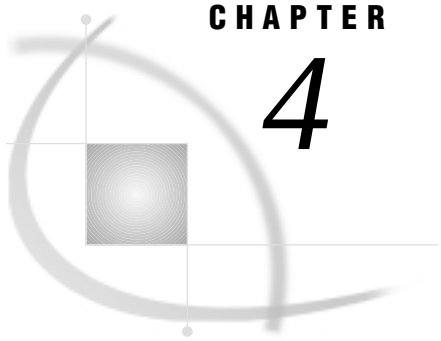


CHAPTER

4

Using SYSTEM 2000 Data in SAS Programs

Introduction

The advantage of the SAS/ACCESS interface to SYSTEM 2000 software is that it enables the SAS System to read and write SYSTEM 2000 data directly using SAS programs. This chapter presents examples of using SYSTEM 2000 data described by view descriptors in SAS programs. For information on the example data used in this chapter, see Appendix 3, "Example Data," on page 143.

Throughout the examples, the SAS terms *variable* and *observation* are used instead of comparable SYSTEM 2000 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 7 SAS data set with data from SYSTEM 2000 software. 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 40 presents some techniques for using view descriptors efficiently in SAS programs.

Reviewing Variables

If you want to use SYSTEM 2000 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.

This example uses the DATASETS procedure to give you information on the view descriptor VLIB.EMPPOS, which you created in Chapter 3, "Defining SAS/ACCESS Descriptor Files," on page 19.

```
proc datasets library=vlib memtype=view;
   contents data=emppos(s2kpw=demo);
run;
```

Output 4.1 on page 24 shows the information for this example.

Output 4.1 Using the DATASETS Procedure with a View Descriptor

```
                              The SAS System                                1

                           DATASETS PROCEDURE

     Data Set Name: VLIB.EMPPOS               Observations:         887
     Member Type:   VIEW                      Variables:            5
     Engine:        SASIOS2K                  Indexes:              0
     Created:       03NOV89:16:17:59          Observation Length:   53
     Last Modified: 07SEP89:14:15:58          Deleted Observations: 0
     Data Set Type:                           Compressed:           NO
     Label:

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

#    Variable    Type    Len    Pos    Format    Informat    Label
-------------------------------------------------------------------------------
4    DEPARTME    Char     14     36    $14.      $14.        DEPARTMENT
2    FIRSTNME    Char     10     10    $10.      $10.        FORENAME
1    LASTNAME    Char     10      0    $10.      $10.        LAST NAME
5    MANAGER     Char      3     50    $3.       $3.         MANAGER
3    POSITION    Char     16     20    $16.      $16.        POSITION TITLE
```

Note the following points about Output 4.1 on page 24:

- You cannot change a view descriptor's variable labels using the DATASETS procedure. The labels are generated to be the complete SYSTEM 2000 item 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 SYSTEM 2000 database was updated.
- The Observations number shown is the highest number of schema records that has ever occurred in the database. This number of observations does not correspond to the number of observations that the view descriptor accesses.

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

Printing Data

Printing SYSTEM 2000 data described by a view descriptor is exactly like printing a SAS data file, as shown in the following example:

```
proc print data=vlib.emppos(s2kpw=demo);
   title2 'Subset of EMPLOYEE Database
       Information';
run;
```

Output 4.2 on page 25 shows the first page of output for this example.

Output 4.2 Results of PROC PRINT

Subset of EMPLOYEE Database Information 1

OBS	LASTNAME	FIRSTNME	POSITION	DEPARTME	MANAGER
1			PROGRAMMER	INFORMATION SY	MYJ
2	AMEER	DAVID	SR SALES REPRESE	MARKETING	VPB
3	AMEER	DAVID	JR SALES REPRESE	MARKETING	VPB
4	BOWMAN	HUGH E.	EXECUTIVE VICE-P	CORPORATION	CPW
5	BROOKS	RUBEN R.	JR SALES REPRESE	MARKETING	MAS
6	BROWN	VIRGINA P.	MANAGER WESTERN	MARKETING	OMG
7	CAHILL	JACOB	MANAGER SYSTEMS	INFORMATION SY	JBM
8	CANADY	FRANK A.	MANAGER PERSONNE	ADMINISTRATION	PRK
9	CHAN	TAI	SR SALES REPRESE	MARKETING	TZR
10	COLLINS	LILLIAN	MAIL CLERK	ADMINISTRATION	SQT
11	FAULKNER	CARRIE ANN	SECRETARY	CORPORATION	JBM
12	FERNANDEZ	SOPHIA	STANDARDS & PROC	INFORMATION SY	JLH
13	FREEMAN	LEOPOLD	SR SYSTEMS PROGR	INFORMATION SY	JLH

