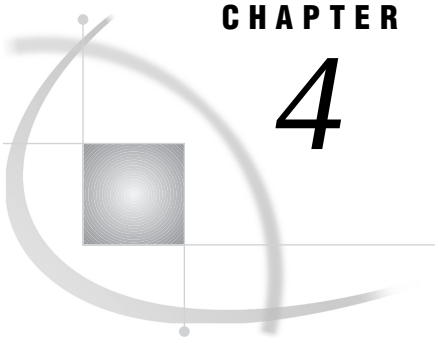


CHAPTER

4

Using CA-DATACOM/DB Data in SAS Programs

Introduction

An advantage of the SAS/ACCESS interface to CA-DATACOM/DB is that it enables the SAS System to read and write CA-DATACOM/DB data directly using SAS programs. This chapter presents examples using CA-DATACOM/DB data described by view descriptors in SAS programs. For information on the views and sample data, see Appendix 3, “Data and Descriptors for the Examples,” on page 125.

Throughout the examples, the SAS terms *column* and *row* are used instead of comparable CA-DATACOM/DB 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 Version 6 and Version 8 SAS data sets with data from CA-DATACOM/DB. For more information on 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 43 presents some techniques for using view descriptors efficiently in SAS programs.

Reviewing Columns

If you want to use CA-DATACOM/DB data described by a view descriptor in your SAS program but cannot remember the SAS column names or formats and informats, you can use the CONTENTS or DATASETS procedure to display this information.

The following example uses the DATASETS procedure to give you information on the view descriptor VLIB.CUSPHON, which is based on the CA-DATACOM/DB table CUSTOMERS.

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

Output 4.1 on page 24 shows the information for this example. The data described by VLIB.CUSPHON are shown in Output 4.9 on page 34.

Output 4.1 Using the DATASETS Procedure with a View Descriptor

```
                                The SAS System                                1
                              DATASETS PROCEDURE

Data Set Name: VLIB.CUSPHON                          Observations:          22
Member Type:   VIEW                                  Variables:             3
Engine:        SASIODDB                              Indexes:               0
Created:       11:19 Friday, October 12, 1990        Observation Length:    80
Last Modified: 12:03 Friday, October 12, 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 column labels using the DATASETS procedure. The labels are generated as the complete CA-DATACOM/DB field name when the view descriptor is created, and they cannot be overridden.
- The Created date is when the access descriptor for this view descriptor was created.
- The Last Modified date is the last time the view descriptor was updated or created.
- The Observations number shown is the number of records in the CA-DATACOM/DB table.

For more information on the DATASETS procedure, see the *SAS Language Reference: Dictionary* and the *SAS Procedures Guide*.

Printing Data

Printing CA-DATACOM/DB data described by a view descriptor is exactly like printing a SAS data file, as shown by the following example:

```
proc print data=vlib.empinfo;
   title2 'Brief Employee Information';
run;
```

VLIB.EMPINFO derives its data from the EMPLOYEES table. Output 4.2 on page 25 shows the first page of output for this example.

Output 4.2 Results of the PRINT Procedure

```
                Brief Employee Information                          1
 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 row 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 rows described by the view descriptor VLIB.CUSORDR:

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

VLIB.CUSORDR accesses data from the table ORDER. Output 4.3 on page 26 shows the result of this example.

