOB) and the date the job began (OLD\_DATE). The domain table JOB\_TITLE\_TABLE contains translations for job codes stored in OLD\_JOB. (For more information on using DATATRIEVE tables, see the *VAX DATATRIEVE Handbook*.)

There are several ways to provide DATATRIEVE with the proper context for each list item to search the JOB\_TITLE\_TABLE:

- Use nested FOR statements outside of the Report Writer to provide context for DATATRIEVE. Now you can access list items in the same way as other elementary fields.

- Use the CROSS clause to join the domain with the field controlling the list. This puts every field on the same logical level and creates virtual records with one list item per record.
- Use SET SEARCH to activate the DATATRIEVE Context Searcher and then report on that data.
- Use the REPORT statement, providing context by means of inner print lists in the PRINT statement.

The following sections describe each of these in more detail. Each section shows how Abacus Analog Associates could use the JOB\_HISTORY field of the domain EMPLOYEE to create a report that displays the name and job history of each employee. The report also displays the job code and job title for each old job as contained in the JOB\_TITLE\_TABLE.

### 3.6.1 Using Nested FOR Statements to Flatten the Hierarchy

Nested FOR statements flatten the record, giving you access to each OLD\_JOB list item subordinate to the list field JOB\_HISTORY. This construction is not possible within the Report Writer.

These statements provide context for the list items in the following way. The first FOR statement, FOR EMPLOYEE, instructs DATATRIEVE to process the records from EMPLOYEE, one record at a time. The second FOR statement, FOR JOB\_HISTORY, asks DATATRIEVE to process JOB\_HISTORY from a record one list item at a time.

```
DTR> FOR EMPLOYEE
[Looking for statement]
CON> FOR JOB_HISTORY
[Looking for statement]
CON> PRINT NAME, OLD_JOB,
[Looking for next element in list]
CON> (OLD_JOB VIA JOB_TITLE_TABLE) ("JOB"/"TITLE"), OLD_DATE
```

(continued on next page)

LAST NAME	FIRST NAME	OLD JOB	JOB TITLE	EFFECTIVE DATE
FOSTER	DANA	A03	SENIOR ACCOUNTANT	12-Dec-1981
FOSTER	DANA	A02	INTERNAL AUDITOR	11-Dec-1980
FOSTER	DANA	A01	ACCOUNTANT	10-Dec-1979
MOODY	JOAN	M03	MANUFACTURING MGR	12-Nov-1981
MOODY	JOAN	M03	MANUFACTURING MGR	14-Nov-1980
MOODY	JOAN	M03	MANUFACTURING MGR	12-Oct-1979
MOODY	JOAN	M02	ASSEMBLER	11-Nov-1978
MOODY	JOAN	M01	APPRENTICE	21-Oct-1977
MOODY	JOAN	M01	APPRENTICE	22-Oct-1976
CASADAY	JULIAN	A02	INTERNAL AUDITOR	10-Jan-1982
CASADAY	JULIAN	A01	ACCOUNTANT	9-Jan-1981
DENN	RONALD	M03	MANUFACTURING MGR	12-Dec-1981
DENN	RONALD	M02	ASSEMBLER	14-Dec-1980
DENN	RONALD	M01	APPRENTICE	11-Dec-1979
DENN	RONALD	M01	APPRENTICE	10-Dec-1978
DEPALMA	LOUISE	S03	MARKETING ANALYST	11-Jan-1982
DEPALMA	LOUISE	S02	SALES MANAGER	10-Jan-1981
DEPALMA	LOUISE	S02	SALES MANAGER	11-Dec-1979

DTR)

### 3.6.2 Using CROSS to Flatten the Hierarchy

It is easier to deal with flat records than hierarchical records because with flat records all of the fields are on the same level. Use the CROSS clause to flatten the hierarchy, joining each item in the list with the other fields of the record. Then you can specify the list items in PRINT, AT TOP, or AT BOTTOM statements.

The procedure FLATTEN uses the CROSS clause to join each record of EMPLOYEE with each item of the list JOB\_HISTORY:

```
DTR) SHOW FLATTEN
PROCEDURE FLATTEN
READY EMPLOYEE
REPORT EMPLOYEE CROSS JOB_HISTORY
SET COLUMNS_PAGE = 70
SET REPORT_NAME = "EMPLOYEE HISTORY REPORT"
PRINT NAME, OLD_JOB,
      (OLD_JOB VIA JOB_TITLE_TABLE)("JOB"/"TITLE"),
      OLD_DATE ("EFFECTIVE"/"DATE")
END_REPORT
END_PROCEDURE
```

This procedure generates a report like the one in the section on using nested FOR statements, but with the addition of the Report Writer headings and formatting.



### 3.6.3 Accessing List Items with the SET SEARCH Command

You can use the Report Writer to report hierarchical records by activating the Context Searcher. Use the SET SEARCH command. The Context Searcher is able to locate each OLD\_JOB entry, even though each entry is embedded within a list. Here is the procedure HIER\_REPORT that produces the report:

```
DTR> SHOW HIER_REPORT
PROCEDURE HIER_REPORT
READY EMPLOYEE
SET SEARCH
REPORT EMPLOYEE
SET COLUMNS_PAGE = 70
SET REPORT_NAME = "EMPLOYEE HISTORY REPORT"
PRINT NAME, OLD_JOB,
      (OLD_JOB VIA JOB_TITLE_TABLE)("JOB"/"TITLE"), OLD_DATE
END_REPORT
FINISH EMPLOYEE
END_PROCEDURE
```

Running the procedure produces a message from the Context Searcher, followed by the report:

```
DTR> :HIER_REPORT
```

Not enough context. Some field names resolved by Context Searcher.

```
                EMPLOYEE HISTORY REPORT                28-Apr-1982
                                                         Page 1
```

LAST NAME	FIRST NAME	OLD JOB	JOB TITLE	EFFECTIVE DATE
FOSTER	DANA	A03	SENIOR ACCOUNTANT	12-Dec-1981
		A02	INTERNAL AUDITOR	11-Dec-1980
		A01	ACCOUNTANT	10-Dec-1979
MOODY	JOAN	M03	MANUFACTURING MGR	12-Nov-1981
		M03	MANUFACTURING MGR	14-Nov-1980
		M03	MANUFACTURING MGR	12-Oct-1979
		M02	ASSEMBLER	11-Nov-1978
		M01	APPRENTICE	21-Oct-1977
		M01	APPRENTICE	22-Oct-1976
CASADAY	JULIAN	A02	INTERNAL AUDITOR	10-Jan-1982
		A01	ACCOUNTANT	9-Jan-1981
DENN	RONALD	M03	MANUFACTURING MGR	12-Dec-1981
		M02	ASSEMBLER	14-Dec-1980
		M01	APPRENTICE	11-Dec-1979
		M01	APPRENTICE	10-Dec-1978
DEPALMA	LOUISE	S03	MARKETING ANALYST	11-Jan-1982
		S02	SALES MANAGER	10-Jan-1981
		S02	SALES MANAGER	11-Dec-1979

```
DTR>
```



This approach differs from the first solution because it does not flatten the hierarchy. Each name appears just once. Only the fields within the list have multiple occurrences, depending on the entry made for NUMBER\_JOBS.

The Context Searcher provided DATATRIEVE with the proper context. For more information about context in DATATRIEVE, see the *VAX DATATRIEVE User's Guide*.

### 3.6.4 Using the REPORT Statement to Report List Data

You can produce the employee history report with the REPORT statement without invoking the Context Searcher or flattening the hierarchy. However, you must provide the proper context for DATATRIEVE within the PRINT statement by using inner print lists for the list items.

The following PRINT statement uses inner print lists to specify the relationship between the list items and the list field:

```
PRINT NAME, ALL OLD_JOB, OLD_JOB VIA JOB_TITLE_TABLE,  
        OLD_DATE OF JOB_HISTORY
```

HIER\_REPORT2, which includes this PRINT statement, contains the report specification for the employee history report:

```
DTR> SHOW HIER_REPORT2  
PROCEDURE HIER_REPORT2  
READY EMPLOYEE  
REPORT EMPLOYEE  
SET COLUMNS_PAGE = 70  
SET REPORT_NAME = "EMPLOYEE HISTORY REPORT"  
PRINT NAME, ALL OLD_JOB, OLD_JOB VIA JOB_TITLE_TABLE,  
        OLD_DATE OF JOB_HISTORY  
END_REPORT  
END_PROCEDURE
```

The output is the same as the previous report.

### 3.7 Printing a Variety of Detail Lines in One Report

As you become more proficient with the DATATRIEVE Report Writer, you will probably experiment with progressively more complex report specifications. Sometimes you may want to print different types of detail lines in the same report. For example, you might want to print a column indicating whether a salesperson is experienced or a trainee, depending on the number of months on the sales force. A logical approach would be to test the value of MONTHS\_EMP. If MONTHS\_EMP is greater than 6, DATATRIEVE should print an “experienced worker” detail line. Otherwise, DATATRIEVE should print a “trainee” detail line. But you cannot include conditional statements or more than one PRINT statement within a report specification.

To generate this type of report with the DATATRIEVE Report Writer, you must take a different approach. This section presents two ways to solve this type of problem, each using the CHOICE value expression.

Figure 3-10 shows the logical structure of the record SALES\_REC for the SALES domain. SALES is used in the example that follows. (See Appendix A for the complete record definition.)

01 SALESREC				
05 ID	05 SALES_NAME	05 START_DATE	05 MONTHS_EMP	05 AMOUNT

MK-01122-00

**Figure 3-10: Field Structure of SALES\_REC**

In the following example, the Acme Computer Company divides its sales force into two categories. Trainees are those who have been employed for fewer than six months. Experienced workers have been employed for six months or more. Each salesperson’s commission depends on how long he or she has been employed as well as the amount sold, as illustrated in Table 3-1.

**Table 3-1: Commission Schedule for the Sales Division**

Months Employed	Amount Sold	Commission Percent	Rating
GT 6	GT 10000	12%	Above quota
GT 6	LE 10000	7%	Below quota
LE 6	GT 5000	10%	Above quota
LE 6	LE 5000	5%	Below quota

The report displays the name, months employed, total sales, commission percentage, total commission, and rating (above quota or below quota) for each salesperson.

An analysis of the report requirements shows that you need six values for each detail line of the report:

- Three desired values are field values contained in the input record: the salesperson's name (SALES\_NAME), months employed (MONTHS\_EMP), and amount sold (AMOUNT).
- Two other values must be assigned depending on the months employed and sales amount: the salesperson's rating (above quota or below quota) and the commission percentage.
- The final value, total commission, can be computed from the values for AMOUNT and COMM\_PCT. Use the following formula:  $(AMOUNT * COMM\_PCT) / 100$ . You can either add a new COMPUTED BY field or variable (COMMISSION), or you can include the formula directly in the PRINT statement.

In effect, you can print a detail line for a salesperson only after testing for months employed and total sales. This problem is representative of a common need within report writing: testing field values to derive new values and printing both the field and the derived values on the same detail line.

To solve this type of problem, use the CHOICE value expression to conduct a series of tests for each record based on values of MONTHS\_EMP and AMOUNT. You can do this in either of two ways:

- Edit the original record definition to set up two new COMPUTED BY fields for COMM\_PCT and RATING. Use the CHOICE value expression in the COMPUTED BY clause.
- Use the CHOICE value expression within the PRINT statement. DATATRIEVE tests for the values of MONTHS\_EMP and AMOUNT while processing each record.

### 3.7.1 CHOICE Value Expression in COMPUTED BY Fields

One way to solve the testing problem is to edit the record definition before invoking the Report Writer. You need to add two new COMPUTED BY fields: COMM\_PCT and RATING. Use the CHOICE value expression within the COMPUTED BY clause. (For more information on the CHOICE value expression, see the *VAX DATATRIEVE Reference Manual*).

Because these are virtual fields whose values are not actually stored, you are not changing the size of the record. Therefore, there is no need to restructure the domain. The following shows the resulting change in SALES\_REC:

```
DTR) SHOW SALES_REC
RECORD SALES_REC USING
01 SALESREC.
    05 ID          PIC IS 9(5).
    05 SALES_NAME PIC IS X(20).
    05 START_DATE  USAGE IS DATE.
    05 MONTHS_EMP  COMPUTED BY ("TODAY" - START_DATE)/(30)
                        EDIT_STRING IS ZZ9.
    05 AMOUNT PIC IS 9(5)V99
                        QUERY_NAME IS AMT
                        EDIT_STRING IS $$$,$$$ .99.
    05 COMM_PCT    COMPUTED BY
                        CHOICE
                        (MONTHS_EMP LE 6 AND AMOUNT > 5000) THEN 10
                        (MONTHS_EMP LE 6) THEN 05
                        (AMOUNT > 10000) THEN 12
                        ELSE 07
                        END_CHOICE
                        EDIT_STRING IS Z9% .
    05 RATING      COMPUTED BY
                        CHOICE
                        (MONTHS_EMP LE 6 AND AMOUNT > 5000) THEN "ABOVE QUOTA"
                        (AMOUNT > 10000) THEN "ABOVE QUOTA"
                        ELSE "BELOW QUOTA"
                        END_CHOICE
                        EDIT_STRING IS X(11).
```

Figure 3-11 shows the new structure of SALES\_REC with the addition of the two COMPUTED BY fields.

01 SALESREC						
05 ID	05 SALES_NAME	05 START_DATE	05 MONTHS_EMP	05 AMOUNT	05 COMM_PCT	05 RATING

MK-01129-00

**Figure 3-11: Revised Field Structure of SALES\_REC**

Follow these steps to produce a control group report on sales commission:

1. Declare a variable to compute the commission.
2. Report the records in SALES and sort them by the new field COMM\_PCT. Choosing COMM\_PCT as the sort key enables you to break up the detail lines into four groups. This corresponds to the four possible commission percentages.
3. Name the report.

4. At the top of each group, print the values for COMM\_PCT and RATING.
5. Print the field values, specifying appropriate column headers where necessary.
6. Summarize the data about the sales personnel in each commission-percentage category with an AT BOTTOM OF COMM\_PCT statement.
7. Summarize the data about the entire sales force with an AT BOTTOM OF REPORT statement.

