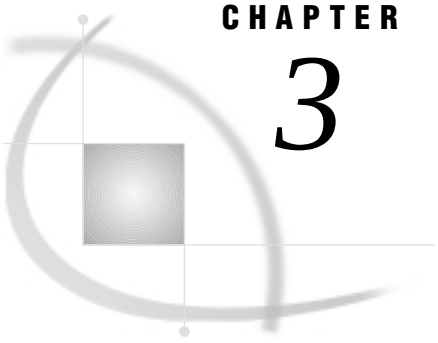


CHAPTER

3

Using ADABAS Data in SAS Programs

Introduction

An advantage of the SAS/ACCESS interface to ADABAS is that it enables the SAS System to read and write ADABAS data directly using SAS programs. This chapter presents examples using ADABAS data accessed through view descriptors as input data for SAS programs.

Throughout the examples, the SAS terms *variable* and *observation* are used instead of comparable ADABAS terms because this chapter illustrates using SAS System procedures and the DATA step. The examples include printing and charting data, using the SQL procedure to combine data from various sources, and updating a Version 6 SAS data file with ADABAS data. For more information about the SAS language and procedures used in the examples, refer to the books listed at the end of each section.

At the end of this chapter, “Performance Considerations” on page 36, presents some techniques for using view descriptors efficiently in SAS programs.

For definitions of all view descriptors referenced in this chapter, see Appendix 3, "Example Data." This appendix also contains the ADABAS data and SAS data files used in this book.

Reviewing Variables

If you want to use ADABAS data described by a view descriptor in your SAS program but cannot remember the variable names or formats and informats, you can use the CONTENTS or DATASETS procedures to display this information.

The following examples use the DATASETS procedure to give you information on the view descriptor VLIB.CUSPHON, which references the NATURAL DDM named CUSTOMERS.

```
proc datasets library=vlib memtype=view;
   contents data=cusphon;
quit;
```

Output 3.1 on page 18 shows the information for this example. The data described by VLIB.CUSPHON are shown in Output 3.9 on page 28.

Output 3.1 Using the DATASETS Procedure with a View Descriptor

```
                              The SAS System

                            DATASETS PROCEDURE

Data Set Name: VLIB.CUSPHON                          Observations:         .
Member Type:   VIEW                                  Variables:            3
Engine:        SASIOADB                              Indexes:              0
Created:       14:09 Friday, October 5, 1990         Observation Length:   80
Last Modified: 14:33 Friday, October 5, 1990         Deleted Observations: 0
Data Set Type:                                       Compressed:           NO
Label:

                -----Engine/Host Dependent Information-----


            -----Alphabetic List of Variables and Attributes-----

     #    Variable    Type    Len    Pos    Format    Informat    Label
     ------------------------------------------------------------------------
     1    CUSTNUM     Char      8      0    $8.       $8.         CUSTOMER
     3    NAME        Char     60     20    $60.      $60.        NAME
     2    PHONE       Char     12      8    $12.      $12.        TELEPHONE
```

Note the following points about this output:

- You cannot change a view descriptor's variable labels using the DATASETS procedure. The labels are generated to be the complete ADABAS data field name when the view descriptor is created and therefore cannot be overwritten.
- The Created date is the date the access descriptor for this view descriptor was created.
- The Last Modified date is the last time the view descriptor was updated.
- The Observations number field contains a null value.

For more information about the DATASETS procedure, see the *SAS Procedures Guide*.

Printing Data

Printing ADABAS data described by a view descriptor is like printing any other SAS data set, as shown in the following examples.

The following example contains the code for printing the ADABAS data described by the view descriptor VLIB.EMPINFO:

```
proc print data=vlib.empinfo;
   title "Brief Employee Information";
run;
```

VLIB.EMPINFO accesses data from the NATURAL DDM named EMPLOYEE. Output 3.2 on page 19 shows the output for this example.

Output 3.2 Results of the PRINT Procedure

Brief Employee Information

OBS	EMPID	DEPT	LASTNAME
1	119012	CSR010	WOLF-PROVENZA
2	120591	SHP002	HAMMERSTEIN
3	123456		VARGAS
4	127845	ACC024	MEDER
5	129540	SHP002	CHOULAI
6	135673	ACC013	HEMESLY
7	212916	CSR010	WACHBERGER
8	216382	SHP013	PURINTON
9	234967	CSR004	SMITH
10	237642	SHP013	BATTERSBY
11	239185	ACC024	DOS REMEDIOS
12	254896	CSR011	TAYLOR-HUNYADI
13	321783	CSR011	GONZALES
14	328140	ACC043	MEDINA-SIDONIA
15	346917	SHP013	SHIEKELESLAM
16	356134	ACC013	DUNNETT
17	423286	ACC024	MIFUNE
18	456910	CSR010	ARDIS
19	456921	SHP002	KRAUSE
20	457232	ACC013	LOVELL
21	459287	SHP024	RODRIGUES
22	677890	CSR010	NISHIMATSU-LYNCH