Output 4.3 Results of Using the OBS= Option

```
First Five Data Records Described by VLIB.CUSORDR                1
            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, but the FIRSTOBS= option does not improve performance significantly because each record must still be read and its position calculated.

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

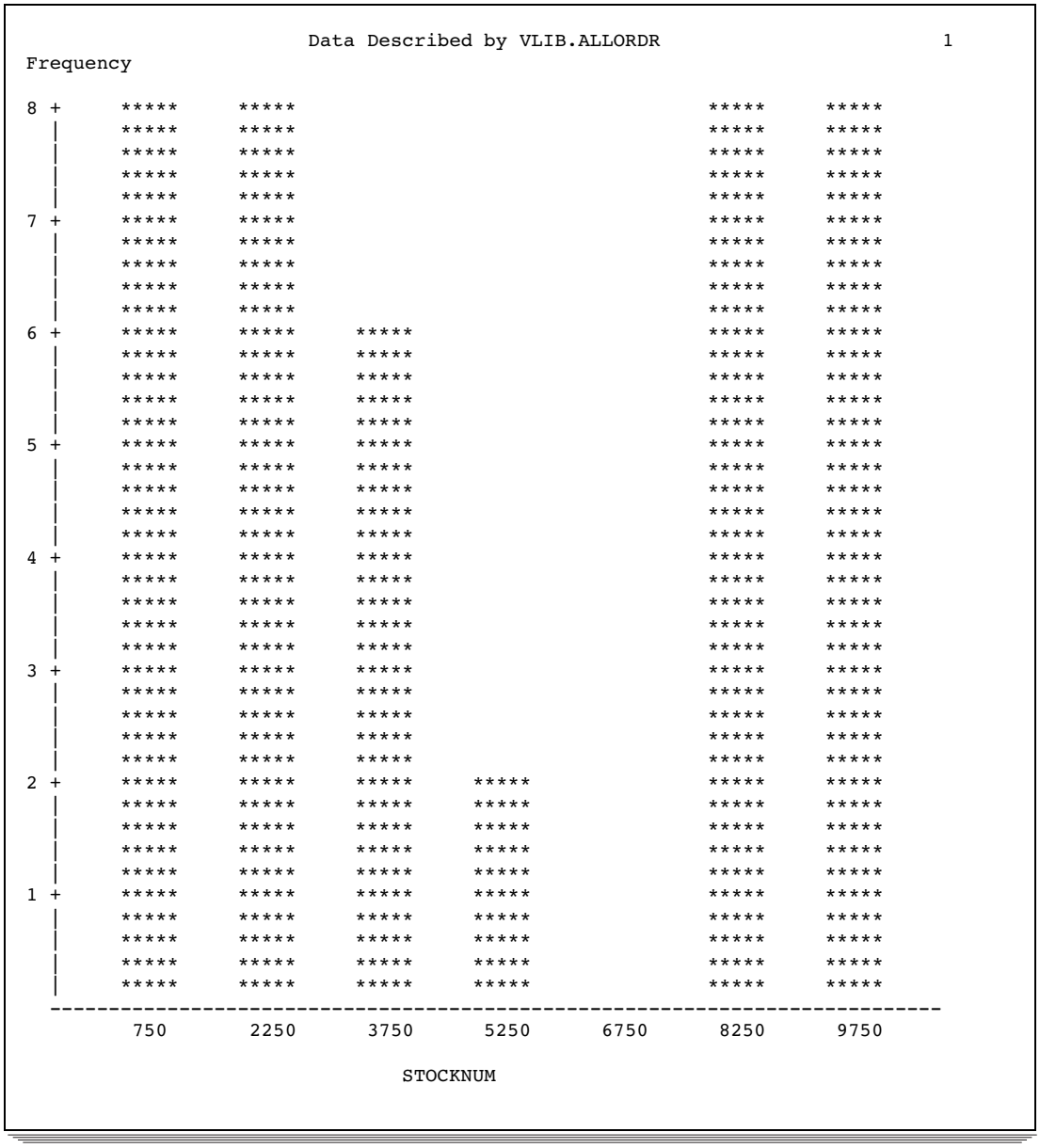
Charting Data

CHART procedure programs work with data described by view descriptors just as they do with SAS data files. The following example uses 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 table ORDER. Output 4.4 on page 27 shows the information for this example. STOCKNUM represents each product. The number of orders for each product is represented by the height of the bar.

Output 4.4 Vertical Bar Chart Showing Number of Orders per Product



For more information on 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 CA-DATACOM/DB data. See the *SAS/GRAPH Software: Reference* for more information on the kinds of graphics you can produce with this SAS software product.

Calculating Statistics

You can also use statistical procedures with CA-DATACOM/DB data. This section shows simple examples using the FREQ and MEANS procedures.

Using the FREQ Procedure

Suppose you want 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 percentage of invoices for each country appearing in the CA-DATACOM/DB table INVOICE using the view descriptor VLIB.INV:

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

Output 4.5 on page 28 shows the one-way frequency table this example generates.

Output 4.5 Frequency Table for Field COUNTRY described by View Descriptor VLIB.INV