The following is the procedure COMM\_REPORT that produces the desired report:

```

DTR> SHOW COMM_REPORT
PROCEDURE COMM_REPORT
READY SALES
DECLARE COMMISSION COMPUTED BY _____(1)
      ((AMOUNT * COMM_PCT) / 100)
      EDIT_STRING IS $$$,$$$.$99.
REPORT SALES SORTED BY COMM_PCT _____(2)
SET REPORT_NAME = "SALES COMMISSION REPORT" _____(3)
SET COLUMNS_PAGE = 70
AT TOP OF COMM_PCT PRINT RATING, COMM_PCT _____(4)
PRINT SALES_NAME, MONTHS_EMP, AMOUNT, COMMISSION ("COMMISSION") _____(5)
AT BOTTOM OF COMM_PCT PRINT SKIP, COL 19, _____(6)
  "NUMBER:", SPACE, COUNT(-) USING Z9,
  COL 35, "TOTAL SALES:", TOTAL AMOUNT USING
  $$$,$$$.$99, TOTAL COMMISSION USING $$$,$$$.$99, SKIP 2
AT BOTTOM OF REPORT PRINT COL 5, _____(7)
  "*****",
  SKIP 2, COL 14, "SALES FORCE:", SPACE, COUNT(-) USING Z9, COL 35,
  "TOTAL SALES:", TOTAL AMOUNT USING $$$,$$$.$99,
  TOTAL COMMISSION USING $$$,$$$.$99
END_REPORT
END_PROCEDURE

```

Figure 3-12 shows the report produced by COMM\_RPT.



RATING	COMM PCT	SALES NAME	MONTHS EMP	AMOUNT	COMMISSION
BELOW QUOTA 5%					
		ANNE DINNAN	3	\$2,389.90	\$119.50
		RICK LANGHART	4	\$4,999.99	\$250.00
		LYDIA BARNET	1	\$2,598.79	\$129.94
		JOSEPH FREDERICK	4	\$5,000.00	\$250.00
		NUMBER: 4	TOTAL SALES:	\$14,988.68	\$749.53
BELOW QUOTA 7%					
		WILLIAM SULLIVAN	9	\$8,672.99	\$607.11
		LINDA REINE	7	\$8,532.22	\$597.26
		HENRY MAILER	7	\$9,999.99	\$700.00
		NUMBER: 3	TOTAL SALES:	\$27,205.20	\$1,904.36
ABOVE QUOTA 10%					
		NANCY ROTHBLATT	2	\$6,325.88	\$632.59
		WAYNE SMITH	5	\$9,853.52	\$985.35
		SEYMOUR KIMMELMAN	5	\$7,325.67	\$732.57
		NUMBER: 3	TOTAL SALES:	\$23,505.07	\$2,350.51
ABOVE QUOTA 12%					
		DAN DERRICK	8	\$11,456.87	\$1,374.82
		JAMES STORER	14	\$25,876.02	\$3,105.12
		SANDY LEVINE	8	\$10,000.01	\$1,200.00
		DENNIS MCADOO	11	\$12,345.62	\$1,481.47
		NUMBER: 4	TOTAL SALES:	\$59,678.52	\$7,161.42

\*\*\*\*\*

SALES FORCE: 14      TOTAL SALES: \$125,377.47    \$12,165.73

DTR>

### Figure 3-12: Control Group Report with Variety of Detail Lines

Note that if you try to duplicate this report, the results may differ because the COMM\_PCT field is based on length of employment, which is in turn dependent on the date the report is produced.

### 3.7.2 CHOICE Value Expression Within a PRINT Statement

If you do not edit the record definition, you can still produce a sales commission report by using the CHOICE value expression in the Report Writer PRINT statement. You need to use it three times: for determining the rating, the commission percentage, and the commission. Because these are not fields in the record definition, you must also specify appropriate headings and edit strings.

When you use the CHOICE value expression to calculate the commission, you use a slightly different formula than before. You can multiply AMOUNT by the decimal equivalent of the commission percentage. Hence, there is no need to divide the result by 100.

Because you have not defined a field for commission percentage, you cannot produce a control group report sorted by the values for commission percentage. Therefore, the report specification does not have an AT BOTTOM OF field-name statement. However, you can still generate the same detail lines as before with the PRINT statement.

The following procedure SALES\_RPT uses the CHOICE value expression three separate times in the same PRINT statement:

```
DTR> SHOW SALES_RPT
PROCEDURE SALES_RPT
READY SALES
REPORT SALES
SET REPORT_NAME = "SALES COMMISSION REPORT"
PRINT SALES_NAME, MONTHS_EMP, AMOUNT,
      CHOICE
      (MONTHS_EMP LE 6 AND AMOUNT > 5000) THEN 10
      (MONTHS_EMP LE 6) THEN 05
      (AMOUNT > 10000) THEN 12
      ELSE 07
END_CHOICE ("COMM"/"PCT") USING Z9%,
CHOICE
      (MONTHS_EMP LE 6 AND AMOUNT > 5000) THEN (.1 * AMOUNT)
      (MONTHS_EMP LE 6) THEN (.05 * AMOUNT)
      (AMOUNT > 10000) THEN (.12 * AMOUNT)
      ELSE (.07 * AMOUNT)
END_CHOICE ("COMMISSION") USING $$,$$$$.99,
CHOICE
      (MONTHS_EMP LE 6 AND AMOUNT > 5000) THEN "ABOVE QUOTA"
      (AMOUNT > 10000) THEN "ABOVE QUOTA"
      ELSE "BELOW QUOTA"
END_CHOICE ("RATING")
END_REPORT
FINISH SALES
END_PROCEDURE
```

Running SALES\_RPT produces the report:

DTR> :SALES\_RPT

SALES COMMISSION REPORT

2-Jul-1982

Page 1

SALES NAME	MONTHS EMP	AMOUNT	COMM PCT	COMMISSION	RATING
ANNE DINNAN	3	\$2,389.90	5%	\$119.50	BELOW QUOTA
NANCY ROTHBLATT	2	\$6,325.88	10%	\$632.59	ABOVE QUOTA
LINDA REINE	7	\$8,532.22	7%	\$597.26	BELOW QUOTA
WAYNE SMITH	5	\$9,853.52	10%	\$985.35	ABOVE QUOTA
JAMES STORER	14	\$25,876.02	12%	\$3,105.12	ABOVE QUOTA
SANDY LEVINE	8	\$10,000.01	12%	\$1,200.00	ABOVE QUOTA
SEYMOUR KIMMELMAN	5	\$7,325.67	10%	\$732.57	ABOVE QUOTA
JOSEPH FREDERICK	4	\$5,000.00	5%	\$250.00	BELOW QUOTA
RICK LANGHART	4	\$4,999.99	5%	\$250.00	BELOW QUOTA
WILLIAM SULLIVAN	9	\$8,672.99	7%	\$607.11	BELOW QUOTA
DAN DERRICK	8	\$11,456.87	12%	\$1,374.82	ABOVE QUOTA
LYDIA BARNET	1	\$2,598.79	5%	\$129.94	BELOW QUOTA
HENRY MAILER	7	\$9,999.99	7%	\$700.00	BELOW QUOTA
DENNIS MCADOO	11	\$12,345.62	12%	\$1,481.47	ABOVE QUOTA

DTR>

These sales commission reports illustrate the flexibility of COMPUTED BY fields and PRINT statements that include the CHOICE value expression. As you consider more complex reports, you may need to test each record to generate detail line items. Using the CHOICE value expression is the most direct way to produce this type of report.

The DATATRIEVE Report Writer can also produce reports using data from VAX DBMS, VAX Rdb/VMS, VAX Rdb/ELN, or VIDA. Several examples are presented in the following chapters.



# Writing DBMS Reports 4

The data management and access capabilities of DATATRIEVE provide you with an easy-to-use query language and report writer for VAX DBMS. If VAX DBMS is installed on your system, you can use the same DATATRIEVE commands and statements for most tasks whether you are working with data stored in RMS files or in DBMS databases.

See the *VAX DATATRIEVE User's Guide* for more basic information about using DATATRIEVE to access records in DBMS databases.

## 4.1 Accessing VAX DBMS Data with DATATRIEVE

To demonstrate accessing and reporting VAX DBMS data with DATATRIEVE, this chapter uses the PARTS sample database included with both VAX DBMS and VAX DATATRIEVE. See the documentation that accompanies VAX DBMS for more details on the PARTS database.

The following command sets the CDD default to the directory that contains the database domain definitions used in this chapter:

```
DTR> SET DICTIONARY CDD$TOP.DTR$LIB.DEMO.DBMS
```

The DBMS domains defined in the specified CDD directory were created when the DATATRIEVE/DBMS UETP (User Environment Test Package) was run. Perform a SHOW command to see what was defined by the UETP:

```
DTR> SHOW PARTS_DB
DATABASE PARTS_DB
  USING SUBSCHEMA DTR_SUBSCHEMA
    OF SCHEMA PARTS
  ON DTR$LIBRARY:DTRPARTDB;
DTR>
```



You can ready the database directly and use DBMS records as sources in your DATATRIEVE queries, or you can ready DBMS domains individually.

There are several DBMS domains that are associated with PARTS\_DB. These are CLASSES, COMPONENTS, DIVISIONS, EMPLOYEES, PART\_S, QUOTES, SUPPLIES, and VENDORS. You can use several SHOW commands to see how these domains were defined. Each of the definitions refers to PARTS\_DB.

The examples in this chapter use DBMS domains. If the DBMS database definition or domains are not in the specified CDD directory, see the person responsible for VAX DATATRIEVE at your site.

## 4.2 Writing a Simple Report with VAX DBMS Data

The DATATRIEVE Report Writer is a useful tool for writing reports with data stored through VAX DBMS. You can define DBMS domains based on records in a VAX DBMS database. Then use the DATATRIEVE data access and formatting capabilities to report on the data.

The following example uses the EMPLOYEES and DIVISIONS domains, defined in DATATRIEVE from the PARTS database. For further information on DBMS domains, see the *VAX DATATRIEVE User's Guide*.

The example shows how to develop a procedure to report personnel information on all the employees in a given division. The total number of employees working in that division is displayed at the bottom of the report.

The information on divisions can be found in the DIVISIONS domain. All personnel information on all employees in all divisions can be found in the EMPLOYEES domain. First, ready the two domains. Second, with a SHOW FIELDS command, display the field structure of the records in these domains:

```
DTR> READY EMPLOYEES, DIVISIONS
DTR> SHOW FIELDS FOR EMPLOYEES, DIVISIONS
EMPLOYEES
  EMPLOYEE
    EMP_ID (ID) <Number>
    EMP_LAST_NAME (LAST_NAME) <Character string>
    EMP_FIRST_NAME (FIRST_NAME) <Character string>
    EMP_PHONE (PHONE_NUMBER) <Number>
    EMP_LOC (LOCATION) <Character string>
DIVISIONS
  DIVISION
    DIV_NAME <Character string>
DTR>
```

In the definition of the procedure to locate employees in a particular division, follow these steps:

1. Use a FIND statement to establish a current collection from DIVISIONS. With a prompt, you can let the user enter the name of the particular division when the procedure is run.
2. Select the record to give DATATRIEVE the proper context for further queries.
3. Report all records of EMPLOYEES connected to that selected record as members of the set CONSISTS\_OF.
4. Use a prompt with SET REPORT\_NAME to let the user supply an appropriate name for the report.
5. Print each record in the specified record stream from EMPLOYEES by indicating the field names and appropriate edit strings.
6. Summarize the number of records with an AT BOTTOM OF REPORT statement.
7. End with an END\_REPORT statement.

Here is the procedure EMPLOYEE\_RPT:

```
DTR> SHOW EMPLOYEE_RPT
PROCEDURE EMPLOYEE_RPT
READY DIVISIONS, EMPLOYEES
FIND DIVISIONS WITH DIV_NAME = *."the division"_____ (1)
SELECT_____ (2)
REPORT EMPLOYEES MEMBER CONSISTS_OF_____ (3)
SET REPORT_NAME = *."the report name enclosed in quotes"_____ (4)
SET COLUMNS_PAGE = 70
PRINT EMP_ID ("Ident"),_____ (5)
    LAST_NAME ("Last"/"Name") USING X(10),
    FIRST_NAME ("First"/"Name") USING X(10),
    EMP_PHONE ("Phone"/"Number") USING XXX_XXXX,
    EMP_LOC ("Loc")
AT BOTTOM OF REPORT SKIP 2,_____ (6)
    COL 30, "TOTAL EMPLOYEES:", SPACE,
    COUNT (-) USING Z9
END_REPORT_____ (7)
FINISH
END_PROCEDURE
DTR>
```

To produce the report, invoke the procedure and respond to the prompts:

```
DTR) :EMPLOYEE_RPT
Enter the division: VT100 DEVELOPMENT
Enter the report name: "VT100 DEVELOPMENT EMPLOYEES"
```

VT100 DEVELOPMENT EMPLOYEES

19-May-1984  
Page 1

Ident	Last Name	First Name	Phone Number	Loc
65437	FRANK	BEBI	456-8901	89012
12333	HOFFMAN	MIKE	456-8901	89012
54332	IGLESIAS	RAFAEL	234-6789	67890

TOTAL EMPLOYEES: 3

DTR)

### 4.3 Developing a Procedure for a Bill-of-Materials Report

Report statements are usually stored as procedures, allowing you to run the procedure whenever necessary. For example, you might frequently generate a *bill of materials* or *parts explosion*. The following statements define a sample procedure that produces a part-component report:

1. Begin with a DEFINE PROCEDURE statement.
2. Ready the PART\_S and COMPONENTS domains.
3. Identify the data and invoke the Report Writer with a REPORT rse statement.
4. Use a prompting value expression to direct the output to a file or device.
5. Use a SET REPORT\_NAME statement to name the report.
6. Indicate the detail line format with a PRINT statement. This statement contains an inner print list:

```
ALL COMP_QUANTITY, ALL PART_DESC OF PART_S OWNER
PART_USED_ON OF COMPONENTS MEMBER PART_USES
```

7. Specify an edit string for the component quantity (COMP\_QUANTITY) so that the column will contain integers instead of the two decimal places defined for the field.