When you use the PRINT procedure, you may want to use the OBS= option, which enables you to specify the last observation to be processed. This is especially useful when the view descriptor describes large amounts of data or when you just want to see an example of the output. The following example uses the OBS= option to print the first five observations described by the view descriptor VLIB.CUSORDR.

```
proc print data=vlib.cusordr (obs=5);
   title "First Five Observations Described
          by VLIB.CUSORDR";
run;
```

VLIB.CUSORDR accesses data from the NATURAL DDM named ORDER. Output 3.3 on page 20 shows the result of this example.

Output 3.3 Results of Using the OBS= Option

First Five Observations Described by VLIB.CUSORDR

OBS	STOCKNUM	SHIPTO
1	9870	19876078
2	1279	39045213
3	8934	18543489
4	3478	29834248
5	2567	19783482

In addition to the OBS= option, the FIRSTOBS= option also works with view descriptors. The FIRSTOBS= option does not improve performance significantly because each observation must still be read and its position calculated. The POINT= option in the SET statement is not currently supported by the SAS/ACCESS interface to ADABAS.

For more information about the PRINT procedure, see the *SAS Procedures Guide*. For more information about the OBS= and FIRSTOBS= options, see the *SAS Language Reference: Dictionary*.

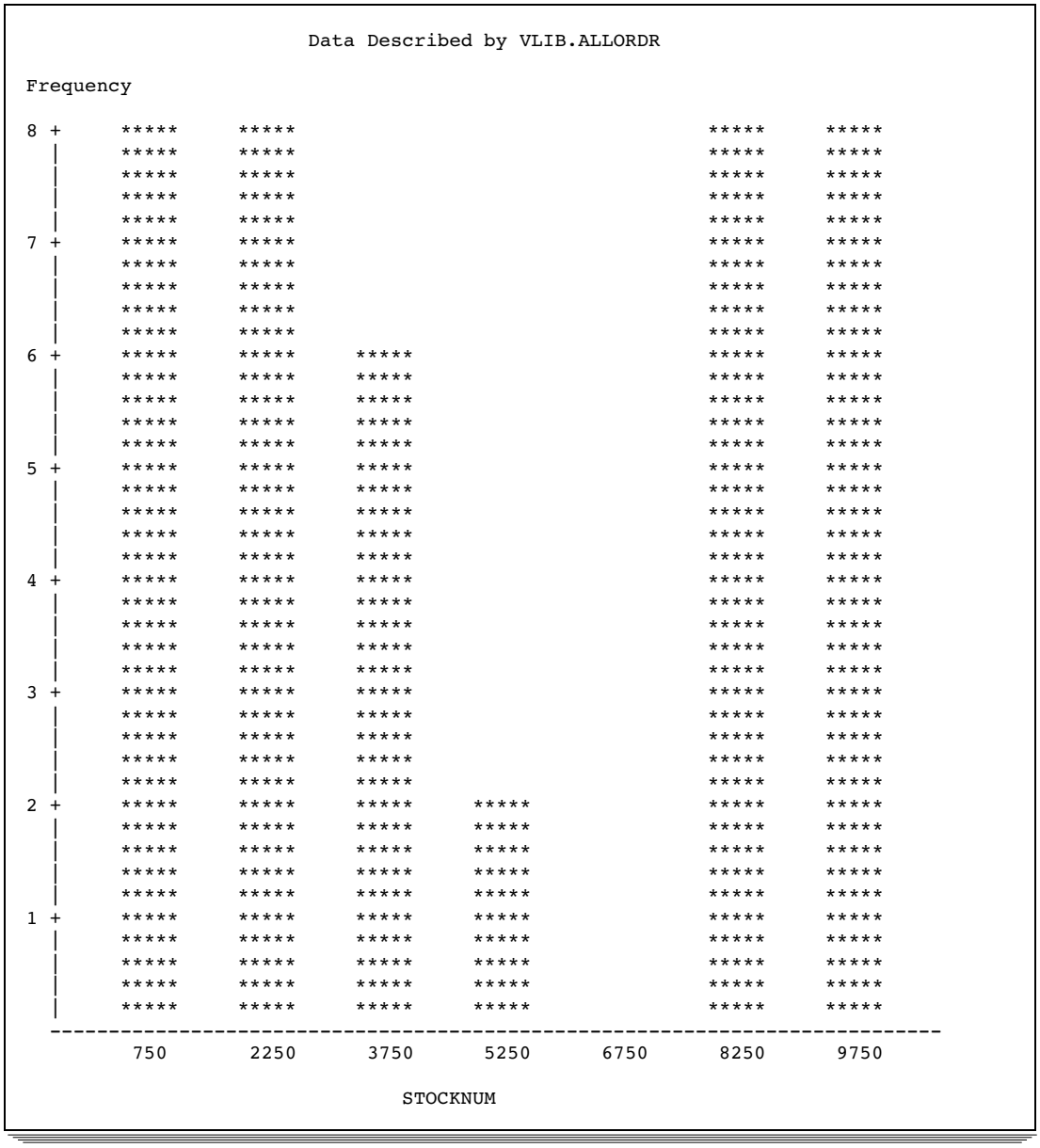
Charting Data

CHART procedure programs work with data described by view descriptors just as they do with other SAS data sets. The following examples use the view descriptor VLIB.ALLORDR to create a vertical bar chart of the number of orders per product.

```
proc chart data=vlib.allordr;
   vbar stocknum;
   title "Data Described by VLIB.ALLORDR";
run;
```

VLIB.ALLORDR accesses data from the NATURAL DDM named ORDER. Output 3.4 on page 21 shows the output for this example. STOCKNUM represents each product; the number of orders for each product is represented by the height of the bar.

Output 3.4 Vertical Bar Chart Showing Number of Orders per Product



For more information about the CHART procedure, see the *SAS Procedures Guide.*

If you have SAS/GRAPH software, you can create colored block charts, plots, and other graphics based on ADABAS data. See the *SAS/GRAPH Software: Reference* for more information about the types of graphics you can produce with SAS/GRAPH software.

Calculating Statistics

You can also use statistical procedures on ADABAS data. This section shows simple examples using the FREQ and MEANS procedures.

Using the FREQ Procedure