```
                    Data Described by VLIB.INV                          1
                              COUNTRY

                                                 Cumulative  Cumulative
COUNTRY             Frequency    Percent   Frequency    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
```

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

Using the MEANS Procedure

Still analyzing recent orders, suppose you want to determine some statistics for each USA customer. The view descriptor VLIB.USAORDR accesses records from the ORDER table that have a SHIPTO value beginning with a 1, indicating a USA customer.

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 rows (N) and the number of missing values (NMISS).

```
proc means data=vlib.usaordr mean sum n nmiss maxdec=0;
   by shipto;
   var length fabricch;
```

```
   title 'Data Described by VLIB.USAORDR';
run;
```

The BY statement causes the interface view engine to generate ordering criteria so that the data are sorted. Output 4.6 on page 30 shows some of the information produced by this example.

Output 4.6 Statistics on Fabric Length and Charges for Each USA Customer

```
                          Data Described by VLIB.USAORDR                          1
--------------------------------- SHIPTO=14324742 ---------------------------------

         Variable  Label           N  Nmiss          Mean           Sum
         ------------------------------------------------------------
         LENGTH    LENGTH          4      0          1095          4380
         FABRICCH  FABRICCHARGES   2      2       1934460       3868920
         ------------------------------------------------------------

--------------------------------- SHIPTO=14898029 ---------------------------------

         Variable  Label           N  Nmiss          Mean           Sum
         ------------------------------------------------------------
         LENGTH    LENGTH          2      0          2500          5000
         FABRICCH  FABRICCHARGES   2      0       1400825       2801650
         ------------------------------------------------------------

--------------------------------- SHIPTO=15432147 ---------------------------------

         Variable  Label           N  Nmiss          Mean           Sum
         ------------------------------------------------------------
         LENGTH    LENGTH          4      0           725          2900
         FABRICCH  FABRICCHARGES   2      2        252149        504297
         ------------------------------------------------------------

--------------------------------- SHIPTO=18543489 ---------------------------------

         Variable  Label           N  Nmiss          Mean           Sum
         ------------------------------------------------------------
         LENGTH    LENGTH          6      0           303          1820
         FABRICCH  FABRICCHARGES   4      2      11063836      44255344
         ------------------------------------------------------------

--------------------------------- SHIPTO=19783482 ---------------------------------

         Variable  Label           N  Nmiss          Mean           Sum
         ------------------------------------------------------------
         LENGTH    LENGTH          4      0           450          1800
         FABRICCH  FABRICCHARGES   4      0        252149       1008594
         ------------------------------------------------------------

--------------------------------- SHIPTO=19876078 ---------------------------------

         Variable  Label           N  Nmiss          Mean           Sum
         ------------------------------------------------------------
         LENGTH    LENGTH          2      0           690          1380
         FABRICCH  FABRICCHARGES   0      2             .             .
         ------------------------------------------------------------
```

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

Using the RANK Procedure

You can also use more advanced statistics procedures with CA-DATACOM/DB data. The following example uses the RANK procedure with data described by the view descriptor VLIB.EMPS 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. It assigns the column name DATERANK to the new field created by the procedure. (The VLIB.EMPS view descriptor includes a WHERE clause to select only the employees whose job code is 602.)

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

VLIB.EMPS is based on the CA-DATACOM/DB table EMPLOYEES. Output 4.7 on page 31 shows the result of this example.

Output 4.7 Ranking of Employee Birthdays

```
                        Order of Employee Birthdays                          1
 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 on the RANK procedure and other advanced statistics procedures, see the *SAS Procedures Guide*.