8. End the report specification with an END\_REPORT statement.
9. Clear the workspace by using the FINISH command.
10. Conclude the procedure with an END\_PROCEDURE statement.

```

DTR> DEFINE PROCEDURE COMPONENT_REPORT_____ (1)
DFN> READY PART_S, COMPONENTS_____ (2)
DFN> REPORT PART_S WITH ANY COMPONENTS WITHIN_____ (3)
DFN>     PART_USES ON *."output file or device name"_____ (4)
DFN> SET REPORT_NAME = "LISTING OF"/
DFN>     "PARTS AND COMPONENTS"_____ (5)
DFN> SET COLUMNS_PAGE = 70
DFN> PRINT PART_ID,
DFN>     PART_DESC ("Part"/"Description") USING X(11),_____ (6)
DFN>     SKIP 2, COL 1,
DFN>     ALL COMP_QUANTITY (-) USING ZZ9, COL 10,_____ (7)
DFN>     ALL PART_DESC (-) OF PART_S OWNER
DFN>     PART_USED_ON OF
DFN>     COMPONENTS MEMBER PART_USES, SKIP
DFN> END_REPORT_____ (8)
DFN> FINISH PART_S, COMPONENTS_____ (9)
DFN> END_PROCEDURE_____ (10)
DTR>

```

To produce the report, invoke the procedure COMPONENT\_REPORT:

```

DTR> :COMPONENT_REPORT
Enter output file or device name: TT:

```

```

                LISTING OF                19-May-1984
                PARTS AND COMPONENTS      Page 1

```

	Part Number	Part Description
	TI-6789-01	RK HOUSING
2	3/4 INCH SCREWS	
8	1/4 INCH SCREWS	
	BR-8901-23	LA36
6	1/4 INCH BOLTS	
4	3/8 INCH BOLTS	
6	1/2 INCH SCREWS	
2	3/4 INCH SCREWS	
4	4/5 INCH CLAMP	

```

DTR>

```

## 4.4 Using Control Groups with VAX DBMS Data

As illustrated in the last chapter, the Report Writer can separate the data into control groups based on a sort key. The following example asks you to set up control groups based on the employee's ID (EMP\_ID). Because each ID number is unique, each control group consists of one employee.

A common query might be "Who is responsible for a specific part?" The following example shows how to answer that question by defining a procedure for reporting the names of employees responsible for parts and the parts for which they are responsible. The report will indicate how many parts each employee is responsible for.

Follow these steps:

1. Create a current collection with a FIND statement using the CROSS operator, relating the EMPLOYEES and PART\_S domains.
2. Develop control groups based on an employee's ID (EMP\_ID) with an AT TOP OF EMP\_ID statement.
3. Use concatenation operators to control the spacing between fields. (See the *VAX DATATRIEVE Reference Manual* for more information.)
4. Compute control group totals with an AT BOTTOM OF EMP\_ID statement.

```
DTR> DEFINE PROCEDURE RESP_REPORT
DFN> READY EMPLOYEES, PART_S _____(1)
DFN> FIND EMPLOYEES CROSS PART_S WITHIN RESPONSIBLE_FOR
DFN> REPORT CURRENT ON *."device or file name"
DFN> SET REPORT_NAME = "Responsibility List"
DFN> AT TOP OF EMP_ID PRINT COL 1, _____(2)
DFN>   EMP_FIRST_NAMEEMP_LAST_NAME ("Name"), SKIP _____(3)
DFN> PRINT COL 21, PART_DESC ("Parts")
DFN> AT BOTTOM OF EMP_ID PRINT COL 1, _____(4)
DFN> "A total of "COUNT"parts", SKIP, _____(3)
DFN> "-----"
DFN> END_REPORT
DFN> END_PROCEDURE
DTR>
```



To produce the responsibility report, invoke the procedure RESP\_REPORT:

DTR> RESP\_REPORT

Enter device or file name: TT:

Responsibility List

13-Jan-1984

Page 1

Name	Parts
MIKE HOFFMAN	

TERMINAL TABLE VT100  
VT100 NUMERIC KEYPAD ASSY

A total of 2 parts

-----  
OLA HILL

.47 UF 50V CER CAPACITOR  
1N970B DIODE  
100K 1/4W RESISTOR  
CLIP CONTACTS  
ETCHED BOARD (VT100 KEYBOARD)  
4/5 INCH CLAMP  
1 INCH CLAMP  
1.5 AMP 250V FUSE  
3 AMP 250V FUSE  
12 AMP 250V FUSE  
15 AMP 250V FUSE  
TU60 FLOPPY  
TTY RIBBON  
DISK HEAD CLEANER KIT  
TONER CONCENTRATE  
FU60 FLOPPY CASE

A total of 16 parts  
-----  
:  
:  
:



# Writing Relational Database Reports **5**

DATATRIEVE provides you with an easy-to-use query language and Report Writer for the DIGITAL family of relational database management systems. If VAX Rdb/VMS, VAX Rdb/ELN, or VIDA is installed on your system, you can use the same DATATRIEVE commands and statements for most tasks whether you are working with data stored in RMS files or in the relational databases.

See the chapter on accessing VAX relational databases in the *VAX DATATRIEVE User's Guide* for more basic information about using DATATRIEVE with one of the DIGITAL relational database products.

## **5.1 Accessing VAX Relational Databases with DATATRIEVE**

To demonstrate accessing and reporting relational source data with DATATRIEVE, this chapter uses the PERSONNEL sample database installed with VAX DATATRIEVE. This database is similar to the sample PERSONNEL database installed with VAX Rdb/VMS but contains only a subset of the relations and records in that database.

The following command sets the CDD default to the directory that contains the database domain definitions used in the examples in this chapter:

```
DTR> SET DICTIONARY CDD$TOP.DTR$LIB.DEMO.RDB  
DTR>
```

The relational domains defined in the specified CDD directory were created by the DATATRIEVE installation procedure. To see the sample data, set your default to the proper dictionary and perform a SHOW command:

```
DTR> SHOW ALL
Domains:
COLLEGES;1      DEGREES;1      DEPARTMENTS;1  DEPARTMENT_STAFF;1
EMPLOYEES;1    EMPLOYEE_EDUCATION;1  JOBS;1
JOB_HISTORY;1  SALARY_HISTORY;1  WORK_STATUS;1

Procedures:
EMPLOYEE_INFO;1  READY_PERSONNEL;1  READY_PERSONNEL_WRITE;1
SALARY_REPORT;1

Tables:
DEPARTMENT_TABLE;1      NAME_TABLE;1

Databases:
PERSONNEL
The default directory is CDD$TOP.DTR$LIB.DEMO.RDB
No established collections.
No ready sources.
No loaded tables.
```

Note that the sample directory contains domain definitions for each relation in the PERSONNEL database. To access data in a relational database, you can define and ready domains for each relation. The following example shows the contents of a single domain definition and then readies all the domains in the PERSONNEL database.

```
DTR> SHOW COLLEGES
DOMAIN COLLEGES
        USING COLLEGES OF DATABASE PERSONNEL;

DTR> READY COLLEGES, DEGREES, DEPARTMENTS, EMPLOYEES, -
        JOBS, JOB_HISTORY, SALARY_HISTORY, WORK_STATUS
```

Figure 5-1 shows the relations and fields for the sample PERSONNEL database for which DATATRIEVE domains are defined. Some examples in this chapter refer to the relations and field names in the PERSONNEL database.

EMPLOYEES  EMPLOYEE_ID LAST_NAME FIRST_NAME MIDDLE_INITIAL ADDRESS_DATA STREET TOWN STATE ZIP SEX BIRTHDAY SOCIAL_SECURITY STATUS_CODE	DEGREES  EMPLOYEE_ID COLLEGE_CODE YEAR_GIVEN DEGREE DEGREE_FIELD	JOBS  JOB_CODE WAGE_CLASS JOB_TITLE MINIMUM_SALARY MAXIMUM_SALARY
SALARY_HISTORY  EMPLOYEE_ID SALARY_AMOUNT SALARY_START SALARY_END	JOB_HISTORY  EMPLOYEE_ID DEPARTMENT_CODE JOB_CODE JOB_START JOB_END SUPERVISOR_ID	COLLEGES  COLLEGE_CODE COLLEGE_NAME ADDRESS_DATA STREET TOWN STATE ZIP
	DEPARTMENTS  DEPARTMENT_CODE DEPARTMENT_NAME MANAGER_ID BUDGET_PROJECTED BUDGET_ACTUAL	WORK_STATUS  STATUS_CODE STATUS_NAME STATUS_TYPE

MK-01601-00

**Figure 5-1: Sample Relational Database**

Rather than define domains for each relation, you can define a DATATRIEVE database definition for the relational database and then ready that database definition. The following example shows a database definition and the results of the READY database command. Notice that the relations are readied directly by the READY database command; there are no domain definitions:

```
DTR> DEFINE DATABASE PERSONNEL ON DTR$LIBRARY:PERSONNEL;
DTR> SHOW DATABASES
Databases:
      PERSONNEL;1
```

(continued on next page)



```

DTR> READY PERSONNEL
DTR> SHOW READY
Ready sources:
  WORK_STATUS: Relation, Rdb, snapshot read, consistency
               <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  SALARY_HISTORY: Relation, Rdb, snapshot read, consistency
                 <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  JOB_HISTORY: Relation, Rdb, snapshot read, consistency
              <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  JOBS: Relation, Rdb, snapshot read, consistency
       <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  EMPLOYEES: Relation, Rdb, snapshot read, consistency
            <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  DEPARTMENTS: Relation, Rdb, snapshot read, consistency
               <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  DEGREES: Relation, Rdb, snapshot read, consistency
          <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
  COLLEGES: Relation, Rdb, snapshot read, consistency
           <CDD$TOP.DTR$LIB.DEMO.RDB.PERSONNEL;1>
No loaded tables.

DTR>

```

For more information on how to access relational databases and the advantages or disadvantages of defining domains for each relation, see the *VAX DATATRIEVE User's Guide*. If the relational database definition or domains are not in the specified CDD directory, see the person responsible for VAX DATATRIEVE at your site.

## 5.2 Writing a Simple Report with Relational Data

The DATATRIEVE Report Writer is a useful tool for generating reports with data stored in one of the DIGITAL relational database products. You can define relational domains based on relations in a relational database or access the relations directly. Then you can use the DATATRIEVE data access and formatting capabilities to report on the data.

The following example uses the JOB\_HISTORY and EMPLOYEES domains to print all those employees who work in selected departments. Notice with relational databases that you frequently use the CROSS clause to combine related data stored in one or more relations.

Follow these steps to create this report:

1. Ready the JOB\_HISTORY and EMPLOYEES domains.
2. Invoke the Report Writer and form an RSE of all the employees in the EMPLOYEES domain who currently work in selected departments:
  - The EMPLOYEES domain gives you the employee name and employee ID.

- The CROSS and OVER clauses combine each employee record with a matching record in the JOB\_HISTORY domain. DATATRIEVE forms a new record stream for each employee containing the data from both those relations matched on the EMPLOYEE\_ID field.
  - The Boolean expression, WITH JOB\_END MISSING, restricts the record stream to only those employees who are currently employed. The JOB\_HISTORY record that has no data in the JOB\_END field is the current JOB\_HISTORY record.
  - The Boolean expression, WITH DEPARTMENT = "ADMN", "ELEL" further restricts the record stream to the specified departments.
3. Sort the record stream alphabetically by department name.
  4. Give the report a name with the SET REPORT\_NAME statement.
  5. At the top of each department grouping, print the department code from the JOB\_HISTORY domain.
  6. For each employee in each department, print the employee ID, the name, and the start date for the current job.
  7. End the report specification with an END\_REPORT statement.

```

DTR> READY JOB_HISTORY, EMPLOYEES _____(1)
DTR> REPORT EMPLOYEES CROSS JOB_HISTORY OVER _____(2)
RW>   EMPLOYEE_ID -
RW>       WITH JOB_END MISSING AND
RW>       DEPARTMENT_CODE = "ELEL", "ADMN" - _____(3)
RW>       SORTED BY DEPARTMENT_CODE
RW> SET REPORT_NAME = "EMPLOYEES BY DEPARTMENT" _____(4)
RW> SET COLUMNS_PAGE = 60
RW> AT TOP OF DEPARTMENT_CODE PRINT SKIP, COL 1, _____(5)
RW>   DEPARTMENT_CODE
RW> PRINT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, _____(6)
RW>   JOB_START
RW> END_REPORT _____(7)

```

(continued on next page)

DEPARTMENT CODE	EMPLOYEE ID	FIRST NAME	LAST NAME	JOB START
ADMN	00271	Karen	Gramby	8-Jun-1980
	00228	Lisa	Harrison	2-Jul-1980
	00190	Rick	O'Sullivan	25-Feb-1982
	00300	Marjorie	Gramby	11-Feb-1982
	00330	Christine	Williams	6-Feb-1981
	00359	Jesse	Crain	28-Dec-1980
	00204	Charles	Myotte	24-Jan-1982
	00267	Roger	Saninocencio	28-Feb-1982
	00415	Kathleen	Mistretta	10-Jun-1981
	00225	Mary Lou	Jackson	3-Jan-1983
	00435	Johanna	MacDonald	17-Nov-1980
	00438	Mark	Wilkins	25-Apr-1980
	00439	Mary Lou	Smoot	26-Nov-1982
	00188	Karen	Clarke	8-Apr-1982
	00494	Barbara	Raiola-Paul	28-Dec-1979
	00488	Tom	McGrath	6-Mar-1982
	00472	Al	Delano	27-Apr-1981
	00471	James	Herbener	26-Jun-1980
ELEL	00458	Peter	Mambelli	5-Nov-1980
	00460	Adele	Meckl	5-Feb-1982
	00461	George	Boutin	17-Jun-1980
	00211	Ernest	Gutierrez	25-Jan-1982
	00443	James	Piche	5-Sep-1981
	00480	Cora	Jones	14-Jan-1982
	00238	Peter	Flynn	2-Feb-1982
	00489	Ellen	Morin	28-Oct-1980
	00428	Thomas	Augusta	10-Jan-1982
	00222	Norman	Lasch	28-Dec-1979
	00240	Bill	Johnson	18-Aug-1981
	00231	Rick	Clairmont	22-Aug-1981
	00393	Jesse	Siciliano	1-Nov-1980
	00206	Marty	Stornelli	17-Oct-1980
	00377	Lawrence	Lobdell	26-Jan-1982
	00296	Adele	Leger	7-Mar-1982
	00273	Daniel	Iacobone	31-Jan-1982
	00172	Janis	Peters	28-Oct-1980

DTR)

### 5.3 Combining Data from Many Relations in a Single Report

In the previous example, you combined data from two domains to form a report. In a relational database, data that is not directly related is usually stored in separate relations. To create a report, you typically combine many domains to generate the necessary data. For instance, to get an employee profile including salary history, job history, job title and class, and degree information, you must bring data together from five different relations.

When you combine data from different domains or relations, you must be sure to relate the records correctly. For instance, you want to join the salary history information with the job history information for the same employee. In addition, if data has a hierarchical relationship, (for example several salary history records for one job history record), you may want to show that in your reports.

There are several ways you can combine data from many relations or domains to create reports. One way uses the **CROSS** and **OVER** clauses and requires **DATATRIEVE** to combine sources when it executes the statement. This method is discussed in the section on using **CROSS** and **OVER** clauses to combine data. A second way is to use **DATATRIEVE** view domains and is discussed in the following section.