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 rows described by the view descriptor VLIB.EMPPOS:

```
proc print data=vlib.emppos(s2kpw=demo obs=5);
   title2 'First Five Data Rows Described by
      VLIB.EMPPOS';
run;
```

Output 4.3 on page 26 shows the result of this example.

Output 4.3 Results of Using the OBS= Option

First Five Data Rows Described by VLIB.EMPPOS 1

OBS	LASTNAME	FIRSTNME	POSITION	DEPARTME	MANAGER
1			PROGRAMMER	INFORMATION SY	MYJ
2	AMEER	DAVID	SR SALES REPRESE	MARKETING	VPB
3	AMEER	DAVID	JR SALES REPRESE	MARKETING	VPB
4	BOWMAN	HUGH E.	EXECUTIVE VICE-P	CORPORATION	CPW
5	BROOKS	RUBEN R.	JR SALES REPRESE	MARKETING	MAS

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 *SAS Language Reference: Dictionary*.

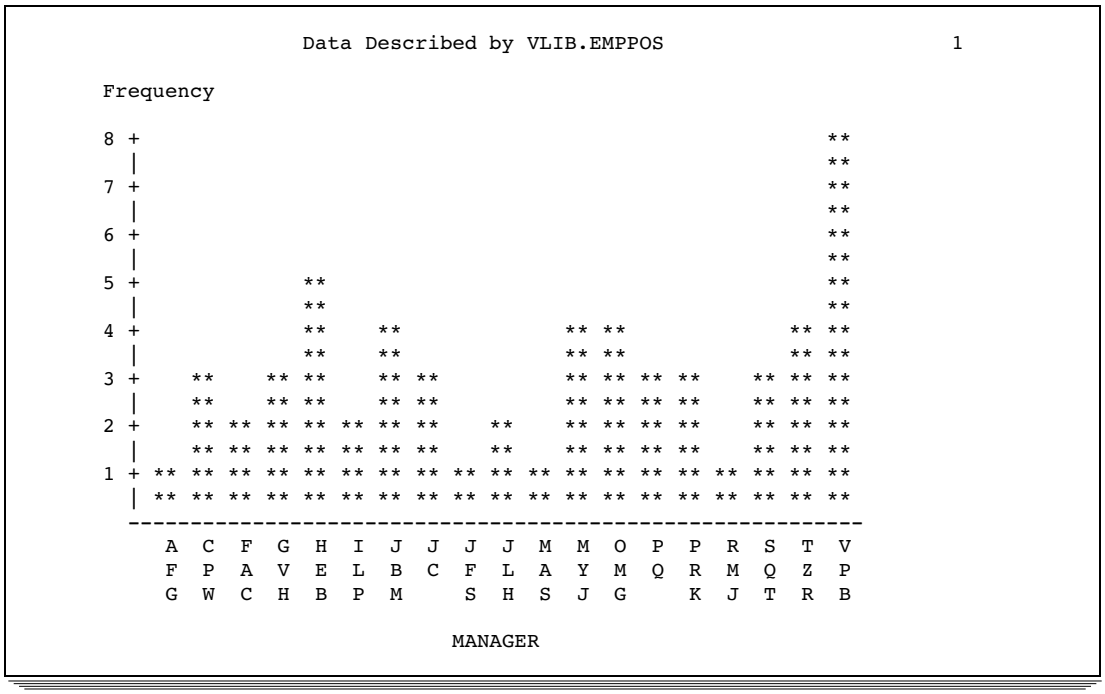
Charting Data

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.EMPPOS to create a vertical bar chart of the number of employees each manager has had:

```
proc chart data=vlib.emppos(s2kpw=demo);
   vbar manager;
   title2 'Data Described by VLIB.EMPPOS';
run;
```

Output 4.4 on page 27 shows the information for this example. The number of employees for each manager is represented by the height of the bar.