Suppose you wanted to find what percentage of your invoices went to each country so that you can decide where to increase your overseas marketing. The following example calculates the percentages of invoices for each country accessed by the NATURAL DDM named INVOICE, using the view descriptor VLIB.INV.

```
proc freq data=vlib.inv;
   tables country;
   title "Data Described by VLIB.INV";
run;
```

Output 3.5 on page 22 shows the one-way frequency table this example generates.

Output 3.5 Frequency Table for Variable COUNTRY Described by View Descriptor VLIB.INV

Data Described by VLIB.INV

COUNTRY

COUNTRY	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Argentina	2	11.8	2	11.8
Australia	1	5.9	3	17.6
Brazil	4	23.5	7	41.2
USA	10	58.8	17	100.0

Frequency Missing = 2

For more information about the FREQ procedure, see the *SAS Procedures Guide*.

Using the MEANS Procedure

In your analysis of recent orders, suppose you also wanted to determine some statistics for each USA customer. In the following SAS program, the view descriptor VLIB.USAORDR accesses data from the NATURAL DDM named ORDER, the SAS WHERE statement selects observations that have a SHIPTO value beginning with a 1, which indicates a USA customer, and the SAS BY statement sorts the data by order number. (Note that both ORDERNUM and SHIPTO are ADABAS descriptor data fields.)

The following example generates the mean and sum of the length of material ordered and the fabric charges for each USA customer. Also included are the number of observations (N) and the number of missing values (NMISS).

```
proc means data=vlib.usaordr mean sum n nmiss
     maxdec=0;
   where shipto like "1%";
   by ordernum;
   var length fabricch;
   title "Data Described by VLIB.USAORDR";
run;
```

Output 3.6 on page 24 shows the output for this example.

Output 3.6 Statistics on Fabric Length and Charges for Each USA Customer

Data Described by VLIB.USAORDR

ORDERNUM=11269

Variable	Label	N	Nmiss	Mean	Sum
LENGTH	LENGTH	1	0	690	690
FABRICCH	FABRICCHARGES	1	0	0	0

ORDERNUM=11271

Variable	Label	N	Nmiss	Mean	Sum
LENGTH	LENGTH	1	0	110	110
FABRICCH	FABRICCHARGES	1	0	11063836	11063836

ORDERNUM=11273

Variable	Label	N	Nmiss	Mean	Sum
LENGTH	LENGTH	1	0	450	450
FABRICCH	FABRICCHARGES	1	0	252149	252149

ORDERNUM=11274

Variable	Label	N	Nmiss	Mean	Sum
LENGTH	LENGTH	1	0	1000	1000
FABRICCH	FABRICCHARGES	1	0	0	0

ORDERNUM=11276

Variable	Label	N	Nmiss	Mean	Sum
LENGTH	LENGTH	1	0	1500	1500
FABRICCH	FABRICCHARGES	1	0	1934460	1934460

ORDERNUM=11278

Variable	Label	N	Nmiss	Mean	Sum
LENGTH	LENGTH	1	0	2500	2500
FABRICCH	FABRICCHARGES	1	0	1400825	1400825