### 5.3.1 Using View Domains to Combine Data from Many Relations

A view domain lets you define the complex relationships between data and store it in the CDD. View domains tend to work more quickly than comparable queries formed using **CROSS** and **OVER** clauses. You can use the **CROSS** and **OVER** clauses in a view domain.

---

#### Note

---

You cannot base view domains directly on relations. You must first define a **DATATRIEVE** domain for each relation the view accesses.

---

View domains that use two or more domains and contain more than a single **OCCURS FOR** clause are called hierarchical views. They relate records so that for each record of one domain, many records from another domain can repeat. These repeating fields are called **list fields**. For instance, for each employee, there may be many jobs held over a period of years. In addition, for each job held there may be many salary records indicating raises while on a particular job. The view domain that represents these relationships is a view with three levels of hierarchy.

The following view domain creates a three level hierarchical relationship between the data from the employee, jobs, and salary history domains. This view domain defines the records that are used in generating an employee profile report using the **DATATRIEVE** Report Writer.



Notice that 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:

```
DTR> SHOW VIEW_PROFILE
DOMAIN VIEW_PROFILE OF EMPLOYEES, SALARY_HISTORY,
                JOB_HISTORY, JOBS, DEGREES USING
01 ONE_OF_US OCCURS FOR EMPLOYEES.----- (1)
   03 EMPLOYEE_ID FROM EMPLOYEES.
   03 FIRST_NAME FROM EMPLOYEES.
   03 LAST_NAME FROM EMPLOYEES.
   03 JOBS_HERE OCCURS FOR JOB_HISTORY CROSS JOBS----- (2)
      OVER JOB_CODE WITH
      (EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID) AND
      (JOB_END NOT MISSING) SORTED BY DECREASING JOB_START.
   05 JOB_TITLE FROM JOBS.
   05 WAGE_CLASS FROM JOBS.
   05 JOB_CODE FROM JOB_HISTORY.
   05 DEPARTMENT_CODE FROM JOB_HISTORY.
   05 JOB_START FROM JOB_HISTORY.
   05 SALARIES_FOR_JOBS OCCURS FOR SALARY_HISTORY WITH----- (3)
      (EMPLOYEE_ID = JOB_HISTORY.EMPLOYEE_ID) AND
      (SALARY_START BT JOB_START AND JOB_END) SORTED BY
      DECREASING SALARY_START.
   07 SALARY_START FROM SALARY_HISTORY.
   07 SALARY_AMOUNT FROM SALARY_HISTORY.
03 DEGREE_INFO OCCURS FOR DEGREES WITH----- (4)
   EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID SORTED BY
   DECREASING YEAR_GIVEN.
   05 DEGREE FROM DEGREES.
   05 YEAR_GIVEN FROM DEGREES.
   05 DEGREE_FIELD FROM DEGREES.
```

Examine the view domain and then the fields that it contains. Note the following:

1. The field ONE\_OF\_US is the top hierarchical field; all others are subordinate.
2. JOBS\_HERE is a list field. It uses the CROSS and OVER clauses to combine records from the JOB\_HISTORY and JOBS domains. The fields subordinate to JOBS\_HERE (JOB\_TITLE, WAGE\_CLASS, JOB\_CODE, DEPARTMENT\_CODE, JOB\_START, SALARIES\_FOR\_JOBS) repeat. They repeat at least once for every past job in the JOB\_HISTORY domain for each employee. So, for instance, if an employee with id of 00168 has had three job starts, the field JOB\_START repeats three times, once for each job.

The syntax, JOB\_END NOT MISSING, ensures that your view includes only those jobs that are not current.



3. SALARIES\_FOR\_JOBS is a list field. It is subordinate to the JOBS\_HERE field and contains all the salary starts and salary amounts that occur between the beginning and ending date of each job from the JOB\_HISTORY domain. For instance, for each job start an employee record may have three salary changes.
4. Degrees is a list field. It is subordinate to ONE\_OF\_US.

The fields that participate in the view domain and the hierarchy it creates can be seen by entering the SHOW FIELDS command after you ready the domain:

```
DTR> READY VIEW_PROFILE
DTR> SHOW FIELDS
VIEW_PROFILE
  ONE_OF_US
    EMPLOYEE_ID <Character string>
    FIRST_NAME <Character string>
    LAST_NAME <Character string>
    JOBS_HERE <List>
      JOB_TITLE <Character string>
      WAGE_CLASS <Character string>
      JOB_CODE <Character string>
      DEPARTMENT_CODE <Character string>
      JOB_START <Date>
      SALARIES_FOR_JOBS <List>
        SALARY_START <Date>
        SALARY_AMOUNT <Number>
    DEGREE_INFO <List>
      DEGREE <Character string>
      YEAR_GIVEN <Number>
      DEGREE_FIELD <Character string>
```

After you create a view domain that relates the relational domains, you can write a report that uses the definition. The following example illustrates an employee profile report based on the preceding view domain. The report presents historical data on the employee you select, including salary history, job history, and academic background. The data in this report is drawn from five different relations or domains.

The report uses inner print lists to access the list fields you established in your view domain. Use the techniques for accessing list fields described in the *VAX DATATRIEVE User's Guide*.

Follow these steps:

1. Ready the view domain. This readies the five participating domains.
2. Declare a variable to contain the employee ID for which you will prompt the user.
3. Call the report writer and form an RSE for the selected employee.

4. Give your report a name.
5. At the top of the EMPLOYEE\_ID field, print the employee name and ID.
6. Print all the column headings.
7. Form your first inner print list to access the department code, job start, and job title information from the JOBS and JOB\_HISTORY domains. Use the syntax for an inner print list, PRINT ALL OF RSE. Note that the JOBS\_HERE field from the view domain is the RSE that provides context in an inner print list. Use the FN\$STR\_EXTRACT function to shorten the data in the JOB\_TITLE field to 13 characters so that it will fit in the space you provide for that column of the report.
8. Now form a second inner print list to access the salary information.
9. Using the AT BOTTOM statement, print the data from the list field DEGREE\_INFO with an inner print list.
10. End your report with an END\_REPORT statement.

```

DTR> READY VIEW_PROFILE_____ (1)
DTR> DECLARE EMP_ID PIC X(5)._____ (2)
DTR> EMP_ID = *."Employee ID for Report"
DTR> REPORT VIEW_PROFILE WITH EMPLOYEE_ID = EMP_ID_____ (3)
RW> SET REPORT_NAME = "Employee Profile"_____ (4)
RW> AT TOP OF EMPLOYEE_ID PRINT COL 1,_____ (5)
RW>     "ID:", COL 15, EMPLOYEE_ID(-) USING X(10), COL 1,
RW>     "Name:", COL 15, FIRST_NAME LAST_NAME, SKIP 2,
RW>     COL 1, "DEPT CODE", COL 14, "JOB START",_____ (6)
RW>     COL 26, "JOB TITLE", COL 43,
RW>     "SALARY START", COL 62, "SALARY AMOUNT", SKIP 2
RW> PRINT COL 1,
RW>     ALL DEPARTMENT_CODE (-), COL 13,_____ (7)
RW>     JOB_START (-), COL 26,
RW>     FN$STR_EXTRACT(JOB_TITLE, 1,13) (-), COL 43,
RW>     ALL SALARY_START (-), SALARY_AMOUNT (-) OF_____ (8)
RW>     SALARIES_FOR_JOBS OF JOBS_HERE
RW> AT BOTTOM OF REPORT PRINT SKIP 2, COL 1, "DEGREE",
RW>     COL 24, "DEGREE FIELD", COL 68, "YEAR AWARDED",
RW>     SKIP, COL 1,
RW>     ALL DEGREE (-), COL 24, DEGREE_FIELD (-), COL 68,_____ (9)
RW>     YEAR_GIVEN (-) OF DEGREE_INFO
RW> END_REPORT_____ (10)

```

DTR>  
Enter Employee ID for Report: 00168

Employee Profile

9-Jul-1985  
Page 1

ID: 00168  
Name: Norman Nash

DEPT CODE	JOB START	JOB TITLE	SALARY START	SALARY AMOUNT
SUWE	23-Feb-1979	Programmer	10-Oct-1981	\$27,126.00
			15-Oct-1980	\$25,057.00
			21-Oct-1979	\$23,919.00
			23-Feb-1979	\$23,605.00
ENG	30-Oct-1977	Programmer	26-Aug-1978	\$21,520.00
			30-Oct-1977	\$20,883.00
ELMC	1-Jul-1975	Associate Pro	21-Apr-1977	\$15,977.00
			24-Aug-1976	\$15,851.00
			1-Jul-1975	\$15,179.00

DEGREE	DEGREE FIELD	YEAR AWARDED
PhD	Applied Math	1983
MA	Applied Math	1978
BA	Arts	1973
AA	Arts	1969

DTR>

### 5.3.2 Using the CROSS and OVER Clauses to Combine Data for a Report

You can also use the CROSS and OVER clause to join domains. The CROSS clause joins the two record streams, and the OVER clause results in records being matched on a common field. For example, examine the following RSE:

```
DTR> FOR EMPLOYEES CROSS SALARY_HISTORY -  
CON> OVER EMPLOYEE_ID CROSS JOB_HISTORY -  
CON> OVER EMPLOYEE_ID WITH EMPLOYEE_ID = "00168" AND  
CON> (SALARY_START BT JOB_START AND JOB_END)  
CON>
```

When you enter this statement, DATATRIEVE combines the selected employee record, 00168, from the EMPLOYEES domain with the SALARY\_HISTORY records matched on the same employee ID. Note that, because you specify the Boolean SALARY\_START BT JOB\_START AND JOB\_END, DATATRIEVE does not combine each job record with every salary record. Instead, it combines each job record with only those salary records that fall between the related job start and job end dates.



```
CON> PRINT EMPLOYEE_ID, JOB_START, SALARY_START, SALARY_AMOUNT
```

EMPLOYEE ID	JOB START	SALARY START	SALARY AMOUNT
00168	1-Jul-1975	1-Jul-1975	\$15,179.00
00168	30-Oct-1977	26-Aug-1978	\$21,520.00
00168	23-Feb-1979	23-Feb-1979	\$23,605.00
00168	23-Feb-1979	21-Oct-1979	\$23,919.00
00168	23-Feb-1979	15-Oct-1980	\$25,057.00
00168	23-Feb-1979	10-Oct-1981	\$27,126.00
00168	1-Jul-1975	24-Aug-1976	\$15,851.00
00168	1-Jul-1975	21-Apr-1977	\$15,977.00
00168	30-Oct-1977	30-Oct-1977	\$20,883.00

```
DTR>
```

Note that when you print out such a record stream, the data from the `EMPLOYEE_ID` and `JOB_START` fields repeat for as many times as there are unique salary history records for that employee. Thus, you can get repetitious data in your record stream. This repetition of information is due to the fact that the `CROSS` and `OVER` clauses result in a flat record display.

In order to format record streams in a more useful way, you may want to use collections and inner print lists to present a more hierarchical output. For instance, you may want the employee ID and name to repeat only once for each employee. The following report shows how, by using collections and inner print lists, you can do this. The inner print lists in this example use collections as record sources within the `REPORT` statement.

Notice that with `DATATRIEVE`, typically, you need not use context variables to qualify field names. `DATATRIEVE` knows, for instance, that in the query `EMPLOYEES CROSS SALARY_HISTORY OVER EMPLOYEE_ID`, the employee ID is a common field in both the preceding domains.

Follow these steps:

1. Ready the necessary domains.
2. Declare a variable to contain the truncated job title data. The `JOB_TITLE` field is 30 characters in the relational database but the report only has room for a shorter field. You can use this technique to shorten the output from that field for your report.
3. Declare a variable and prompt for an employee ID.
4. Form a collection with the employee record you select at the prompt.

5. Form a collection by joining the selected employee record with matching data from the `JOB_HISTORY` and `JOBS` domains. The domains are matched based on the common fields, `EMPLOYEE_ID` and `JOB_CODE`. This collection contains historical job data for the employee you selected, sorted by the job start date.
6. Now you can cross the current collection with the domain containing salary history information. The Boolean expression `WITH SALARY_START BETWEEN JOB_START AND JOB_END` lets you identify a hierarchical relationship between the job data and the salary data.
7. Now form a new collection, `JOB_SALARY`, to contain all the current data. Use this collection as the record source in the `REPORT` statement.
8. Find a collection, `DEGREE_INFO`, to contain all the academic data about the employee you selected. You can then use this collection to provide context for the degree information using an inner print list:
 

```
ALL...DEGREE, DEGREE_FIELD, YEAR_GIVEN OF DEGREES
```
9. Declare a variable to contain the combined data from the `DEPARTMENT_CODE` and `JOB_START` fields. You use this variable as a control break in your report.
10. Call the Report Writer and report the `JOB_SALARY` collection.
11. Use the `AT TOP` statement to print the employee name, the employee ID, and address. Use concatenation operators to control the spacing between fields.
12. Print the headers for your report.
13. Use a second `AT TOP` statement to print the department code, the job start date, and the truncated job title.
14. Use the `PRINT` statement (you can use only one in a `DATATRIEVE REPORT` statement) to print salary data from the current collection.
15. Use the `At BOTTOM` statement to print out the related college information from the `DEGREES` domain using an inner print list.
16. End the report specification with the `END_REPORT` statement.



```

DTR>
DTR> READY JOB_HISTORY, SALARY_HISTORY, JOBS, _____(1)
DTR> DEGREES, EMPLOYEES
DTR> DECLARE SHORT_JOB_TITLE COMPUTED BY _____(2)
DTR> JOB_TITLE""EDIT_STRING X(13).
DTR> DECLARE EMP_ID PIC X(5). _____(3)
DTR> EMP_ID = *."Employee ID for Report"
DTR> FIND EMPLOYEES WITH EMPLOYEE_ID = EMP_ID _____(4)
DTR> FIND CURRENT CROSS JOB_HISTORY OVER _____(5)
CON> EMPLOYEE_ID CROSS
CON> JOBS OVER JOB_CODE WITH
CON> JOB_END NOT MISSING AND
CON> EMPLOYEE_ID = EMP_ID -
CON> SORTED BY DECREASING JOB_START
DTR> FIND CURRENT CROSS SALARY_HISTORY OVER _____(6)
CON> EMPLOYEE_ID WITH
CON> EMPLOYEE_ID EQ EMP_ID AND
CON> JOB_END NOT MISSING AND
CON> (SALARY_START BT JOB_START AND JOB_END) -
CON> SORTED BY DECREASING JOB_START, SALARY_START
DTR> FIND JOB_SALARY IN CURRENT _____(7)
DTR> FIND DEGREE_INFO IN DEGREES WITH _____(8)
CON> EMPLOYEE_ID =
CON> EMP_ID SORTED BY DECREASING YEAR_GIVEN
DTR> DECLARE BREAK_F COMPUTED BY _____(9)
CON> DEPARTMENT_CODEJOB_START.
DTR> PRINT "Doing Report..."
DTR> REPORT JOB_SALARY _____(10)
RW> SET REPORT_NAME = "Employee Profile"
RW> AT TOP OF EMPLOYEE_ID PRINT COL 1, _____(11)
RW> "Id:", COL 15, EMPLOYEE_ID(-) USING X(10),
RW> COL 1,"Name:", COL 15, FIRST_NAMELAST_NAME,
RW> SKIP 2, COL 1, "DEPT CODE", COL 14, _____(12)
RW> "JOB START", COL 26, "JOB TITLE", COL 43,
RW> "SALARY START", COL 62, "SALARY AMOUNT",
RW> SKIP 2
RW> AT TOP OF BREAK_F PRINT COL 1, _____(13)
RW> DEPARTMENT_CODE (-), COL 13,
RW> JOB_START (-), COL 26,
RW> SHORT_JOB_TITLE (-)
RW> PRINT COL 43, _____(14)
RW> SALARY_START (-), SALARY_AMOUNT (-)
RW> AT BOTTOM OF REPORT PRINT SKIP 2, COL 1, _____(15)
RW> "DEGREE", COL 24, "DEGREE FIELD",
RW> COL 68, "YEAR AWARDED", SKIP,
RW> ALL COL 1, DEGREE (-), COL 24,
RW> DEGREE_FIELD (-), COL 68,
RW> YEAR_GIVEN (-) OF DEGREES WITH
RW> EMPLOYEE_ID = EMP_ID
RW> END_REPORT _____(16)

```

Enter Employee ID for Report: 00168  
Doing report...