Output 4.4 Vertical Bar Chart Showing Number of Employees per Manager



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 SYSTEM 2000 data. See *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 SYSTEM 2000 data. The following sections show simple examples using the FREQ and MEANS procedures.

Using the FREQ Procedure

Suppose you want to find what percentages of your employees have specific degrees. The following example calculates the percentage of employees for each degree appearing in the EMPLOYEE database using the view descriptor VLIB.EMPEDUC:

```
proc freq data=vlib.empeduc;
   tables degree;
   title2 'Data Described by VLIB.EMPEDUC';
run;
```

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

Output 4.5 Frequency Table for Item DEGREE Described by View Descriptor VLIB.EMPEDUC

Data Described by VLIB.EMPEDUC 1

DEGREE/CERTIFICATE

DEGREE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AA	5	7.9	5	7.9
BA	12	19.0	17	27.0
BS	23	36.5	40	63.5
HIGH SC	6	9.5	46	73.0
MA	3	4.8	49	77.8
MBA	1	1.6	50	79.4
MS	9	14.3	59	93.7
PHD	4	6.3	63	100.0

Frequency Missing = 12

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

Using the MEANS Procedure

In your further analysis of employee education, suppose you also want to develop some statistics on employees' skill types and their years of experience. The view descriptor VLIB.EMPSKIL accesses values from the EMPLOYEE database for skill type and years of experience.

The following example generates the mean and sum of the years of experience by skill type. Also included are the number of observations (N) and the number of missing values (NMISS).

```
proc means data=vlib.empskil mean sum n nmiss
 maxdec=0;
   by skilltyp;
   var years;
   title2 'Data Described by VLIB.EMPSKIL';
run;
```

Note that the BY statement causes the interface view engine to generate a SYSTEM 2000 ordering-clause so that the data is sorted by skill type. Output 4.6 on page 29 shows some of the information produced by this example.

Output 4.6 Statistics on Skill Type and Years of Experience

```
                    Data Described by VLIB.EMPSKIL                        1

                  Analysis Variable : YEARS YEARS OF EXPERIENCE

-------------------------- SKILL TYPE=  ----------------------------------

                    N  Nmiss          Mean           Sum
                   ---------------------------------------
                    0      6             .             .
                   ---------------------------------------

-------------------------- SKILL TYPE=ACCOUNTING -------------------------

                    N  Nmiss          Mean           Sum
                   ---------------------------------------
                    6      0             8            47
                   ---------------------------------------

-------------------------- SKILL TYPE=ASSEMBLER -------------------------

                    N  Nmiss          Mean           Sum
                   ---------------------------------------
                   14      0            10           141
                   ---------------------------------------

-------------------------- SKILL TYPE=CARTOON ART ------------------------

                    N  Nmiss          Mean           Sum
                   ---------------------------------------
                    1      0             1             1
                   ---------------------------------------

-------------------------- SKILL TYPE=CHINESE ----------------------------

                    N  Nmiss          Mean           Sum
                   ---------------------------------------
                    1      0             8             8
                   ---------------------------------------

-------------------------- SKILL TYPE=COBOL ------------------------------

                    N  Nmiss          Mean           Sum
                   ---------------------------------------
                   12      0             7            88
                   ---------------------------------------
```

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 SYSTEM 2000 data. The following example uses the RANK procedure with data described by the view descriptor VLIB.EMPBD to calculate the order of birthdays for a set of employees and to assign the variable name DATERNK to the new item created by the procedure. (The VLIB.EMPBD view descriptor includes a SYSTEM 2000 where-clause to select only the employees in the Marketing Department.)

```
proc rank data=vlib.empbd out=mydata.rankexm;
   var birthday;
   ranks daternk;
run;

proc print data=mydata.rankexm;
   title2 'Order of Marketing Employee Birthdays';
run;
```

Output 4.7 on page 30 shows a portion of the result of this example.