For more information about the MEANS procedure, see the *SAS Procedures Guide.*

Using the RANK Procedure

You can also use more advanced statistics procedures on ADABAS data. The following example uses the RANK procedure to calculate the order of birthdays for a set of employees. This example creates a SAS data file MYDATA.RANKEX from the view descriptor VLIB.EMPS and assigns the name DATERANK to the new variable (in the data file) created by the procedure.

```
proc rank data=vlib.emps out=mydata.rankex;
   var birthdat;
   ranks daterank;
run;
proc print data=mydata.rankex;
   title "Order of Employee Birthdays";
run;
```

VLIB.EMPS accesses data from the NATURAL DDM named EMPLOYEE. Output 3.7 on page 25 shows the result of this example.

Output 3.7 Ranking of Employee Birthdays

Order of Employee Birthdays

OBS	EMPID	JOBCODE	BIRTHDAT	LASTNAME	DATERANK
1	456910	602	24SEP53	ARDIS	5
2	237642	602	13MAR54	BATTERSBY	6
3	239185	602	28AUG59	DOS REMEDIOS	7
4	321783	602	03JUN35	GONZALES	2
5	120591	602	12FEB46	HAMMERSTEIN	4
6	135673	602	21MAR61	HEMESLY	8
7	456921	602	12MAY62	KRAUSE	9
8	457232	602	15OCT63	LOVELL	11
9	423286	602	31OCT64	MIFUNE	12
10	216382	602	24JUL63	PURINTON	10
11	234967	602	21DEC67	SMITH	13
12	212916	602	29MAY28	WACHBERGER	1
13	119012	602	05JAN46	WOLF-PROVENZA	3

For more information about the RANK procedure and other advanced statistics procedures, see the *SAS Procedures Guide*.

Selecting and Combining Data

The great majority of SAS programs select and combine data from various sources. The method you use depends on the configuration of the data. The next three examples show you how to select and combine data using two different methods. When choosing

between these methods, you should consider the issues described in “Performance Considerations” on page 36.

Using the WHERE Statement

Suppose you have two view descriptors, VLIB.USAINV and VLIB.FORINV, that list the invoices for USA and foreign customers, respectively. You can use the SET statement to concatenate these files into a SAS data file containing information on customers who have not paid their bills and whose bills amount to at least $300,000.

The following example contains the code to create the SAS data file containing the information you want on the customers:

```
data notpaid(keep=invoicen billedto amtbille
     billedon paidon);
   set vlib.usainv vlib.forinv;
   where paidon is missing and
      amtbille>=300000;
run;
proc print;
   title "High Bills--Not Paid";
run;
```

In the SAS WHERE statement, you must use the SAS variable names, not the ADABAS data field names. Both VLIB.USAINV and VLIB.FORINV access data in the NATURAL DDM named INVOICE. Output 3.8 on page 26 shows the result of the new temporary data file, WORK.NOTPAID.

Output 3.8 NOTPAID Data File Created Using a SAS WHERE Statement

High Bills--Not Paid

OBS	INVOICEN	BILLEDTO	AMTBILLE	BILLEDON	PAIDON
1	12102	18543489	11063836.00	17NOV88	.
2	11286	43459747	12679156.00	10OCT88	.
3	12051	39045213	1340738760.90	02NOV88	.
4	12471	39045213	1340738760.90	27DEC88	.
5	12476	38763919	34891210.20	24DEC88	.

The first line of the DATA step uses the KEEP= data set option. This option works with view descriptors just as it works with other SAS data sets; that is, the KEEP= option specifies that you want only the listed variables to be included in the new data file, NOTPAID, although you can use the other variables within the DATA step.

Notice that the WHERE statement includes two conditions to be met. First, it selects only observations that have missing values for the variable PAIDON. As you can see, it is important to know how the ADABAS data are configured before you can use this data in a SAS program.

Second, the WHERE statement requires that the amount in each bill be higher than a certain figure. Again, you need to be familiar with the ADABAS data so that you can determine a reasonable figure for this expression.

When referencing a view descriptor in a SAS procedure or DATA step, it is more efficient to use a SAS WHERE statement than to use a subsetting IF statement. A DATA step or SAS procedure passes the SAS WHERE statement as a WHERE clause to the interface view engine, which adds it (using the Boolean operator AND) to any WHERE clause defined in the view descriptor. The view descriptor is then passed to ADABAS for processing. Processing ADABAS data using a WHERE clause may reduce the number of logical records read and therefore often improves performance.

For more information about the SAS WHERE statement, see the *SAS Language Reference: Dictionary*.