Employee Profile

9-Jul-1985  
Page 1

ID: 00168  
Name: Norman Nash

DEPT CODE	JOB START	JOB TITLE	SALARY START	SALARY AMOUNT
SUWE	23-Feb-1979	Programmer	10-Oct-1981	\$27,126.00
			15-Oct-1980	\$25,057.00
			21-Oct-1979	\$23,919.00
			23-Feb-1979	\$23,605.00
ENG	30-Oct-1977	Programmer	26-Aug-1978	\$21,520.00
			30-Oct-1977	\$20,883.00
ELMC	1-Jul-1975	Associate Pro	21-Apr-1977	\$15,977.00
			24-Aug-1976	\$15,851.00
			1-Jul-1975	\$15,179.00

DEGREE	DEGREE FIELD	YEAR AWARDED
PhD	Applied Math	1983
MA	Applied Math	1978
BA	Arts	1973
AA	Arts	1969

DTR>

Notice in this report that you create the same hierarchical relationships between the employee, salary history, and job history data that you did with the view domain in the previous section.

## 5.4 Developing a Procedure for a Report

Report statements are usually stored as procedures. A report that is defined as a procedure lets you generate a required report whenever you want, without having to type in a new report statement. Defining procedures for reports with relational databases becomes especially important, as the RSEs can become very complex. The following report takes data from six domains in the PERSONNEL database to create a report. Follow these steps:

1. Define a procedure using the DEFINE PROCEDURE statement.
2. Ready all the domains that contain the data needed in your report.
3. Declare a global variable to contain the value Y. Your procedure prompts the user to indicate how many departments the report should contain.

4. Invoke the Report Writer and create an RSE containing records from all the domains you readied.

Note that you use the CROSS and OVER clauses with each domain to indicate an additional domain and the field on which you want the records matched.

5. Use the WITH Boolean expression only after you cross all the required domains. The WITH clause lets you restrict the record stream to just those records desired.

The prompting value expression lets the user specify the department or departments contained in the report.

Note that, by specifying JOB\_END MISSING and SALARY\_END MISSING, you can select just the latest records of those employees currently working.

6. Assign the output of the Report Writer to a file in the desired directory.
7. Give the report a name.
8. Prompt the user to enter a request for information on another employee.

```

DTR> DEFINE PROCEDURE ACTIVE_EMPLOYEES_REPORT _____(1)
DFN> READY EMPLOYEES, JOB_HISTORY, SALARY_HISTORY, _____(2)
DFN>     JOBS, DEPARTMENTS, WORK_STATUS
DFN> DECLARE MORE_DEPT PIC X. _____(3)
DFN> MORE_DEPT = "Y"
DFN> WHILE MORE_DEPT= "Y"
DFN> BEGIN
DFN> REPORT EMPLOYEES - _____(4)
CON>     CROSS SALARY_HISTORY OVER EMPLOYEE_ID -
CON>     CROSS JOB_HISTORY OVER EMPLOYEE_ID -
CON>     CROSS JOBS OVER JOB_CODE -
CON>     CROSS DEPARTMENTS OVER DEPARTMENT_CODE -
CON>     CROSS WORK_STATUS OVER STATUS_CODE -
CON>     WITH FN$UPCASE(DEPARTMENT_CODE) =
DFN>     *."Department Code" AND _____(5)
DFN>     JOB_END MISSING AND
DFN>     SALARY_END MISSING AND
DFN>     WORK_STATUS.STATUS_CODE EQ "1" AND
DFN>     SALARY_AMOUNT > 15000 -
CON>     SORTED BY DEPARTMENT_CODE, LAST_NAME, SALARY_AMOUNT -
CON>     ON DB2:[DEPT]DEPT.LIS _____(6)
DFN> SET REPORT_NAME = "Current Employees by Department" _____(7)
DFN> AT TOP OF DEPARTMENT_CODE PRINT SKIP, COL 1,
DFN>     FN$UPCASE(DEPARTMENT_NAME) (-), SKIP
DFN> AT TOP OF EMPLOYEE_ID PRINT SKIP, COL 1,
DFN>     FIRST_NAME LAST_NAME (-)
DFN> PRINT COL 1, JOB_TITLE, JOB_START,
DFN>     WAGE_CLASS, SALARY_AMOUNT, STATUS_CODE
DFN> AT BOTTOM OF DEPARTMENT_CODE PRINT
DFN> "*****"
DFN> END_REPORT

```



```

DFN> MORE_DEPT =
DFN> *."Y to see more Departments , N to end report" -----(8)
DFN> MORE_DEPT = FN$UPCASE(MORE_DEPT)
DFN> END
DFN> END_PROCEDURE
DTR> :ACTIVE_EMPLOYEE_REPORT
Enter Department Code: ADMN
  Enter Y to see more Departments, N to end report: N

DTR> exit

$ TYPE Dept.lis

```

Current Employees by Department 3-May-1984  
Page 1

JOB TITLE	JOB START	WAGE CLASS	SALARY AMOUNT	STATUS CODE
CORPORATE ADMINISTRATION				
Karen Clarke Electrical Engineer	8-Apr-1982	4	\$21,093.00	1
Al Delano Dept. Supervisor	27-Apr-1981	4	\$39,531.00	1
Marjorie Gramby Electrical Engineer	11-Feb-1982	4	\$23,856.00	1
Karen Gramby Vice President	8-Jun-1980	4	\$75,113.00	1
Lisa Harrison Vice President	2-Jul-1980	4	\$77,307.00	1
James Herbener Vice President	26-Jun-1980	4	\$83,905.00	1
Johanna MacDonald Vice President	17-Nov-1980	4	\$84,147.00	1
*****				









## Reference Section

Replace this page with heavy page entitled **Reference Section** located at the end of this package.





## DATATRIEVE Report Writer Reference Section 6

This chapter describes the DATATRIEVE Report Writer statements. These are the DATATRIEVE statements that you use *only* within the Report Writer. You use these statements in addition to the statements and commands described in the *VAX DATATRIEVE Reference Manual*.

Each section of the chapter divides its presentation of a statement into the following categories:

- **Format**

Includes the spelling and placement of keywords and the placement of required and optional syntax elements. As a rule, statement names and other keywords cannot be abbreviated. The sequence of statement elements is also critical. You must follow the sequence shown in the format. If you omit an optional element, leave its relative position in the statement empty and proceed to the remaining elements in a left-to-right order.

- **Arguments**

Explain each element of the syntax in greater detail.

- **Restrictions**

Tell you what requirements and limits there are on the use and action of a statement.

- **Results**

Tell you what action DATATRIEVE takes when you use the statement and its various options.

- **Usage Notes**

Present some common uses of the statement and its elements and indicate what other statements you can use in conjunction with the statement.

Table 6-1 identifies the Report Writer statements. The (RW) indicates that the PRINT and SET statements are different from the DATATRIEVE PRINT statement and SET command used at the DTR> prompt.

**Table 6-1: Summary of Report Writer Statements**

Statement	Function
AT BOTTOM	Displays summary lines at the bottom of a report, page, or control group.
AT TOP	Displays header lines at the top of a report, page, or control group.
END_REPORT	Indicates the end of a report specification.
PRINT (RW)	Displays value expressions for each detail line of a report.
REPORT	Invokes the Report Writer and specifies the data you want to report and the output device.
SET (RW)	Sets the page format for a report.

## 6.1 AT BOTTOM Statement

Displays summary lines at the bottom of reports, report pages, and control groups. Calculates the summary information based on the records of the page, group, or report that you specify in the statement.

The AT BOTTOM statement accepts the same types of print objects as a PRINT statement.

### Format

```

AT BOTTOM OF { REPORT
              PAGE
              field-name } PRINT summary-element [...]
```

### Arguments

field-name

Is a field from the record definition for the report's record stream or a variable.

summary-element

Specifies the value, position, and format of the fields. These elements are summarized in Table 6-2.

**Table 6-2: AT BOTTOM Statement Summary Elements**

Summary Element	Function	Usage Notes
AVERAGE value expression	Displays the average value of the value expressions.	Calculated for the detail lines of the report, page, or group specified.
COL n	Specifies where the output of the next element begins.	Same usage as in the PRINT statement.

(continued on next page)



## AT BOTTOM

**Table 6-2: AT BOTTOM Statement Summary Elements (Cont.)**

Summary Element	Function	Usage Notes
COUNT	Displays the number of detail lines.	Calculated for the detail lines of the report, page, or group specified.
field-name [modifier]	In AT BOTTOM OF field-name, prints the common value for a control group.	Same usage as in the PRINT statement.
MAX value expression	Displays the maximum value of the value expression.	Calculated for the detail lines of the report, page, or group specified.
MIN value expression	Displays the minimum value of the value expression.	Calculated for the detail lines of the report, page, or group specified.
NEW_PAGE	Causes the Report Writer to start a new page before the output of the next summary element.	Same usage as in the PRINT statement. Cannot be used in AT BOTTOM OF PAGE.
NEW_SECTION	Starts a new page and a new sequence of page numbers, beginning with Page 1.	Cannot be used in AT BOTTOM OF PAGE.
RUNNING COUNT	In AT BOTTOM OF field-name, prints the number of control breaks for that field to that point in the report.	In AT BOTTOM OF PAGE, prints the number of pages to that point in the report.
SKIP [n]	Prints the next element n lines down.	Same usage as in the PRINT statement.
SPACE [n]	Inserts spaces between the output of the preceding and following elements.	Same usage as in the PRINT statement.
STD_DEV value expression	Displays the standard deviation of values for the value expression.	Calculated for the detail lines of the report, page, or group specified.

(continued on next page)

Table 6-2: AT BOTTOM Statement Summary Elements (Cont.)

Summary Element	Function	Usage Notes
TAB [n]	Inserts the space of one or more tabs before the output of the next element.	Same usage as in the PRINT statement. DATATRIVEE assumes that tabs are set every 8 spaces and inserts enough spaces to begin the next field in the appropriate column.
TOTAL value expression	Displays the total of values for the expression.	Calculated for the detail lines of the report, page, or group specified.
value expression	Displays the value in a summary line.	Same usage as in the PRINT statement.

### Restrictions

- You cannot specify the summary elements `NEW_PAGE` and `NEW_SECTION` in an `AT BOTTOM OF PAGE PRINT` statement.
- For relational sources, `DATATRIVEE` uses the relation's name as the top-level group field. This top-level field cannot be used in `AT TOP` or `AT BOTTOM` statements. For example, with the `EMPLOYEES` relational domain, you cannot use this statement:  
  

```
AT BOTTOM OF EMPLOYEES PRINT EMPLOYEE_ID
```
- You cannot use `OCCURS` fields from views in `AT TOP OF` or `AT BOTTOM OF` statements. Using view domain fields defined with `OCCURS` clauses in Report Writer `AT` statements causes a `NODATA` error message.

### Results

- The `AT BOTTOM OF` field-name statement establishes a pattern of control breaks for the entire report, dividing the report into groups of records with a common value for the field. `DATATRIVEE` displays the summary line at the bottom of the control group for which the field name is the sort key. Statistical expressions are computed for the records for the detail lines in that control group.

## AT BOTTOM

When you specify `AT BOTTOM OF field-name PRINT field-name`, the Report Writer prints the value in the specified field of the last detail line in the control group.

- If you use `AT BOTTOM OF field-name`, `DATATRIEVE` checks the sort order of the collection or record stream you want to report. You can establish a sort order with the `FIND`, `SORT`, or `REPORT` statements.
- If `AT BOTTOM OF REPORT` is included in a report specification, `DATATRIEVE` displays the header line or summary line below the detail lines on the last page of the report. Statistical expressions are computed for the records for all of the detail lines in the report.
- If an `AT BOTTOM OF PAGE` statement is included in a report specification, `DATATRIEVE` displays the header line or summary line at the bottom of each page of the report. Statistical expressions are computed for the records for the detail lines on that page.

### Usage Note

If you use `AT BOTTOM OF field-name` without having sorted the records, 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.

You may want to follow this approach in the following situation. For example, the records may already be sorted according to an indexed key field. If there is only one level of control break, the Report Writer divides the control groups at the appropriate places. However, if you want multiple levels of control groups, you must sort the records in advance with a `FIND`, `SORT`, or `REPORT` statement.

### Examples

See the chapter on mastering report writing techniques for examples of the `AT BOTTOM` statement.

## 6.2 AT TOP Statement

Displays header elements at the top of reports, report pages, and control groups. It can also be used for summary lines. However, an AT TOP statement summarizes information for the entire group of records in the current collection, not the records of the page, group, or report that you specify in the statement.

The AT TOP statement accepts the same types of print objects as a PRINT statement.