Output 4.7 Ranking of Employee Birthdays

Order of Marketing Employee Birthdays 1

OBS	LASTNAME	FIRSTNME	BIRTHDAY	DATERNK
1	AMEER	DAVID	10OCT51	14.0
2	BROOKS	RUBEN R.	25FEB52	15.0
3	BROWN	VIRGINA P.	24MAY46	9.0
4	CHAN	TAI	04JUL46	10.0
5	GARRETT	OLAN M.	23JAN35	2.0
6	GIBSON	GEORGE J.	23APR46	8.0
7	GOODSON	ALAN F.	21JUN50	13.0
8	JUAREZ	ARMANDO	28MAY47	11.0
9	LITTLEJOHN	FANNIE	17MAY54	17.0
10	RICHARDSON	TRAVIS Z.	30NOV37	4.0
11	RODRIGUEZ	ROMUALDO R	09FEB29	1.0
12	SCHOLL	MADISON A.	19MAR45	7.0
13	SHROPSHIRE	LELAND G.	04SEP49	12.0
14	SMITH	JERRY LEE	13SEP42	5.5
15	VAN HOTTEN	GWENDOLYN	13SEP42	5.5
16	WAGGONNER	MERRILEE D	27APR36	3.0
17	WILLIAMSON	JANICE L.	19MAY52	16.0

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

Selecting and Combining Data with the SQL Procedure

The next three examples show you how to select and combine data using the SAS System SQL procedure.

Using the WHERE Clause

Suppose you have two view descriptors, VLIB.EMPPOS and VLIB.EMPEDUC, that access employee positions and employee education, respectively. You could use the SQL procedure to combine these files into a single SAS data file. The WHERE clause specifies that you want a data file containing information on employees who do not have a degree (that is, the value is missing) and who are in the CORPORATION department.

Note: The SQL procedure displays the variable labels as stored in the view. However, because you are referencing a view descriptor, you must use the SAS variable names in the WHERE clause, not the SYSTEM 2000 item names. △

```
proc sql;
  title 'Corporation Positions With No Degrees';
  select emppos.lastname, position, degree,
     departme
     from vlib.emppos, vlib.empeduc
     where emppos.lastname=empeduc.lastname and
           empeduc.degree is missing and
           emppos.departme='CORPORATION'
     order by lastname;
```

Output 4.8 on page 31 shows the result of this example. (Notice that Waterhouse appears twice in the output. This is because he has two values for schema item C411 SCHOOL, but neither value has an associated value for C412 DEGREE/CERTIFICATE.)

Output 4.8 SQL Procedure Output Using a WHERE Clause

```
                Corporation Positions With No Degrees                  1

LAST NAME   POSITION TITLE    DEGREE/CERTIFICATE  DEPARTMENT
-------------------------------------------------------------------
FAULKNER    SECRETARY                           CORPORATION
KNIGHT      SECRETARY                           CORPORATION
WATERHOUSE  PRESIDENT                           CORPORATION
WATERHOUSE  PRESIDENT                           CORPORATION
```

Combining Data from Various Sources

Suppose along with view descriptors VLIB.EMPPOS and VLIB.EMPEDUC, you have a SAS data file, MYDATA.CLASSES, that contains in-house continuing education classes taken by employees. You can use the SQL procedure to join these sources of data to form a single output table of employees, their departments, their degrees, and the in-house classes they have taken. For example:

```
proc print data=vlib.emppos;
   title2 'Data Described by VLIB.EMPPOS';
```

```
run;

proc print data=vlib.empeduc;
   title2 'Data Described by VLIB.EMPEDUC';
run;

proc print data=mydata.classes;
   title2 'SAS Data File MYDATA.CLASSES';
run;
```

Note: If you have many PROC SQL views as well as view descriptors, you may want to store your PROC SQL views in a separate SAS data library from your view descriptors. They both have a member type of VIEW, so you cannot tell a view descriptor from a PROC SQL view. △

Output 4.9 on page 32, Output 4.10 on page 33, and Output 4.11 on page 33 show the results of the PRINT procedure performed on the data described by VLIB.EMPPOS, VLIB.EMPEDUC, and MYDATA.CLASSES.