Using the SAS System SQL Procedure

This section provides two examples of using the SAS System SQL procedure on ADABAS data. The SQL procedure implements the Structured Query Language (SQL) in Version 7 of the SAS System and is included in base SAS software. The first example illustrates using the SQL procedure to combine data from three sources. The second example shows how to use the PROC SQL GROUP BY clause to create new variables from data described by a view descriptor.

Combining Data from Various Sources

The SQL procedure provides another way to select and combine data. For example, suppose you have the view descriptors VLIB.CUSPHON and VLIB.CUSORDR based on the NATURAL DDMs CUSTOMERS and ORDER, respectively, and a SAS data file, MYDATA.OUTOFSTK, that contains names and numbers of products that are out of stock. You can use the SQL procedure to join all these sources of data to form a single output file. The SAS WHERE or subsetting IF statements would not be appropriate in this case because you want to compare variables from several sources, rather than simply merge or concatenate the data.

The following example contains the code to print the view descriptors and the SAS data file:

```
proc print data=vlib.cusphon;
   title "Data Described by VLIB.CUSPHON";
run;
proc print data=vlib.cusordr;
   title "Data Described by VLIB.CUSORDR";
run;

proc print data=mydata.outofstk;
   title "SAS Data File MYDATA.OUTOFSTK";
run;
```

Output 3.9 on page 28, Output 3.10 on page 29, and Output 3.11 on page 30 show the results of the PRINT procedure performed on the data described by the view descriptors VLIB.CUSPHON and VLIB.CUSORDER and on the SAS data file MYDATA.OUTOFSTK.

Output 3.9 Data Described by the View Descriptor VLIB.CUSPHON

Data Described by VLIB.CUSPHON

OBS	CUSTNUM	PHONE
1	12345678	919/489-5682
2	14324742	408/629-0589
3	14569877	919/489-6792
4	14898029	301/760-2541
5	15432147	616/582-3906
6	18543489	512/478-0788
7	19783482	703/714-2900
8	19876078	209/686-3953
9	24589689	(012)736-202
10	26422096	4268-54-72
11	26984578	43-57-04
12	27654351	02/215-37-32
13	28710427	(021)570517
14	29834248	(0552)715311
15	31548901	406/422-3413
16	38763919	244-6324
17	39045213	012/302-1021
18	43290587	(02)933-3212
19	43459747	03/734-5111
20	46543295	(03)022-2332
21	46783280	3762855
22	48345514	213445

OBS	NAME
1	
2	SANTA CLARA VALLEY TECHNOLOGY SPECIALISTS
3	PRECISION PRODUCTS
4	UNIVERSITY BIOMEDICAL MATERIALS
5	GREAT LAKES LABORATORY EQUIPMENT MANUFACTURERS
6	LONE STAR STATE RESEARCH SUPPLIERS
7	TWENTY-FIRST CENTURY MATERIALS
8	SAN JOAQUIN SCIENTIFIC AND INDUSTRIAL SUPPLY, INC.
9	CENTAR ZA TECHNICKU I NAUCNU RESTAURIRANJE UMJETNINA
10	SOCIETE DE RECHERCHES POUR DE CHIRURGIE ORTHOPEDIQUE
11	INSTITUT FUR TEXTIL-FORSCHUNGS
12	INSTITUT DE RECHERCHE SCIENTIFIQUE MEDICALE
13	ANTONIE VAN LEEUWENHOEK VERENIGING VOOR MICROBIOLOGIE
14	BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY
15	NATIONAL COUNCIL FOR MATERIALS RESEARCH
16	INSTITUTO DE BIOLOGIA Y MEDICINA NUCLEAR
17	LABORATORIO DE PESQUISAS VETERNINARIAS DESIDERIO FINAMOR
18	HASSEI SAIBO GAKKAI
19	RESEARCH OUTFITTERS
20	WESTERN TECHNOLOGICAL SUPPLY
21	NGEE TECHNOLOGICAL INSTITUTE
22	GULF SCIENTIFIC SUPPLIES

Output 3.10 Data Described by the View Descriptor VLIB.CUSORDR

Data Described by VLIB.CUSORDR

OBS	STOCKNUM	SHIPTO
1	9870	19876078
2	1279	39045213
3	8934	18543489
4	3478	29834248
5	2567	19783482
6	4789	15432147
7	3478	29834248
8	1279	14324742
9	8934	31548901
10	2567	14898029
11	9870	48345514
12	1279	39045213
13	8934	18543489
14	2567	19783482
15	9870	18543489
16	3478	24589689
17	1279	38763919
18	8934	43459747
19	2567	15432147
20	9870	14324742
21	9870	19876078
22	1279	39045213
23	8934	18543489
24	3478	29834248
25	2567	19783482
26	4789	15432147
27	3478	29834248
28	1279	14324742
29	8934	31548901
30	2567	14898029
31	9870	48345514
32	1279	39045213
33	8934	18543489
34	2567	19783482
35	9870	18543489
36	3478	24589689
37	1279	38763919
38	8934	43459747
39	2567	15432147
40	9870	14324742