Selecting and Combining Data

Many SAS programs select and combine data from various sources. The method you use depends on the configuration of the data. The next examples show you how to select

and combine data using two different methods. When choosing between these methods, consider the issues described in “Performance Considerations” on page 43.

Using the WHERE Statement

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

```
data notpaid(keep=invoicen billedto amtbille billedon);
   set vlib.usainv vlib.forinv;
   where paidon is missing and amtbille>=300000.00;
run;

proc print;
   title 'High Bills--Not Paid';
run;
```

In the SAS WHERE statement, be sure to use the SAS column names, not the CA-DATACOM/DB field names. Both VLIB.USAINV and VLIB.FORINV are based on the CA-DATACOM/DB table INVOICE. Output 4.8 on page 32 shows the result of the new temporary data file, WORK.NOTPAID.

Output 4.8 NOTPAID Data File Created with a SAS WHERE Statement

```
                    High Bills--Not Paid                              1
       OBS    INVOICEN    BILLEDTO          AMTBILLE     BILLEDON

         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 data set option works with SAS/ACCESS views just as it works with other SAS data sets. That is, the KEEP= option specifies that you want only the listed columns included in the new data file, NOTPAID, although you can use the other columns within the DATA step.

Notice that the WHERE statement includes two conditions to be met. First, it selects only rows that have a missing value for the field PAIDON. As you can see, it is important to know how the CA-DATACOM/DB data are configured before you use these data in a SAS program. The field PAIDON contains values that translate to missing values in the SAS System. (Also, each of the two view descriptors has its own WHERE clause.)

Second, the WHERE statement requires that the amount in each bill be higher than a certain figure. Again, you should be familiar with the CA-DATACOM/DB 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 WHERE statement than 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 a Boolean AND) to any WHERE clause defined in the view descriptor's selection criteria. The selection criteria are then passed to CA-DATACOM/DB for processing. Processing CA-DATACOM/DB data using a WHERE clause may reduce the number of records read from the database and therefore often improves performance.

For more information on the SAS WHERE statement, refer to the *SAS Language Reference: Dictionary*.

Using the SAS System SQL Procedure

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

Combining Data from Various Sources

The SQL procedure provides another way to select and combine data from one or more database products. For example, suppose you have view descriptors VLIB.CUSPHON and VLIB.CUSORDR based on the CA-DATACOM/DB tables CUSTOMERS and ORDER, respectively, and a SAS data file, MYDATA.OUTOFSTK, which contains product names and numbers that are out of stock. You can use the SQL procedure to join all these sources of data to form a single output file. A WHERE statement or a subsetting IF statement would not be appropriate in this case because you want to compare column values from several sources rather than simply merge or concatenate the data.

Output 4.9 on page 34, Output 4.10 on page 35, and Output 4.11 on page 36 on the following pages show the results of the PRINT procedure performed on the data described by the VLIB.CUSPHON and VLIB.CUSORDR view descriptors and on the MYDATA.OUTOFSTK 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 4.9 Data Described by the View Descriptor VLIB.CUSPHON

Data Described by VLIB.CUSPHON 1

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	DURHAM SCIENTIFIC SUPPLY COMPANY
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 4.10 Data Described by the View Descriptor VLIB.CUSORDR

```
                Data Described by VLIB.CUSORDR                        1
                 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 4.11 Data in the SAS Data File Data File MYDATA.OUTOFSTK

```
SAS Data File MYDATA.OUTOFSTK                          1

 OBS    FIBERNAM    FIBERNUM


  1      olefin       3478
  2      gold         8934
  3      dacron       4789
```

The following SAS code selects and combines data from these three sources (the two view descriptors and the SAS data file) to create a view, SQL.BADORDRS*. This view retrieves customer and product information so that the sales department can notify customers of products no longer available.

```
proc sql;
create view sql.badordrs 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.BADORDRS';
   select * from sql.badordrs;
```

The CREATE VIEW statement incorporates a WHERE clause as part of the SELECT statement, but it is not the same as the SAS WHERE statement illustrated earlier in this chapter. The last SELECT statement retrieves and displays the PROC SQL view, SQL.BADORDRS. To select all fields from the view, an asterisk (*) is used in place of field names. The fields are displayed in the same order as they were specified in the first SELECT clause.