### Format

AT TOP OF $\left\{ \begin{array}{l} \text{REPORT} \\ \text{PAGE} \\ \text{field-name} \end{array} \right\}$ PRINT $\left\{ \begin{array}{l} \text{header-element} \\ \text{summary-element} \end{array} \right\}$ [...]
---

### Arguments

field-name

Is a field from the record definition for the report's record stream or a variable.

header-element

summary-element

Specifies the value, position, and format of the fields. These elements are summarized in Table 6-3.

# AT TOP

**Table 6-3: AT TOP Statement Header and Summary Elements**

Header or Summary Element	Function	Usage Notes
AVERAGE value-expression	Displays the average value of the value expressions in the current collection.	Does not indicate the average for the detail lines of the report, page, or control group.
COL n	Specifies where the output of the next header or summary element begins.	Same usage as in the PRINT statement.
COLUMN_HEADER	Displays the column headers.	Overrides the suppression of column headers for AT TOP OF REPORT or PAGE.
COUNT  field-name [modifier]	Displays the number of records in the current collection.  In AT TOP OF field-name, prints the common field value at the top of each control group.	Does not indicate the number of detail lines of control group specified.  Same usage as in the PRINT statement.
MAX value expression	Displays the maximum value of all value expressions in the current collection.	Does not indicate the maximum value for the detail lines of the report, page, or control group specified.
MIN value expression	Displays the minimum value of all value expressions in the current collection.	Does not indicate the minimum value for the detail lines of the report, page, or control group specified.
NEW_PAGE	Starts a new page for the report.	Same usage as in the PRINT statement. Cannot be used in AT TOP OF PAGE.

(continued on next page)



Table 6-3: AT TOP Statement Header and Summary Elements (Cont.)

Header or Summary Element	Function	Usage Notes
NEW_SECTION	Starts a new page and a new section numbered as Page 1.	Produces a new section for each control group in AT TOP OF field-name. Cannot be used in AT TOP OF PAGE.
REPORT_HEADER	Displays the report header, including the report name, date, and page number.	Overrides the suppression of a report header in AT TOP OF REPORT or PAGE.
RUNNING COUNT	In AT TOP OF field-name, prints the number of control breaks for that field to that point in the report.	In AT TOP OF PAGE, prints the number of pages to that point in the report.
SKIP [n]	Prints the next header or summary element n lines from the current line.	Same usage as in the PRINT statement.
SPACE [n]	Inserts spaces between the output of the preceding and following elements.	Same usage as in the PRINT statement.
STD_DEV value expression	Displays the standard deviation for the value expressions in the current collection.	Does not indicate the standard deviation for the detail lines of the report, page, or control group specified.
TAB [n]	Inserts the space of one or more tabs before the output of the next element.	Same usage as in the PRINT statement. DATATRIEVE assumes that tabs are set every 8 spaces and inserts enough spaces to begin the next field in the appropriate column.

(continued on next page)

**Table 6-3: AT TOP Statement Header and Summary Elements (Cont.)**

Header or Summary Element	Function	Usage Notes
TOTAL value expression	Displays the total for the value expressions in the current collection.	Does not indicate the total for the detail lines of the report, page, or control group.
value expression	Displays the value in a header or summary line.	Same usage as in the PRINT statement.

## Restrictions

- You cannot specify a statistical expression in an AT TOP statement unless you have already formed a current collection. VAX DATATRIEVE evaluates the statistical expression based on the records in the current collection.
- You cannot specify the summary elements NEW\_PAGE and NEW\_SECTION in an AT TOP OF PAGE PRINT statement.
- For relational sources, DATATRIEVE uses the relation's name as the top-level group field. This top-level field cannot be used in AT TOP or AT BOTTOM statements. For example, with the EMPLOYEES relational domain, you cannot use this statement:

```
AT BOTTOM OF EMPLOYEES PRINT EMPLOYEE_ID
```

- You cannot use OCCURS fields from views in AT TOP OF or AT BOTTOM OF statements. Using view domain fields defined with OCCURS clauses in Report Writer AT statements causes a NODATA error message.
- Under certain conditions, specifying an AT TOP OF clause may have an effect on the subsequent column positioning of the field if it is also specified in the PRINT detail line. This occurs under the following combination of circumstances:
  - You specify the field in an AT TOP OF statement.
  - You specify a column number for the start of the field value in the print detail line.

- You do not specify a header for the column in the print detail line.

To print the field value in the column you specify, you must explicitly suppress the header for the individual field using a hyphen in parentheses (-).

## Results

- If an AT TOP OF REPORT is included in a report specification, DATATRIEVE displays the header line or summary line above the detail lines on the first page of the report and suppresses the report header on the first page of the report. The report header is displayed on the following pages of the report, and the page numbers begin with Page 1 on the second physical page of the report. To specify a title page for your report, end the print list with NEW\_PAGE or NEW\_SECTION.
- If an AT TOP OF PAGE is included in a report specification, DATATRIEVE displays the header line or summary line at the top of each page of the report and replaces the report header on every page.
- The AT TOP OF field-name statement establishes a pattern of control breaks for the entire report, dividing the report into groups of records with a common value for the field. DATATRIEVE displays the header line or summary line at the top of the control group for which the field name is the sort key.

## Usage Notes

- The header and summary elements may contain modifiers to specify edit strings or to suppress headers.
- DATATRIEVE checks the sort order of the collection or record stream you want to report. You can establish a sort order with the FIND, SORT, or REPORT statements.

If you use AT TOP OF field-name without having sorted the records, 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.

## Examples

See the chapter on mastering report writing techniques for examples of the AT TOP statement.

# END\_\_REPORT

## 6.3 END\_\_REPORT Statement

Ends the report specification.

### Format

END__REPORT
-------------

### Restriction

The END\_\_REPORT statement must be the last statement in the report specification.

### Results

After you enter the END\_\_REPORT statement, the Report Writer takes one of three courses of action:

- Prompts you for the values you specified with a \*.prompt in the record specification and then produces the report.
- Sends you a message indicating a syntax error in the report specification. When you see the DTR> prompt, type EDIT and press RETURN so that you can make the needed changes. When you leave the editor with the EXIT command, DATATRIEVE executes the new report specification. You can also edit a report specification by enclosing it in a procedure by using the DEFINE PROCEDURE command.
- Produces the report and sends it to the device or file you have specified in the REPORT command.

## 6.4 PRINT Statement (RW)

Specifies the following characteristics of the detail lines in a report:

- The content of the detail lines—field values or other desired values and text strings
- The format of fields in the detail lines—order, column position, and edit string format of print objects
- Column headers for the print objects in the detail line

You can include only one PRINT statement in a report specification. If your report specification contains an AT statement, then it does not have to contain a PRINT statement.

### Format

```
PRINT print-list-element [...]
```

### Argument

print-list-element

Specifies the values, position, and format of the print objects in the detail line.

Table 6-4 indicates the parameters of the report controlled by various print list elements.



# PRINT

**Table 6-4: Report Parameters Controlled by Print List Elements**

Parameter	Print List Element	Usage Notes
Content of detail line	Field-name [modifier]	Can include elementary, group, list, REDEFINES or COMPUTED BY fields; to print all fields, specify the top-level field name.
	Related value-expression [modifier]	Derived from field values using arithmetic operators or RUNNING TOTAL.
	Other value-expression [modifier]	Can include literals, variables, or RUNNING COUNT.
Format of detail line	COL n	Specifies where the output of the next print list element begins.
	TAB [n]	Inserts the space of n tab characters before the output of the next print list element.
	SKIP [n]	Begins printing the next print list element n lines from the current line.
	SPACE [n]	Inserts spaces between the output of the preceding and following print list elements.
Beginning of new page	NEW_PAGE	Causes the Report Writer to start a new report page.

Table 6-5 indicates the parameters of the report controlled by modifiers of print list elements.

**Table 6-5: Report Parameters Affected by Print List Modifiers**

Parameter	Print List Modifier	Usage Notes
Column headers for print items	("header-segment"[/...])  (-)	Specifies one or two line headers for the preceding field or value expression, overriding the field name or query header from the record definition.  Suppresses the header indicated for the field in the record definition.
Format of the detail line item	USING edit-string	Imposes the characteristics of the edit string on the preceding field name or value expression.

**Usage Notes**

- 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.
- 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 may be lost.
- When specifying the detail line, you are not restricted to the field order of the record definition. In the PRINT statement, you can list the fields (and their column headers and edit strings) in any order you choose.
- When the data you are reporting includes a list, use an inner print list element (ALL [print-list] of rse) in the PRINT statement to specify the value, position, and format for fields in the list. Each item of the list takes at least one physical line of printing.

## PRINT

- For relational sources, DATATRIEVE uses the relation's name as the top-level group field. This top-level field cannot be used as a print list element. For example, the following statements using the COLLEGES relational domain produce an error message:

```
DTR> REPORT COLLEGES
RW> PRINT COLLEGES
RW> END_REPORT
"COLLEGES" is undefined or used out of context.
```

- To suppress or specify a column header with a print list element that includes a CDD object, you must enclose the entire value expression in parentheses. Examples of such print list elements include the following:

- A field from a domain table or dictionary table using a VIA clause. An example of such a print list element would be:

```
      .
      .
      .
RW> PRINT (RIG VIA RIG_TABLE) (-)
```

- A value that is determined by a procedure, for example:

```
      .
      .
      .
RW> PRINT (:DURATION) ("DURATION")
```

The use of parentheses around the CDD object expression is necessary in these cases to avoid confusing the column header specification with a password for the CDD object.

For more information on column headers, see the chapter on designing a report with Report Writer.

- 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 field definition contains an edit string, that edit string determines the format for the field.
  - If the field definition has no edit string, the PICTURE clause determines the format for the field.

- If the field definition has neither an edit string nor a PICTURE clause, the Report Writer invents a picture clause to accommodate the data in the field.

To gain full control over the formats of the fields of your detail lines, explicitly define edit strings with a USING clause.

- If the print list includes a variable or a field name and a statistical function (TOTAL, RUNNING TOTAL, AVERAGE, MAX, MIN, or STD\_DEV) based on that variable or field, the Report Writer puts both values in the same column, even if you specify a separate column header with the statistical function:

```
DTR> FIND FIRST 3 PAYABLES
DTR> REPORT CURRENT
RW> PRINT WHSLE_PRICE,
RW> TOTAL WHSLE_PRICE ("GRAND TOTAL"/"OWED") USING $$$,$$$
RW> END_REPORT
```

13-Aug-1987  
Page 1

```
WHSLE
PRICE
$40,000
$82,000
.
.
.
```

You can print the statistical function in a separate column by specifying a column assignment:

```
DTR> FIND FIRST 3 PAYABLES
DTR> REPORT CURRENT
RW> PRINT WHSLE_PRICE,
RW> COL 15, TOTAL WHSLE_PRICE ("GRAND TOTAL"/"OWED") USING $$$,$$$
RW> END_REPORT
```

13-Aug-1987  
Page 1

```
WHSLE      GRAND TOTAL
PRICE      OWED
$40,000    $82,000
.
.
.
```

## PRINT

Note that you may also wish to specify an explicit column for the statistical function in cases where the calculated size of the statistical function value is significantly larger than the size of the field or variable it is based on. This can happen, for example, if the definition of the field or variable includes a **COMPUTED BY** clause.



## 6.5 REPORT Statement

Invokes the Report Writer and is the first entry in a report specification. In the REPORT statement you can specify:

- The data you want to report
- The output device for the report

The other statements in the report specification are AT BOTTOM, AT TOP, END\_REPORT, PRINT, and SET. These statements are discussed in separate sections of this chapter.

### Format

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

### Arguments

rse

Specifies the data for your report. To create a record stream for your report, enter the appropriate RSE in the REPORT statement. You can make reports using data from:

- Readied domains
- Collections
- Lists

When you omit the RSE, the Report Writer uses the data in your current collection for the report. If there is no current collection, DATATRIEVE displays the following error message:

```
A current collection has not been established.
```

file-spec

Is the file to which you want to write the report. A complete file specification has the following format:

```
node-spec::device:[directory]file-name.type;version
```

## REPORT

The minimum file specification consists of a period (.); the specification of such a file stored in your default VMS directory ends with “.;n”, where n is the version number and both the file name and the type are null strings.

If you omit a field in the file specification, DATATRIEVE uses the defaults as listed in Table 6-6.

**Table 6-6: Output File Specification Defaults**

Field	Default
node-spec::	Your local node
device:	Your default device
[directory]	Your default directory
file-name	Null string
.type	.LIS
;version	Highest version number

For more information on how to specify files, see the *VAX DATATRIEVE Reference Manual*.

When you omit the ON clause in a REPORT statement, DATATRIEVE displays the report on your terminal.

\*.prompt

Is a prompting value expression that allows you to specify a file specification when DATATRIEVE processes the report specification.

### Usage Notes

- You can report on data only in domains readied for READ, WRITE, or MODIFY access. Because you cannot establish collections or record streams in domains readied for EXTEND access, you cannot report on data in domains readied for EXTEND access.
- The data you report must be contained in the current collection or in the record stream established by the RSE in the REPORT statement.

## 6.6 SET Statement (RW)

Controls the report header and defines the size of report pages and the length of the report. With Report Writer SET statements, you can specify:

- The report header—the report name (if any), the date, and page numbering
- The size of report pages—the number of columns and the number of lines per page
- The length of the report—the maximum number of lines and the maximum number of pages
- Whether or not column headers are printed

### Format

For naming the report:

```
SET REPORT_NAME = { "string"[/...]
                   *.prompt }
```

For controlling the printing of default page numbers:

```
SET [ NUMBER
     NO NUMBER ]
```

For specifying the beginning page number at the upper right of a page:

```
SET NUMBER = { n
               *.prompt }
```

For controlling the printing of a date:

```
SET NO DATE
```

(continued on next page)

# SET

## Format (Cont.)

For specifying a data or string at the upper right of each page:

```
SET DATE = "string"
```

For controlling the printing of the entire report headers:

```
SET NO REPORT_HEADER
```

For controlling the printing of column headers:

```
SET NO COLUMN_HEADER
```

For specifying page width or length, or overall report length:

```
SET { { COLUMNS_PAGE = } { LINES_PAGE = } { MAX_LINES = } { MAX_PAGES = } } { n } { *.prompt } { [...] }
```

## Arguments

Table 6-7 summarizes information about each type of SET statement.

The first column indicates the general objective for the setting.

The next three columns indicate the argument, the function of the statement, and the Report Writer's default setting if the statement is not used.

The Prompt Option column indicates whether or not a prompting value expression may be included with a form of the SET statement. If you include a prompt, the Report Writer prompts you for a value when it processes the report specification.