Output 4.9 Data Described by the View Descriptor VLIB.EMPPOS

Data Described by VLIB.EMPPOS 1

OBS	LASTNAME	FIRSTNME	POSITION	DEPARTME	MANAGER
1			PROGRAMMER	INFORMATION SY	MYJ
2	AMEER	DAVID	SR SALES REPRESE	MARKETING	VPB
3	AMEER	DAVID	JR SALES REPRESE	MARKETING	VPB
4	BOWMAN	HUGH E.	EXECUTIVE VICE-P	CORPORATION	CPW
5	BROOKS	RUBEN R.	JR SALES REPRESE	MARKETING	MAS
6	BROWN	VIRGINA P.	MANAGER WESTERN	MARKETING	OMG
7	CAHILL	JACOB	MANAGER SYSTEMS	INFORMATION SY	JBM
8	CANADY	FRANK A.	MANAGER PERSONNE	ADMINISTRATION	PRK
9	CHAN	TAI	SR SALES REPRESE	MARKETING	TZR
10	COLLINS	LILLIAN	MAIL CLERK	ADMINISTRATION	SQT
11	FAULKNER	CARRIE ANN	SECRETARY	CORPORATION	JBM
12	FERNANDEZ	SOPHIA	STANDARDS & PROC	INFORMATION SY	JLH
13	FREEMAN	LEOPOLD	SR SYSTEMS PROGR	INFORMATION SY	JLH

Output 4.10 Data Described by the View Descriptor VLIB.EMPEDUC

Data Described by VLIB.EMPEDUC 1

OBS	LASTNAME	FIRSTNME	SEX	DEGREE
1				
2	AMEER	DAVID	MALE	BS
3	BOWMAN	HUGH E.	MALE	MS
4	BOWMAN	HUGH E.	MALE	BS
5	BOWMAN	HUGH E.	MALE	PHD
6	BROOKS	RUBEN R.	MALE	BS
7	BROWN	VIRGINA P	FEMALE	BA
8	CAHILL	JACOB	MALE	BS
9	CAHILL	JACOB	MALE	BS
10	CANADY	FRANK A.	MALE	MA
11	CANADY	FRANK A.	MALE	BS
12	CHAN	TAI	MALE	PHD
13	CHAN	TAI	MALE	BA

Output 4.11 SAS Data File MYDATA.CLASSES

SAS Data File MYDATA.CLASSES 1

OBS	LASTNAME	FIRSTNME	CLASS
1	AMEER	DAVID	PRESENTING IDEAS
2	CANADY	FRANK A.	PRESENTING IDEAS
3	GIBSON	MOLLY I.	SUPERVISOR SKILLS
4	GIBSON	MOLLY I.	STRESS MGMT
5	RICHARDSON	TRAVIS Z.	SUPERVISOR SKILLS

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.EDUC. This view retrieves employee names, their departments, their degrees, and the in-house classes they've taken.

```
proc sql;
 create view sql.educ as
   select emppos.lastname, emppos.firstnme,
     emppos.departme,empeduc.degree,
     classes.class as course
     from vlib.emppos
          vlib.empeduc,
          mydata.classes
```

```
  where (emppos.lastname=empeduc.lastname
         and emppos.firstnme=empeduc.firstnme)
         and
        (empeduc.lastname=classes.lastname
         and empeduc.firstnme=classes.firstnme)
  order by emppos.lastname, course;

title 'Data Described by SQL.EDUC';
select * from sql.educ;
```

The CREATE VIEW statement incorporates a WHERE clause as part of the SELECT statement. The last SELECT statement retrieves and displays the PROC SQL view, SQL.EDUC. To select all items from the view, an asterisk (*) is used in place of item names. The order of the items displayed matches the order of the items as specified in the first SELECT clause.

Output 4.12 on page 34 shows the data described by the SQL.EDUC view. Note that the SQL procedure uses the variable labels in the output by default.