Output 4.12 on page 37 shows the data described by the SQL.BADORDRS view. Note that the SQL procedure uses the DBMS labels in the output by default.

* You may want to store your PROC SQL views in a SAS data library other than the one storing your view descriptors, because they both have member type view.

Output 4.12 Data Described by the PROC SQL View SQL.BADORDRS

```
                    Data Described by SQL.BADORDRS                        1

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.BADORDRS 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 SET statement and the special variable FIRST.PRODUCT. This variable identifies the first row in a particular BY group. You need a customer's name associated only once to notify that customer 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.badordrs;
   by custnum product;
   if first.product;
run;

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

The data file MYDATA.BADNEWS contains a row for each unique combination of customer and out-of-stock product. Output 4.13 on page 38 displays this data file.

Output 4.13 Data in the SAS Data File MYDATA.BADNEWS

```
                              MYDATA.BADNEWS Data File                      1
 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 on the special variable FIRST, see "BY Statement" in the *SAS Language Reference: Dictionary*.

Creating New Fields with the PROC SQL GROUP BY Clause

It is often useful to create new fields with summary or aggregate functions, such as AVG or SUM. Although you cannot use the ACCESS procedure to create new fields, you can easily use the SQL procedure with data described by a view descriptor to display output containing new fields.

This example uses the SQL procedure to retrieve and manipulate data from the view descriptor VLIB.ALLEMP, which is based on the CA-DATACOM/DB table EMPLOYEES. 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 columns displayed matches the order of the columns specified in the SELECT list of the query. Output 4.14 on page 39 shows the query's result.

Output 4.14 Data Retrieved by a PROC SQL Query

```
                    Average Salary Per Department                    1

                                  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 on the SQL procedure, refer to the *SAS Procedures Guide*.

Updating a SAS Data File with CA-DATACOM/DB Data

You can update a SAS data file with CA-DATACOM/DB data described by a view descriptor the same way you update a SAS data file with data from another data file: by using a DATA step UPDATE statement. In this section, the term *transaction data* refers to the new data that are to be added to the original file. Because the SAS/ACCESS interface to CA-DATACOM/DB uses the Version 6 compatibility engine, the transaction data are from a Version 6 source. The original file can be a Version 6 data file or a Version 8 data file.

Updating a Version 6 Data File

You can update a Version 6 SAS data file with CA-DATACOM/DB data the same way you did in Version 6 of the SAS System. 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 CA-DATACOM/DB table EMPLOYEES. To perform the update, enter the following SAS code:

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

proc print data=lib6.birthday;
   format birthdat date7.;
   title 'LIB6.BIRTHDAY Data File';
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 updated SAS data file, MYDATA.NEWBDAY, is a Version 6 data file. It is stored in the Version 6 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 the interface view engine to automatically generate a SORT clause for the column LASTNAME. Thus, the SORT clause causes the CA-DATACOM/DB data to be presented to the SAS System in a sorted order so they can be used to update the MYDATA.NEWBDAY data file. The data file LIB6.BIRTHDAY had to be sorted (by the SAS SORT procedure) before the update, because the UPDATE statement expects the data to be sorted by the BY column.

Output 4.15 on page 41, Output 4.16 on page 41, and Output 4.17 on page 42 show the results of the PRINT procedure on the original data file, the transaction data, and the updated data file.

Output 4.15 Data File To Be Updated, LIB6.BIRTHDAY

LIB6.BIRTHDAY Data File 1

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 4.16 Data Described by VLIB.EMPS

Data Described by VLIB.EMPS 1

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 4.17 Updated Data File, MYDATA. NEWBDAY

MYDATA.NEWBDAY Data File 1

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

Updating a Version 8 Data File

Versions 6 and 8 of the SAS System support different naming conventions, therefore, there could be character-length discrepancies between the columns in the original data file and the transaction data. You have two choices when updating a Version 8 data file:

- let the compatibility engine truncate names exceeding 8 characters. The truncated names will be added to the updated data file as new columns.
- rename the columns in the Version 8 data file to match the columns in the descriptor file.

The following example resolves character-length discrepancies by using the RENAME DATA step option with the UPDATE statement. A Version 8 data file, LIB8.BIRTHDAYS, is updated with data described by VLIB.EMPS.