The Maximum Value column indicates the largest value you can specify for a form of the SET statement.

Table 6-7: SET Statement Arguments

Setting For	Argument	Function	Default	Prompt Option	Maximum Value
Report Header	REPORT_NAME	Centers a report name on the first line of each page.	No report name	Yes (Response to prompt must be enclosed in quotation marks.)	—
	DATE	Specifies a date or string that is printed on the upper right of each page.	Current system date	No	—
	NO DATE	Suppresses the printing of a date.	Current system date	No	—
	NUMBER	Causes the printing of a page number below the date.	Current page number	Yes	99,999
	NO NUMBER	Suppresses the printing of a page number.	Current page number	No	—
	* NO REPORT_HEADER	Suppresses the printing of report name, date, column headers, and page number on each page.	Header printed on each page	No	—
<p>* SET NO REPORT_HEADER overrides all other SET commands that control or specify elements in the report header.</p>					

(continued on next page)



# SET

**Table 6-7: SET Statement Arguments (Cont.)**

Setting For	Argument	Function	Default	Prompt Option	Maximum Value
Column Headers	NO COLUMN_ HEADER	Suppresses the printing of column headers.	Headers printed on each page	No	—
	COLUMNS_ PAGE	Specifies the page width in columns.	Value at DCL or 80 columns	Yes	255 columns
Page Size	LINES_ PAGE	Specifies the page length in number of lines.	60 lines	Yes	About 2 billion lines
	MAX_ LINES	Specifies the maximum lines for the report.	No line limit	Yes	About 2 billion lines
Report Size	** MAX_ PAGES	Specifies the maximum pages for the report.	No page limit	Yes	About 2 billion pages
<p>** Although the maximum number of pages is about 2 billion, the maximum page number printed at the upper right of a page is 99,999.</p>					

## Usage Notes

- DATATRIEVE calculates the maximum lines set for a report at the end of a page, not in the middle, even if the MAX\_ LINES setting is reached in the middle of a page. If DATATRIEVE reaches the MAX\_ LINES setting mid-page, it prints out the full page before stopping the report.

- If you embed a SET REPORT\_NAME statement with a prompting value expression in a compound statement that also has other prompts, DATATRIEVE may execute the prompts in an unexpected order. Consider the following example:

```
WHILE ORDER_NUM NOT STARTING WITH "1"  
  BEGIN  
  ORDER_NUM = .*"Order Number"  
  REPORT JOBITH ON .*"FILE,LP: OR IT"  
  SET REPORT_NAME= .*"Report Name"  
  PRINT JOBNBR  
  END_REPORT  
END
```

In this case, the prompt for "Report Name" occurs before either of the other two prompts.

This is because DATATRIEVE processes the entire compound statement (beginning with the WHILE statement) at one time. However, in order to process a REPORT statement, DATATRIEVE must know the length of the report name first so that it can format the report. This means that DATATRIEVE must prompt for the report header out of sequence with the other prompts. DATATRIEVE then prompts for the other values in the same order as the prompting value expressions appear.



# Listing of Record Definitions **A**

The following are listings of the record definitions used in the examples in this book. The definitions are listed alphabetically by record names.

## A.1 CATS\_\_REC Record Definition

```
DTR> SHOW CATS_REC
RECORD CATS_REC USING
01 TOP.
   05 CATEGORY    PIC 9.
   05 DESC        PIC X(10).
   05 BUDGET      PIC 99999V99
                  EDIT_STRING IS $$$,###.99.
;
```

## A.2 CHECKS\_\_REC Record Definition

```
DTR> SHOW CHECKS_REC
RECORD CHECKS_REC USING
01 TOP.
   05 CAT PIC 9.
   05 NUM      PIC 999
               EDIT_STRING Z29.
   05 CHECK_DATE USAGE DATE.
   05 AMOUNT PIC 999V99
               EDIT_STRING IS ####.99.
;
```

### A.3 EMP\_REC Record Definition

```
DTR) SHOW EMP_REC
RECORD EMP_REC USING
  01 EMP_REC.
    03 NAME.
      05 LAST_NAME      PIC X(10).
      05 FIRST_NAME     PIC X(10).
    03 NUMBER_JOBS     PIC 9.
    03 JOB_HISTORY     OCCURS 0 TO 9 TIMES DEPENDING ON NUMBER_JOBS.
      05 OLD_JOB        PIC X(3).
      05 OLD_DATE       USAGE DATE
                        QUERY_HEADER "EFFECTIVE"/"DATE".
;
```

### A.4 EMP\_REVIEW\_REC Record Definition

```
DTR) SHOW EMP_REVIEW_REC
RECORD EMP_REVIEW_REC USING
  01 EMP_REVIEW_REC.
    05 ID              PIC IS 9(5).
    05 NAME            PIC IS X(10).
    05 JOB             PIC IS X(15).
    05 EVALUATION     PIC IS X(60)
                      QUERY_NAME IS EVAL.
    05 EVAL_DATE      USAGE IS DATE
                      EDIT_STRING IS DD-MMM-YY.
;
```

### A.5 PAYABLES\_REC Record Definition

```
DTR) SHOW PAYABLES_REC
RECORD PAYABLES_REC USING
  01 PAYABLE.
    05 ORDER_NUM      PIC 9(7).
    05 TYPE.
      10 MANUFACTURER PIC IS X(10)
        QUERY_NAME IS BUILDER
        QUERY_HEADER IS "VENDOR".
      10 MODEL         PIC IS X(10)
        QUERY_HEADER IS "ITEM_TYPE".
    05 WHSLE_PRICE    PIC 9(5)
                      EDIT_STRING IS $$$,$$$.
    05 ITEMS_RECEIVED USAGE IS DATE
                      MISSING VALUE IS 010101
                      EDIT_STRING IS NN/DD/YY?"NO GOODS".
    05 INVOICE_DUE    USAGE IS DATE
                      MISSING VALUE IS 010101
                      EDIT_STRING IS NN/DD/YY?"NO INVCE".
    05 BILL_PAID      USAGE IS DATE
                      MISSING VALUE IS 010101
                      EDIT_STRING IS NN/DD/YY?"NOT PAID".
    05 AGE COMPUTED BY FN$FLOOR(("TODAY" - INVOICE_DUE)/30)
                      EDIT_STRING IS Z9.
;
```



## A.6 PERSONNEL\_REC Record Definition

```
DTR> SHOW PERSONNEL_REC
RECORD PERSONNEL_REC USING
01 PERSONNEL_REC.
  05 ID PIC IS 9(5).
  05 NAME.
    10 FIRST_NAME PIC IS X(10).
    10 LAST_NAME PIC IS X(10).
  05 DEPT PIC IS XXX.
  05 START_DATE USAGE IS DATE.
  05 SALARY PIC IS 99999
    EDIT_STRING IS $$$,$$$..
  05 STATUS PIC IS X(11)
    VALID IF STATUS EQ "TRAINEE","EXPERIENCED".
  05 SUP_ID PIC IS 9(5)
    MISSING VALUE IS 00000.
```

## A.7 SALES\_REC Record Definitions

There are two definitions for SALES\_REC shown here: the original definition and the expanded definition at the end of the chapter on mastering report writing techniques.

### A.7.1 SALES\_REC Record Definition (Original)

```
DTR> SHOW SALES_REC
RECORD SALES_REC USING
01 SALESREC.
  05 ID PIC IS 9(5).
  05 SALES_NAME PIC IS X(20).
    05 START_DATE USAGE IS DATE.
  05 MONTHS_EMP COMPUTED BY ("TODAY" - START_DATE)/30
    EDIT_STRING IS ZZ9.
  05 AMOUNT PIC IS 9(5)V99
    EDIT_STRING IS $$$,$$$..99.
```

## A.7.2 SALES\_REC Record Definition (Expanded)

```
DTR> SHOW SALES_REC
RECORD SALES_REC USING
  01 SALESREC.
    05 ID          PIC IS 9(5).
    05 SALES_NAME  PIC IS X(20).
                   05 START_DATE  USAGE IS DATE.
    05 MONTHS_EMP  COMPUTED BY ("TODAY" - START_DATE)/30
                   EDIT_STRING IS ZZ9.
    05 AMOUNT      PIC IS 9(5)V99
                   EDIT_STRING IS $$$,$$$99.
    05 COMM_PCT    COMPUTED BY
                   CHOICE
                   MONTHS_EMP LE 6 AND AMOUNT > 5000 THEN 10
                   MONTHS_EMP LE 6 THEN 05
                   AMOUNT > 10000 THEN 12
                   ELSE 07
                   END_CHOICE
                   EDIT_STRING IS Z9%.
    05 RATING      PIC IS X(11)
                   COMPUTED BY
                   CHOICE
                   MONTHS_EMP LE 6 AND AMOUNT > 5000 THEN "ABOVE QUOTA"
                   AMOUNT > 10000 THEN "ABOVE QUOTA"
                   ELSE "BELOW QUOTA"
                   END_CHOICE.
```

## A.8 WAGES\_REC Record Definition

```
DTR> SHOW WAGES_REC
RECORD WAGES_REC USING
  01 WAGE.
    05 LAST_NAME   PIC IS X(10)
                   QUERY_NAME IS L_NAME.
    05 GROSS_PAY   PIC IS S9(4)V99
                   EDIT_STRING IS $$,$$$99
                   QUERY_NAME IS GROSS.
    05 FICA        PIC IS 9(3)V99
                   EDIT_STRING IS $$$99.
    05 FEDERAL_TAX PIC IS 9(3)V99
                   EDIT_STRING IS $$$99
                   QUERY_NAME IS FED.
    05 STATE_TAX   PIC IS 9(3)V99
                   EDIT_STRING IS $$$99
                   QUERY_NAME IS STATE.
    05 NET_PAY     COMPUTED BY
                   (GROSS_PAY - (FICA + FEDERAL_TAX + STATE_TAX))
                   EDIT_STRING IS $$$,$$$99.
```

## A.9 YACHT Record Definition

```
DTR> SHOW YACHT
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 MISSING VALUE IS 0.
    06 PRICE PIC 99999
      MISSING VALUE IS 0
      VALID IF PRICE>DISP.*1.3 OR PRICE EQ 0
      EDIT_STRING IS $$$,$$$.
```



## Index

In this index, a page number followed by a "t" indicates a table reference. A page number followed by an "f" indicates a figure reference. A page number followed by an "e" indicates an example reference.

### A

- Accounts payable sample report, 2-15 to 2-16
- Accounts receivable sample report, 3-13
- AT BOTTOM statement, 3-3 to 3-9, 6-3 to 6-6
  - See also* Control groups format, 6-3
  - header elements, 6-3t to 6-5t
  - OF field-name, 6-3, 6-5, 6-6
  - OF REPORT, 6-6
  - summary elements, 2-22, 6-3t to 6-5t
- AT TOP statement, 3-3 to 3-5, 3-21 to 3-25, 6-7 to 6-11
  - See also* Control groups

- format, 6-7
- header elements, 6-8t to 6-10t
- OF field-name, 6-7, 6-11
- OF PAGE, 6-6, 6-11
- OF REPORT, 6-11
- summary elements, 2-22, 6-8t to 6-10t

- AVERAGE statistical operator, 2-22 to 2-24
  - in AT BOTTOM statement, 6-3t
  - in AT TOP statement, 6-8t

### B

- Bill-of-materials sample report, 4-4 to 4-5
- Boats sample report, 1-3
- Budget sample report, 3-17 to 3-20

### C

- CHOICE value expression, 3-33 to 3-39
  - in record definitions, 3-35 to 3-39
- COL print list element, 2-13
  - in AT BOTTOM statement, 6-3t
  - in AT TOP statement, 6-8t
- Column headers, 2-13, 2-19 to 2-21
  - See also* Headers
  - print restrictions, 6-16
  - specifying, 2-14, 2-20 to 2-21

- suppressing, 2-20
- width, 2-17
- COLUMN\_HEADER element, 3-24
  - in AT TOP statement, 6-8t
- Command files
  - invoking reports with, 2-24 to 2-26
- COMPUTED BY clause
  - with CHOICE value expression, 3-35 to 3-39
- Concatenation characters, 3-4
- CONTROL C to exit Report Writer, 2-4
- Control groups, 3-1 to 3-15
  - based on dates, 3-10 to 3-13
  - DBMS, 4-6 to 4-7
  - group summaries
    - Report Writer, 3-8 to 3-9
    - SUM statement, 3-9 to 3-10
  - multiple levels, 3-5 to 3-7
  - sample report, 3-3 to 3-5
  - sort keys, 3-2 to 3-5, 6-11
- CONTROL Z to exit the editor, 2-4
- Correcting errors, 2-4
- COUNT statistical operator, 2-22 to 2-24
  - in AT BOTTOM statement, 6-4t
  - in AT TOP statement, 6-8t
- CROSS clause, 4-6, 5-5, 5-11 to 5-15
- CTRL/C
  - to exit Report Writer, 2-4
- CTRL/Z
  - to exit the editor, 2-4

## D

- Database
  - See DBMS reports
  - See Relational database reports
- Dates, 2-9 to 2-13
  - assigning, 2-11
  - default, 2-9
  - suppressing, 2-12, 2-13
- DBMS reports, 1-1, 4-1 to 4-7
  - CDD demonstration directory, 4-1
  - control groups, 4-6 to 4-7

- PARTS\_DB sample database, 4-1
- procedures, 4-4 to 4-5
- samples
  - bill-of-materials, 4-4 to 4-5
  - personnel, 4-2 to 4-4
  - responsibility, 4-6 to 4-7
- Detail lines, 2-13 to 2-21
- content, 2-14 to 2-16
- edit strings
  - See Edit strings
- field values, 2-14
- format, 2-16 to 2-21
- print item order, 2-16
- print item position, 2-17
- value expressions, 2-14
- varying, 3-33 to 3-39

## E

- Edit strings, 2-15
  - negative values, 2-18
  - staged output, 2-18 to 2-19
  - text, 2-18 to 2-19
- Editing
  - report specifications, 2-4
- Employee history sample report, 5-4 to 5-6
- Employee profile sample report, 5-9 to 5-11
- END\_REPORT statement (RW), 2-3, 6-12
  - format, 6-12
- Exiting
  - Report Writer, 2-3 to 2-4, 6-12

## F

- Field names
  - in AT BOTTOM statement, 6-4t
  - in AT TOP statement, 6-8t
- FN\$WIDTH
  - using to set screen width, 2-8
- Formatting page size
  - See Page formats