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

```
                Data Described by SQL.EDUC                          1

LAST NAME   FORENAME      DEPARTMENT      DEGREE/CERTIFICATE
COURSE
-------------------------------------------------------------
AMEER       DAVID         MARKETING       BS
PRESENTING IDEAS

AMEER       DAVID         MARKETING       BS
PRESENTING IDEAS

CANADY      FRANK A.      ADMINISTRATION  MA
PRESENTING IDEAS

CANADY      FRANK A.      ADMINISTRATION  BS
PRESENTING IDEAS

GIBSON      MOLLY I.      INFORMATION SY  BA
STRESS MGMT

GIBSON      MOLLY I.      INFORMATION SY  BA
SUPERVISOR SKILLS

RICHARDSON  TRAVIS Z.     MARKETING       BS
SUPERVISOR SKILLS
```

The view SQL.EDUC lists entries for employees, their departments, and their degrees that have taken in-house classes. However, it contains duplicate observations because some employees have more than one degree and have taken more than one in-house class. To make the data more readable, you can create a final SAS data file, MYDATA.UPDATE, using the SET statement and the special variable FIRST. This variable identifies which observation is the first in a particular BY group. You only need an employee's name associated once with his or her degrees and in-house education classes, regardless of the number of degrees or the number of classes taken.

```
data mydata.update;
   set sql.educ;
   by lastname course;
   if first.lastname then output;
run;

proc print;
   title2 'MYDATA.UPDATE Data File';
run;
```

The data file MYDATA.UPDATE contains an observation for each unique combination of employee, degree, and in-house class. Output 4.13 on page 35 displays this data file.

Output 4.13 SAS Data File MYDATA.UPDATE

```
                          MYDATA.UPDATE Data File                        1

OBS    LASTNAME      FIRSTNME     DEPARTME         DEGREE    COURSE

  1    AMEER         DAVID        MARKETING        BS        PRESENTING IDEAS
  2    CANADY        FRANK A.     ADMINISTRATION   MA        PRESENTING IDEAS
  3    GIBSON        MOLLY I.     INFORMATION SY   BA        STRESS MGMT
  4    RICHARDSON    TRAVIS Z.    MARKETING        BS        SUPERVISOR SKILLS
```

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

Creating a New Item with the PROC SQL GROUP BY Clause

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

This example uses the SQL procedure to retrieve and manipulate data from the view descriptor VLIB.EMPVAC. When this query (as a SELECT statement is often called) is submitted, it calculates and displays the average vacation time (in hours) for each department.

```
proc sql;
   title 'Average Vacation Per Department';
   select distinct departme,
         avg(accruedv) label='Avg Vac'
      from vlib.empvac
      where departme is not missing
      group by departme;
```

The order of the items displayed matches the order of the items as specified in the SELECT clause of the query. Output 4.14 on page 36 shows the SELECT statement's result.

Output 4.14 Data Retrieved by a PROC SQL Query

```
          Average Vacation Per Department

               DEPARTMENT       Avg
             ------------------------
             ADMINISTRATION        43
             CORPORATION     40.72727
             INFORMATION SY     61.75
             MARKETING       47.61905
```

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

Updating a SAS Data File with SYSTEM 2000 Data

You can update a SAS data file with SYSTEM 2000 data described by a view descriptor just as you can update a SAS data file using 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 SYSTEM 2000 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 7 data file. See the following sections for an example.

Updating a Version 6 Data File

You can update a Version 6 SAS data file with SYSTEM 2000 data described by a view descriptor the same as you did with Version 6 of the SAS System. Suppose you have a Version 6 data file, V6.BIRTHDY, that contains Marketing employee names and birthdays. The file is out-of-date, and you want to update it with data described by VLIB.EMPBD. To perform the update, enter the following SAS code:

```
proc sort data=v6.birthdy;
   by lastname;
run;

data mydata.newbday;
   update v6.birthdy vlib.empbd;
   by lastname firstnme;
run;
```

In this example, the updated SAS data file, MYDATA.NEWBDAY, is a Version 6 data file. When the UPDATE statement references the view descriptor VLIB.EMPBD and uses a BY statement in the DATA step, the BY statement causes the interface view engine to automatically generate a SYSTEM 2000 ordering-clause for the variable LASTNAME. Thus, the ordering-clause causes the SYSTEM 2000 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 V6.BIRTHDY had to be sorted before the

update, because the UPDATE statement expects the data to be sorted by the BY variable.