Output 3.11 Data in the SAS Data File MYDATA.OUTOFSTK

SAS Data File MYDATA.OUTOFSTK

OBS	FIBERNAM	FIBERNUM
1	olefin	3478
2	gold	8934
3	dacron	4789

The following SAS code selects and combines data from these three sources to create a PROC SQL view, SQL.BADORDR. The SQL.BADORDR view retrieves customer and product information that the sales department can use to notify customers of unavailable products.

```
proc sql;
create view sql.badordr as
   select cusphon.custnum, cusphon.name,
          cusphon.phone, cusordr.stocknum,
          outofstk.fibernam as product
      from vlib.cusphon, vlib.cusordr,
           mydata.outofstk
      where cusordr.stocknum=outofstk.fibernum
           and cusphon.custnum=cusordr.shipto
      order by cusphon.custnum, product;
   title "Data Described by SQL.BADORDR";
   select * from sql.badordr;
```

The CREATE VIEW statement incorporates a WHERE clause as part of its SELECT statement. The last SELECT statement retrieves and displays the PROC SQL view, SQL.BADORDR. To select all columns from the view, use an asterisk (*) in place of variable names. The order of the columns displayed matches the order of the columns as specified in the view descriptor SQL.BADORDR. (Note that an ORDER BY clause requires an ADABAS descriptor data field.)

Output 3.12 on page 31 shows the data described by the SQL.BADORDR view. Note that the SQL procedure uses the column labels in the output by default.

Output 3.12 Data Described by the PROC SQL View SQL.BADORDR

```
                    Data Described by SQL.BADORDR

CUSTOMER  NAME
TELEPHONE     STOCKNUM  PRODUCT
------------------------------------------------------------------------
15432147  GREAT LAKES LABORATORY EQUIPMENT MANUFACTURERS
616/582-3906      4789  dacron

15432147  GREAT LAKES LABORATORY EQUIPMENT MANUFACTURERS
616/582-3906      4789  dacron

18543489  LONE STAR STATE RESEARCH SUPPLIERS
512/478-0788      8934  gold

18543489  LONE STAR STATE RESEARCH SUPPLIERS
512/478-0788      8934  gold

18543489  LONE STAR STATE RESEARCH SUPPLIERS
512/478-0788      8934  gold

18543489  LONE STAR STATE RESEARCH SUPPLIERS
512/478-0788      8934  gold

24589689  CENTAR ZA TECHNICKU I NAUCNU RESTAURIRANJE UMJETNINA
(012)736-202      3478  olefin

24589689  CENTAR ZA TECHNICKU I NAUCNU RESTAURIRANJE UMJETNINA
(012)736-202      3478  olefin

29834248  BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY
(0552)715311      3478  olefin

29834248  BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY
(0552)715311      3478  olefin

29834248  BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY
(0552)715311      3478  olefin

29834248  BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY
(0552)715311      3478  olefin

31548901  NATIONAL COUNCIL FOR MATERIALS RESEARCH
406/422-3413      8934  gold

31548901  NATIONAL COUNCIL FOR MATERIALS RESEARCH
406/422-3413      8934  gold

43459747  RESEARCH OUTFITTERS
03/734-5111       8934  gold

43459747  RESEARCH OUTFITTERS
03/734-5111       8934  gold
```

The view SQL.BADORDR lists entries for all customers who have ordered out-of-stock products. However, it contains duplicate rows because some companies have ordered the same product more than once. To make the data more readable for the sales department, you can create a final SAS data file, MYDATA.BADNEWS, using the

results of the PROC SQL view as input in the SET statement and the special variable FIRST.PRODUCT. This variable identifies which row is the first in a particular BY group. You only need a customer's name once to notify them that a product is out of stock, regardless of the number of times the customer has placed an order for it.

```
data mydata.badnews;
   set sql.badordr;
   by custnum product;
   if first.product;
run;

proc print;
   title "MYDATA.BADNEWS Data File";
quit;
```

The data file MYDATA.BADNEWS contains an observation for each unique combination of customer and out-of-stock product. Output 3.13 on page 32 displays this data file.