```
proc sort data=lib8.birthdays;
  by last_name;
run;

proc print data=lib8.birthdays;
    format birthdate date7.;
    title 'LIB8.BIRTHDAYS Data File';
run;
```

```
data newdata.v8_birthdays;
   update lib8.birthday
   (rename= (last_name=lastname
             firstname=firstnme
             birthdate=birthdat)) vlib.emps;
by lastname firstnme;
run;

proc print data=newdata.v8_birthdays;
   title 'NEWDATA.V8_BIRTHDAYS Data File';
run;
```

In this example, the updated data file NEWDATA.V8_BIRTHDAYS is a Version 8 data file that is stored in a Version 8 data library associated with the libref NEWDATA. Version 8 supports member and column names of up to 32 characters. However, the RENAME= DATA step option is used with the UPDATE statement to change the longer column names in LIB8.BIRTHDAYS to match the 8–character column names in VLIB.EMPS. The columns are renamed *before* the updated data file is created.

Output 4.18 on page 43 shows the results of the PRINT procedure on the original data file. The updated file looks like Output 4.17 on page 42.

Output 4.18 Data File to be Updated, LIB8.BIRTHDAYS

LIB8.BIRTHDAYS Data File 1

OBS	EMPLOYEE_ID	BIRTHDATE	LAST_NAME
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

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

You cannot update a CA-DATACOM/DB table directly using the DATA step, but you can update a CA-DATACOM/DB table using the following procedures: APPEND, FSEDIT, FSVIEW, SQL, and SAS/AF applications. See Chapter 5, "Browsing and Updating CA-DATACOM/DB Data," on page 45 for more information on updating CA-DATACOM/DB data.

Performance Considerations

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

- It is sometimes better to extract CA-DATACOM/DB data and place them in a SAS data file than to read them directly. Here are some circumstances when you should probably extract:
  - If you plan to use the same CA-DATACOM/DB data in several procedures in the same session, you may improve performance by extracting the CA-DATACOM/DB 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 when they read CA-DATACOM/DB data directly.
  - If you plan to read large amounts of data from a CA-DATACOM/DB table and the data are being shared by several users (Multi-User environment), your direct reading of the data could adversely affect all users' response times.
  - If you are the creator of a table, and you 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 more efficient to run the SORT procedure on the view descriptor, using the OUT= option than to request the same sort repeatedly (with a SORT clause) on the CA-DATACOM/DB 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, or with a SORT clause included in the view descriptor. When you use a SAS BY statement with a view descriptor, it is most efficient to use a BY column that is associated with an indexed CA-DATACOM/DB field. Also, if you do not need a certain order, blank out the Default Key. Otherwise, you may cause an unnecessary sort.
- If you use a Default Key, the interface view engine will use an index read instead of a sort if it can. Index reads are faster, but not always possible. For example, an index read is not possible if you specify multiple sort keys, multiple WHERE clause conditions, or a WHERE clause condition with a column that is not a key.
- When you are writing a SAS program and referencing a view descriptor, it is more efficient to use a WHERE statement in the program than it is to use a subsetting IF statement. The interface view engine passes the WHERE statement as CA-DATACOM/DB selection criteria to the view descriptor, connecting it (with the AND operator) to any WHERE clause included in the view descriptor. Applying a WHERE clause to the CA-DATACOM/DB data may reduce the number of records processed, which often improves performance.
- You can provide your own URT with options that are fine-tuned for your applications.
- 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 CA-DATACOM/DB Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 170.

SAS/ACCESS Interface to CA-DATACOM/DB Software: Reference, Version 8

ISBN 1–58025–545–0

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

1st printing, October 1999