Output 4.15 on page 37, Output 4.16 on page 37, and Output 4.17 on page 38 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, V6.BIRTHDY

V6.BIRTHDY Data File 1

OBS	LASTNAME	FIRSTNME	BIRTHDAY
1	JONES	FRANK	22MAY53
2	MCVADE	CURTIS	25DEC54
3	SMITH	VIRGINIA	14NOV49
4	TURNER	BECKY	26APR50

Output 4.16 Data Described by the View Descriptor, VLIB.EMPBD

Data Described by VLIB.EMPBD 1

OBS	LASTNAME	FIRSTNME	BIRTHDAY
1	AMEER	DAVID	10OCT51
2	BROOKS	RUBEN R.	25FEB52
3	BROWN	VIRGINA P.	24MAY46
4	CHAN	TAI	04JUL46
5	GARRETT	OLAN M.	23JAN35
6	GIBSON	GEORGE J.	23APR46
7	GOODSON	ALAN F.	21JUN50
8	JUAREZ	ARMANDO	28MAY47
9	LITTLEJOHN	FANNIE	17MAY54
10	RICHARDSON	TRAVIS Z.	30NOV37
11	RODRIGUEZ	ROMUALDO R	09FEB29
12	SCHOLL	MADISON A.	19MAR45
13	SHROPSHIRE	LELAND G.	04SEP49
14	SMITH	JERRY LEE	13SEP42
15	VAN HOTTEN	GWENDOLYN	13SEP42
16	WAGGONNER	MERRILEE D	27APR36
17	WILLIAMSON	JANICE L.	19MAY52

Output 4.17 Updated Data File, MYDATA. NEWBDAY

MYDATA.NEWBDAY Data File 1

OBS	LASTNAME	FIRSTNME	BIRTHDAY
1	AMEER	DAVID	10OCT51
2	BROOKS	RUBEN R.	25FEB52
3	BROWN	VIRGINA P.	24MAY46
4	CHAN	TAI	04JUL46
5	GARRETT	OLAN M.	23JAN35
6	GIBSON	GEORGE J.	23APR46
7	GOODSON	ALAN F.	21JUN50
8	JONES	FRANK	22MAY53
9	JUAREZ	ARMANDO	28MAY47
10	LITTLEJOHN	FANNIE	17MAY54
11	MCVADE	CURTIS	25DEC54
12	RICHARDSON	TRAVIS Z.	30NOV37
13	RODRIGUEZ	ROMUALDO R	09FEB29
14	SCHOLL	MADISON A.	19MAR45
15	SHROPSHIRE	LELAND G.	04SEP49
16	SMITH	JERRY LEE	13SEP42
17	SMITH	VIRGINIA	14NOV49
18	TURNER	BECKY	26APR50
19	VAN HOTTEN	GWENDOLYN	13SEP42
20	WAGGONNER	MERRILEE D	27APR36
21	WILLIAMSON	JANICE L.	19MAY52

Updating a Version 7 Data File

Versions 6 and 7 of the SAS System support different naming conventions, therefore, there may be character-length discrepancies between the variables in the original data file and the transaction data. You have two choices when updating a Version 7 data file with the data described by a view descriptor:

- let the compatibility engine truncate names exceeding 8 characters. The truncated variables will be added to the updated data file as new variables.
- rename the variables in the Version 7 data file to match the variable names in the descriptor file.

The following example resolves character-length discrepancies by using the RENAME DATA step option with the UPDATE statement. The Version 7 data file V7.CONSULTING_BIRTHDAYS, which contains Consulting names and birthdays, is updated with data described by VLIB.EMPBD.

```
proc sort data=v7.consulting_birthdays;
   by last_name;
run;

data newdata.new_birthdays;
   update v7.consulting_birthdays
   (rename=(last_name=lastname
            first_name=firstnme
```

```
            birthdate=birthday)) vlib.empbd;
   by lastname firstnme;
run;
```

In this example, the updated SAS data file NEWDATA.NEW_BIRTHDAYS is a Version 7 data file stored in the Version 7 SAS library associated with the libref NEWDATA. The RENAME= DATA step option is used with the UPDATE statement to rename the variables before the updated data file NEWDATA.NEW_BIRTHDAYS is created.