Output 3.13 Data in the SAS Data File MYDATA.BADNEWS

MYDATA.BADNEWS Data File

OBS	CUSTNUM	NAME
1	15432147	GREAT LAKES LABORATORY EQUIPMENT MANUFACTURERS
2	18543489	LONE STAR STATE RESEARCH SUPPLIERS
3	24589689	CENTAR ZA TECHNICKU I NAUCNU RESTAURIRANJE UMJETNINA
4	29834248	BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY
5	31548901	NATIONAL COUNCIL FOR MATERIALS RESEARCH
6	43459747	RESEARCH OUTFITTERS

OBS	PHONE	STOCKNUM	PRODUCT
1	616/582-3906	4789	dacron
2	512/478-0788	8934	gold
3	(012)736-202	3478	olefin
4	(0552)715311	3478	olefin
5	406/422-3413	8934	gold
6	03/734-5111	8934	gold

For more information about the FIRST.*variable*, see the *SAS Language Reference: Dictionary*.

Creating New Variables with the GROUP BY Clause

It is often useful to create new variables with summarizing or variable functions such as AVG or SUM. Although you cannot use the ACCESS procedure to create new variables, you can easily use the SQL procedure with data described by a view descriptor to display output that contains new variables.

This example uses the SQL procedure to retrieve and manipulate data accessed by the view descriptor VLIB.ALLEMP, which accesses data in the NATURAL DDM named EMPLOYEE. When this *query* (as a SELECT statement is often called) is submitted, it calculates and displays the average salary for each department; the AVG function is the SQL procedure's equivalent of the SAS MEAN function.

```
proc sql;
   title "Average Salary Per Department";
   select distinct dept,
         avg(salary) label="Average Salary"
            format=dollar12.2
      from vlib.allemp
      where dept is not missing
      group by dept;
```

The order of the variables displayed matches the order of the variables as specified in the SELECT list of the query. Output 3.14 on page 33 shows the query's result.

Output 3.14 Data Retrieved by a PROC SQL Query

```
Average Salary Per Department
                    Average
DEPT                 Salary
--------------------
ACC013            $54,591.33
ACC024            $55,370.55
ACC043            $75,000.34
CSR004            $17,000.00
CSR010            $44,324.19
CSR011            $41,966.16
SHP002            $40,111.31
SHP013            $41,068.44
SHP024            $50,000.00
```

For more information about the SQL procedure, see the SQL chapter in the *SAS Procedures Guide*.

Updating a SAS Data File with ADABAS Data

You can update a SAS data file with ADABAS data described by a view descriptor, just as you can update a SAS data file with data from another data file. In this section, the term *transaction data* refers to the new data that are to be added to the original file. You can even do updates when the file to be updated is a Version 6 data file and the transaction data are from a Version 7 source.

Suppose you have a Version 6 data file, LIB6.BIRTHDAY, that contains employee ID numbers, last names, and birthdays. You want to update this data file with data

described by VLIB.EMPS, a view descriptor based on the EMPLOYEE DDM. To perform the update, enter the following SAS statements.

```
proc sort data=lib6.birthday;
   by lastname;
run;

proc print data=lib6.birthday;
   title "LIB6.BIRTHDAY Data File";
   format birthdat date7.;
run;

proc print data=vlib.emps;
   title "Data Described by VLIB.EMPS";
run;

data mydata.newbday;
   update lib6.birthday vlib.emps;
   by lastname;
run;

proc print;
   title 'MYDATA.NEWBDAY Data File';
run;
```

In this example, the new, updated SAS data file, MYDATA.NEWBDAY, is a Version 7 data file. It is stored in the Version 7 SAS data library associated with the libref MYDATA.

When the UPDATE statement references the view descriptor VLIB.EMPS and uses a BY statement in the DATA step, the BY statement causes a BY clause to be generated for the variable LASTNAME. (Note that a BY statement must reference an ADABAS descriptor data field.) Thus, the BY clause causes the ADABAS data to be presented to the SAS System in a sorted order for use in updating the MYDATA.NEWBDAY data file. However, the data file LIB6.BIRTHDAY had to be sorted before the update, because the UPDATE statement expects both the original file and the transaction file to be sorted by the BY variable.

Output 3.15 on page 35, Output 3.16 on page 35, and Output 3.17 on page 36 show the results of PRINT procedures on the original data file, the transaction data, and the updated data file.

Output 3.15 Data in the Data File to Be Updated, LIB6.BIRTHDAY

LIB6.BIRTHDAY Data File

OBS	EMPID	BIRTHDAT	LASTNAME
1	129540	31JUL60	CHOULAI
2	356134	25OCT60	DUNNETT
3	127845	25DEC43	MEDER
4	677890	24APR65	NISHIMATSU-LYNCH
5	459287	05JAN34	RODRIGUES
6	346917	15MAR50	SHIEKELESLAN
7	254896	06APR49	TAYLOR-HUNYADI