## G

Global variables, 5-15

## H

Header elements, 6-3, 6-7

Headers, 2-9 to 2-13

*See also* Column headers  
assigning a date, 2-11

AT TOP statement

REPORT\_HEADER element,  
6-9t

COLUMN\_HEADER

in AT TOP statement, 6-8t

dates

suppressing, 2-13

default format, 2-9 to 2-10

naming the report, 2-10 to 2-11

page, 3-24 to 3-25

page numbering

*See* Page numbering

suppressing, 2-13

SET statement (RW), 6-21

specifying, 6-15

suppressing a date, 2-12

Hierarchical records

*See* List fields

## I

Index keys

sort order of records, 6-6

Introductory sample report, 1-5 to 1-7

Invoking Report Writer, 2-2 to 2-3

## J

Job profile sample report, 5-12 to 5-15

## L

List fields

flattening hierarchies, 3-29 to 3-30  
reporting data from, 3-20 to 3-21,

3-28 to 3-32

statistical expressions, 3-21

using SET SEARCH, 3-31 to 3-32

## M

MAX (maximum value) statistical  
operator, 2-24 to 2-25

in AT BOTTOM statement, 6-4t

in AT TOP statement, 6-8t

MIN (minimum value) statistical oper-  
ator, 2-24 to 2-25

in AT BOTTOM statement, 6-4t

in AT TOP statement, 6-8t

## N

Naming reports, 2-10 to 2-11

NEW\_PAGE print list element, 2-13,  
3-22, 3-25

in AT BOTTOM statement, 6-4t

in AT TOP statement, 6-8t

NEW\_SECTION element

in AT BOTTOM statement, 6-4t

in AT TOP statement, 6-9t

Numbers, page

*See* Page numbering

## O

OCCURS FOR clause, 5-8

OF clause, 6-15

ON statement, 2-6

Output

file specification defaults, 6-20t

selecting the output medium, 2-4 to  
2-6

file, 2-5

multiple outputs, 2-6

printer, 2-5

prompt option, 2-6

terminal, 2-5

OVER clause of RSE, 5-5, 5-11 to 5-15

## P

Page formats, 2-7 to 2-9

default, 2-7

- headers
  - See* Headers
- line limit, 2-9
- numbers
  - See* Page numbering
- page length, 2-8
- page limit, 2-9
- page width, 2-7
- SET statement (RW), 6-21 to 6-25
- summary, 6-23t
- Page headings, 3-24 to 3-25
- Page numbering, 2-9 to 2-13
  - assigning, 2-12
  - default, 2-9
  - suppressing, 2-12, 2-13
- Personnel sample report, 4-2 to 4-4
- Print list elements
  - COL, 2-13, 6-14t
    - in AT BOTTOM statement, 6-3t
    - in AT TOP statement, 6-8t
  - field name, 6-14t
  - modifier, 6-14t
  - NEW\_PAGE, 2-13, 6-14t
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-8t
  - SKIP, 2-13, 6-14t
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-9t
  - SPACE, 2-13, 6-14t
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-9t
  - summary, 6-14t
  - TAB, 2-13, 6-14t
    - in AT BOTTOM statement, 6-5t
    - in AT TOP statement, 6-9t
- Print list modifiers, 6-15t
- Print lists, inner, 3-18, 3-32
- PRINT statement (RW), 1-4, 6-13 to 6-18
  - See also* Detail lines
  - differences from DATATRIEVE, 6-15
  - format, 2-13, 6-13
  - print list elements, 2-13, 6-13 to 6-14t

- print list modifiers, 6-15t
- Printing totals of rows, 3-26 to 3-27
- Procedures
  - for DBMS reports, 4-4 to 4-5
  - for relational database reports, 5-15 to 5-17
  - for reports, 1-7 to 1-9
- Prompts
  - during report processing, 2-3
  - in reports, 1-7 to 1-9
  - to select the output medium, 2-6

## R

- Record streams, 3-18
- Relational database reports, 1-1, 5-1 to 5-17
  - CDD demonstration directory, 5-1
  - collections, 5-12 to 5-15
  - CROSS clause, 5-5, 5-11 to 5-15
  - defining a database, 5-3 to 5-4
  - inner print lists, 5-12 to 5-15
  - OCCURS FOR clause, 5-8
  - OVER clause, 5-5, 5-11 to 5-15
  - PERSONNEL sample database, 5-2
  - procedures, 5-15 to 5-17
  - reading relations, 5-3 to 5-4
  - relations, 5-2
  - samples
    - employee history, 5-4 to 5-6
    - employee profile, 5-9 to 5-11
    - job profile, 5-12 to 5-15
  - using multiple relations, 5-6 to 5-15
    - CROSS and OVER clauses, 5-11 to 5-15
  - view domains, 5-7 to 5-11
    - sample, 5-7 to 5-9
- Report specifications, 2-1, 2-2
  - AT BOTTOM statement, 6-3 to 6-6
    - See also* AT BOTTOM statement
  - AT TOP statement, 6-7 to 6-11
    - See also* AT TOP statement
  - column headers

*See* Column headers  
complex, 3-33 to 3-39  
detail lines  
    *See* Detail lines  
editing, 2-4  
END\_REPORT statement (RW),  
    2-3, 6-12  
field names, 6-4t, 6-8t  
headers, 2-9 to 2-13, 6-8t, 6-15, 6-21  
    *See also* Column headers  
optional statements, 2-1  
page formats, 6-23t  
page size, 2-7 to 2-9  
print list elements, 2-13, 6-3t, 6-4t,  
    6-5t, 6-8t, 6-9t, 6-14t  
print list modifiers, 6-15t  
PRINT statement (RW), 2-13 to  
    2-16, 6-13 to 6-18  
REPORT statement, 2-2 to 2-3,  
    6-19 to 6-20  
required statements, 2-1  
selecting the output medium, 2-4 to  
    2-6  
    file, 2-5  
    multiple outputs, 2-6  
    printer, 2-5  
    prompt option, 2-6  
    terminal, 2-5  
SET statement (RW), 2-7 to 2-13,  
    6-21 to 6-25  
statistical operators, 2-22 to 2-26, 6-  
    3t, 6-4t, 6-5t, 6-8t, 6-9t, 6-10t  
summary lines, 2-22 to 2-26  
REPORT statement, 2-2 to 2-3, 6-19  
    to 6-20  
    format, 6-19  
    output file specifications, 6-20t  
    selecting the output medium, 2-4 to  
        2-6  
        file, 2-5  
        multiple outputs, 2-6  
        printer, 2-5  
        prompt option, 2-6  
        terminal, 2-5  
Report Writer

    capabilities, 1-2  
    correcting mistakes, 2-4  
    exiting, 2-3 to 2-4, 6-12  
    invoking, 2-2 to 2-3, 6-19  
    summary of statements, 6-2t  
REPORT\_HEADER element, 3-24,  
    3-25  
    in AT TOP statement, 6-9t  
Responsibility (DBMS) sample report,  
    4-6 to 4-7  
RUNNING COUNT statistical opera-  
    tor, 2-15 to 2-16, 2-22  
    in AT BOTTOM statement, 6-4t  
    in AT TOP statement, 6-9t  
RUNNING TOTAL statistical opera-  
    tor, 2-15 to 2-16, 2-22

## S

Sales sample report, 3-33 to 3-39  
Sample reports  
    accounts payable, 2-15 to 2-16  
    accounts receivable, 3-13  
    bill-of-materials, 4-4 to 4-5  
    boats, 1-3  
    budget, 3-17 to 3-20  
    employee history, 5-4 to 5-6  
    employee profile, 5-9 to 5-11  
    introductory, 1-5 to 1-7  
    job profile, 5-12 to 5-15  
    personnel, 4-2 to 4-4  
    responsibility (DBMS), 4-6 to 4-7  
    sales, 3-33 to 3-39  
    yachts, 1-9 to 1-10  
Screen width  
    setting, 2-8  
SET statement (RW), 6-21 to 6-25  
    COLUMNS\_PAGE, 2-7 to 2-8,  
        6-23t  
        prompt option, 2-8  
    DATE, 2-11, 6-23t  
    format, 6-21  
    LINES\_PAGE, 2-8, 6-23t  
        prompt option, 2-8  
    MAX\_LINES, 2-9, 6-23t

- prompt option, 2-9
- MAX\_PAGES, 2-9, 6-23t
  - prompt option, 2-9
- NO COLUMN\_HEADER, 2-20, 6-23t
- NO DATE, 2-12, 6-23t
- NO NUMBER, 2-12, 6-23t
- NO REPORT\_HEADER, 2-13, 6-23t
- NUMBER, 2-12, 6-23t
- REPORT\_NAME, 2-10 to 2-11, 6-23t
  - prompt option, 2-11
  - summary, 6-23t
- SKIP print list element, 2-13
  - in AT BOTTOM statement, 6-4t
  - in AT TOP statement, 6-9t
- Sort keys, 3-2
  - See also Control groups
- SPACE print list element, 2-13, 2-17
  - in AT BOTTOM statement, 6-4t
  - in AT TOP statement, 6-9t
- Staged output
  - See Edit strings
- Statistical expressions
  - with list fields, 3-21
- Statistical operators, 1-9, 2-22 to 2-26
  - AVERAGE, 1-10, 2-22 to 2-24
    - in AT BOTTOM statement, 6-3t
    - in AT TOP statement, 6-8t
  - COUNT, 1-10, 2-22 to 2-24
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-8t
  - MAX (maximum value), 2-24 to 2-25
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-8t
  - MIN (minimum value), 2-24 to 2-25
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-8t
  - RUNNING COUNT, 2-15 to 2-16, 2-22
    - in AT BOTTOM statement, 6-4t
    - in AT TOP statement, 6-9t

- RUNNING TOTAL, 2-15 to 2-16, 2-22
- STD\_DEV (standard deviation), 2-24 to 2-25
  - in AT BOTTOM statement, 6-4t
  - in AT TOP statement, 6-9t
- TOTAL, 2-22 to 2-24
  - in AT BOTTOM statement, 6-5t
  - in AT TOP statement, 6-10t
- STD\_DEV (standard deviation) statistical operator, 2-24 to 2-25
  - in AT BOTTOM statement, 6-4t
  - in AT TOP statement, 6-9t
- SUM statement, 1-4, 3-9 to 3-10
- Summary elements, 6-3, 6-7
- Summary lines, 1-9 to 1-10, 2-22 to 2-26
- Suppressing column headers, 2-20
- Syntax formats
  - AT BOTTOM statement, 6-3
  - AT TOP statement, 6-7
  - END\_REPORT statement, 6-12
  - PRINT statement (RW), 6-13
  - REPORT statement, 6-19
  - SET statement (RW), 6-21

## T

- TAB print list element, 2-13
  - in AT BOTTOM statement, 6-5t
  - in AT TOP statement, 6-9t
- Title page, 3-21 to 3-24
- TOTAL statistical operator, 2-22 to 2-24
  - in AT BOTTOM statement, 6-5t
  - in AT TOP statement, 6-10t

## U

- USING clause specifying edit strings, 2-18

## V

- Value expressions, 2-14

in AT BOTTOM statement, 6-5t  
in AT TOP statement, 6-10t  
View domains, 5-7 to 5-11  
reporting data from, 3-20 to 3-21

**Y**  
Yachts sample report, 1-9 to 1-10





## How to Order Additional Documentation

If you live in:	Call:	or Write:
New Hampshire, Alaska	603-884-6660	Digital Equipment Corp. P.O. Box CS2008 Nashua, NH 03061-2698
Continental USA, Puerto Rico, Hawaii	1-800-258-1710	Same as above.
Canada (Ottawa-Hull)	613-234-7726	Digital Equipment Corp. 940 Belfast Road Ottawa, Ontario K1G 4C2 Attn: P&SG Business Manager or approved distributor
Canada (British Columbia)	1-800-267-6146	Same as above.
Canada (All other)	112-800-267-6146	Same as above.
All other areas	—	Digital Equipment Corp. Peripherals & Supplies Centers P&SG Business Manager c/o DIGITAL's local subsidiary

**Note:** Place prepaid orders from Puerto Rico with the local DIGITAL subsidiary (phone 809-754-7575).

Place internal orders with the Software Distribution Center, Digital Drive, Westminister, MA 01473-0471.





# Reader's Comments

VAX DATATRIEVE  
Guide to Writing Reports  
AA-P862C-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

<b>I rate this manual's:</b>	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less \_\_\_\_\_

What I like best about this manual is \_\_\_\_\_

What I like least about this manual is \_\_\_\_\_

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

I am using **Version** \_\_\_\_\_ of the software this manual describes.

Name/Title \_\_\_\_\_ Dept. \_\_\_\_\_

Company \_\_\_\_\_ Date \_\_\_\_\_

Mailing Address \_\_\_\_\_

Phone \_\_\_\_\_

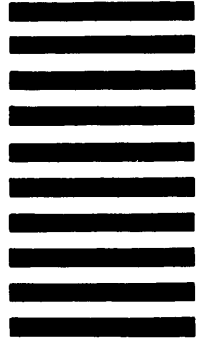


----- Do Not Tear - Fold Here and Tape -----

**digital**<sup>TM</sup>



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

DIGITAL EQUIPMENT CORPORATION  
Corporate User Publications—Spit Brook  
ZK01-3/J35  
110 SPIT BROOK ROAD  
NASHUA, NH 03062-9987



----- Do Not Tear - Fold Here -----

Cut Along Dotted Line



# Reader's Comments

VAX DATATRIEVE  
Guide to Writing Reports  
AA-P862C-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

<b>I rate this manual's:</b>	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less \_\_\_\_\_

What I like best about this manual is \_\_\_\_\_

What I like least about this manual is \_\_\_\_\_

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

I am using **Version** \_\_\_\_\_ of the software this manual describes.

Name/Title \_\_\_\_\_ Dept. \_\_\_\_\_

Company \_\_\_\_\_ Date \_\_\_\_\_

Mailing Address \_\_\_\_\_

Phone \_\_\_\_\_

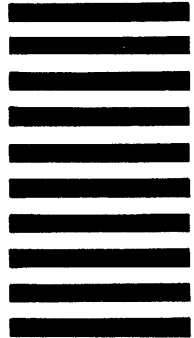


-- Do Not Tear - Fold Here and Tape --

**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

DIGITAL EQUIPMENT CORPORATION  
Corporate User Publications—Spit Brook  
ZK01-3/J35  
110 SPIT BROOK ROAD  
NASHUA, NH 03062-9987



-- Do Not Tear - Fold Here --

Cut Along Dotted Line







**digital**<sup>TM</sup>

DIGITAL EQUIPMENT CORPORATION

Printed in U.S.A.



## Reference Section