Output 4.18 on page 39 and Output 4.19 on page 40 show the results of the PRINT procedure on the original data file and the updated data file.

Output 4.18 Data File to be Updated, V7.CONSULTING_BIRTHDAYS

V7.Consulting_Birthdays Data File 1

obs	last_name	first_name	birthdate
1	JOHNSON	ED	30JAN65
2	LEWIS	THOMAS	25MAY54
3	SMITH	AMANDA	02DEC60
4	WILSON	REBECCA	13APR58

Output 4.19 Updated Data File, V7.NEW_BIRTHDAYS

V7.NEW_BIRTHDAYS Data File 1

obs	lastname	firstnme	birthday
1	AMEER	DAVID	10OCT51
2	BROOKS	RUBEN R.	25FEB52
3	BROWN	VIRGINA P.	24MAY46
4	CHAN	TAI	04JUL46
5	GARRETT	OLAN M.	23JAN35
6	GIBSON	GEORGE J.	23APR46
7	GOODSON	ALAN F.	21JUN50
8	JOHNSON	ED	30JAN65
9	JUAREZ	ARMANDO	28MAY47
10	LEWIS	THOMAS	25MAY54
11	LITTLEJOHN	FANNIE	17MAY54
12	RICHARDSON	TRAVIS Z.	30NOV37
13	RODRIGUEZ	ROMUALDO R	09FEB29
14	SCHOLL	MADISON A.	19MAR45
15	SHROPSHIRE	LELAND G.	04SEP49
16	SMITH	AMANDA	02DEC60
17	SMITH	JERRY LEE	13SEP42
18	VAN HOTTEN	GWENDOLYN	13SEP42
19	WAGGONNER	MERRILEE D	27APR36
20	WILLIAMSON	JANICE L.	19MAY52
21	WILSON	REBBECA	13APR58

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

Note: You cannot update a SYSTEM 2000 database directly using the DATA step, but you can update a SYSTEM 2000 database using the following procedures: APPEND, FSEDIT, FSVIEW, QUEST, and SQL. See Chapter 5, "Browsing and Updating SYSTEM 2000 Data," on page 43 for more information on updating SYSTEM 2000 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 best to extract SYSTEM 2000 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 SYSTEM 2000 data in several procedures over a period of time, you may improve performance by extracting. SAS data files are organized to provide optimal performance with PROC and DATA steps. SAS programs using SAS data files often use less CPU time than when they directly read SYSTEM 2000 data.
  - If you plan to read large amounts of data from a large SYSTEM 2000 database and the database is being shared by several users (Multi-User

environment), your direct reading of the data could adversely affect all users' response time.

- If you are the owner of a database, 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 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 an ORDER BY clause) on the SYSTEM 2000 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 (that generates an ordering-clause), or with an ordering-clause included 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 that corresponds to an item in the SYSTEM 2000 database, the interface view engine automatically generates an ordering-clause for that variable. Thus, the ordering-clause sorts the SYSTEM 2000 data before it uses the data in your SAS program. If the SYSTEM 2000 database is very large, this sorting can affect performance.

  If the view descriptor already has an ordering-clause and you specify a BY statement in your SAS code, the BY statement overrides the view descriptor's ordering-clause. When you use a SAS BY statement with a view descriptor, it is most efficient to use a BY variable that is associated with an indexed SYSTEM 2000 item.
- When writing SAS code and referencing a view descriptor, it is more efficient to use a WHERE statement in the code than it is to use a subsetting IF statement. The interface view engine passes the WHERE statement as a SYSTEM 2000 where-clause to the view descriptor, connecting it (using a Boolean AND) to any where-clause included in the view descriptor. (You can further optimize the selection criteria by using connecting strings. See "Connecting Strings" on page 139. Applying a WHERE clause to the SYSTEM 2000 data may reduce the number of entries processed, which often improves performance.

Refer to "Creating and Using View Descriptors Efficiently" on page 95 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 SYSTEM 2000® Data Management Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Interface to SYSTEM 2000® Data Management Software: Reference, Version 8

ISBN 1–58025–549–3

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

1st printing, October 1999