Output 3.16 Data Described by the View Descriptor VLIB.EMPS

Data Described by VLIB.EMPS

OBS	EMPID	JOBCODE	BIRTHDAT	LASTNAME
1	456910	602	24SEP53	ARDIS
2	237642	602	13MAR54	BATTERSBY
3	239185	602	28AUG59	DOS REMEDIOS
4	321783	602	03JUN35	GONZALES
5	120591	602	12FEB46	HAMMERSTEIN
6	135673	602	21MAR61	HEMESLY
7	456921	602	12MAY62	KRAUSE
8	457232	602	15OCT63	LOVELL
9	423286	602	31OCT64	MIFUNE
10	216382	602	24JUL63	PURINTON
11	234967	602	21DEC67	SMITH
12	212916	602	29MAY28	WACHBERGER
13	119012	602	05JAN46	WOLF-PROVENZA

Output 3.17 Data in the Updated Data File MYDATA. NEWBDAY

MYDATA.NEWBDAY Data File

OBS	EMPID	BIRTHDAT	LASTNAME	JOBCODE
1	456910	24SEP53	ARDIS	602
2	237642	13MAR54	BATTERSBY	602
3	129540	31JUL60	CHOULAI	.
4	239185	28AUG59	DOS REMEDIOS	602
5	356134	25OCT60	DUNNETT	.
6	321783	03JUN35	GONZALES	602
7	120591	12FEB46	HAMMERSTEIN	602
8	135673	21MAR61	HEMESLY	602
9	456921	12MAY62	KRAUSE	602
10	457232	15OCT63	LOVELL	602
11	127845	25DEC43	MEDER	.
12	423286	31OCT64	MIFUNE	602
13	677890	24APR65	NISHIMATSU-LYNCH	.
14	216382	24JUL63	PURINTON	602
15	459287	05JAN34	RODRIGUES	.
16	346917	15MAR50	SHIEKELESLAN	.
17	234967	21DEC67	SMITH	602
18	254896	06APR49	TAYLOR-HUNYADI	.
19	212916	29MAY28	WACHBERGER	602
20	119012	05JAN46	WOLF-PROVENZA	602

For more information about the UPDATE statement, see *SAS Language Reference: Dictionary*.

Note: You cannot update ADABAS data directly using the DATA step, but you can update ADABAS data using the following procedures: APPEND, FSEDIT, FSVIEW, and SQL. For more information about updating ADABAS data, see Chapter 4, “Browsing and Updating ADABAS Data,” on page 39. △

Performance Considerations

While you can generally treat view descriptors like other SAS data sets in SAS programs, here are a few things you should keep in mind:

- It is sometimes better to extract ADABAS data and place them in a SAS data file rather than to read them directly. Here are some circumstances when you should probably extract:
  - If you plan to use the same ADABAS data in several procedures during the same SAS session, you may improve performance by extracting the ADABAS data. Placing these data in a SAS data file requires a certain amount of disk space to store the data and I/O to write the data. However, SAS data files are organized to provide optimal performance with PROC and DATA steps. Programs using SAS data files often use less CPU time than programs that directly read ADABAS data.

  - If you plan to read large amounts of ADABAS data and the data are being shared by several users, your direct reading of the data could adversely affect all users' response time.
  - If you are the creator of an ADABAS file and think that directly reading this data would present a security risk, you may want to extract the data and not distribute information about either the access descriptor or view descriptor.
- If you intend to use the data in a particular sorted order several times, it is usually best to run the SORT procedure on the view descriptor, using the OUT= option. This is more efficient than requesting the same sort repeatedly (with a BY clause) on the ADABAS data. Note that you cannot run the SORT procedure on a view descriptor unless you use the SORT procedure's OUT= option.
- Sorting data can be resource-intensive, whether it is done with the SORT procedure, with a BY statement (which generates a BY clause), or with a SORT clause stored in the view descriptor. You should sort data only when it is needed for your program.
- If you reference a view descriptor in SAS code and the code includes a BY statement for a variable or variables (up to three) that corresponds to a descriptor data field in the ADABAS file, the interface view engine is called, and it will support the BY clause if possible. Thus, the BY clause sorts the ADABAS data before it uses the data in your SAS program. If the ADABAS file is very large, this sorting can affect performance.

  If the view descriptor already has a SORT clause and you specify a BY statement in your SAS code, the BY statement overrides the view descriptor's SORT clause.
- When writing a SAS program and referencing a view descriptor, it is more efficient to use a SAS WHERE statement in the program than it is to use a subsetting IF statement. The SAS program passes the WHERE statement as a WHERE clause to the interface view engine, which adds it (using the Boolean operator AND) to any WHERE clause stored in the view descriptor. The view descriptor is then passed to ADABAS for processing. Applying a WHERE clause to the ADABAS data may reduce the number of logical records read; therefore, it often improves performance.
- Refer to "Creating and Using View Descriptors Efficiently" on page 98 for more details on creating efficient view descriptors.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Interface to ADABAS Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Interface to ADABAS Software: Reference, Version 8

ISBN 1–58025–546–9

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999