CHAPTER

5

Browsing and Updating SYSTEM 2000 Data

Introduction

The SAS/ACCESS interface to SYSTEM 2000 software enables you to browse and update SYSTEM 2000 data directly from a SAS session or program. This chapter shows you how to use SAS procedures for browsing data and updating data described by SAS/ACCESS view descriptors. The examples in this chapter use the EMPLOYEE database, and most of the examples use the view descriptor VLIB.EMPPOS that you created in Chapter 3, "Defining SAS/ACCESS Descriptor Files," on page 19. For the definition of the other view descriptors used in this chapter, see Appendix 3, "Example Data," on page 143.

Note: Many of these examples involve deleting and inserting data. Before you attempt to run these examples, check with your database administrator (DBA) to be sure the data in the database are correct. △

Before you can browse or update SYSTEM 2000 data, you must have access to the data through an appropriate password and authorities. SYSTEM 2000 software has several levels of passwords and authorities, and you may be allowed to display or browse data but not update values. Or you may be able to update values but not change the definition of the database. For these examples, the appropriate SYSTEM 2000 password (DEMO, which is the master password for the EMPLOYEE database) is stored in the view descriptors, so that you can use the SAS procedures described in this chapter to update the database.

Refer to Chapter 2, “SYSTEM 2000 Essentials,” on page 7 and Appendix 1, “Information for the Database Administrator,” on page 119 for more information on SYSTEM 2000 passwords and authorities.

In addition, it is important that you have some understanding of how SYSTEM 2000 logical entries map into SAS observations. This is especially important for updating. Refer to “Mapping Data between the SAS System and SYSTEM 2000 Software” on page 13.

Browsing and Updating with SAS/FSP Procedures

If your site has SAS/FSP software as well as SAS/ACCESS software, you can browse and update SYSTEM 2000 data described by a view descriptor from within a SAS program. You have a choice of three SAS/FSP procedures: FSBROWSE, FSEDIT, and FSVIEW. The FSBROWSE and FSEDIT procedures show you one observation at a time. The FSVIEW procedure displays multiple observations in a tabular format (somewhat similar to the PRINT procedure) where you can see many observations at once. PROC FSVIEW enables you to both browse and update SYSTEM 2000 data, depending on which option you choose.

You cannot use the FSBROWSE, FSEDIT, or FSVIEW procedures on an access descriptor.

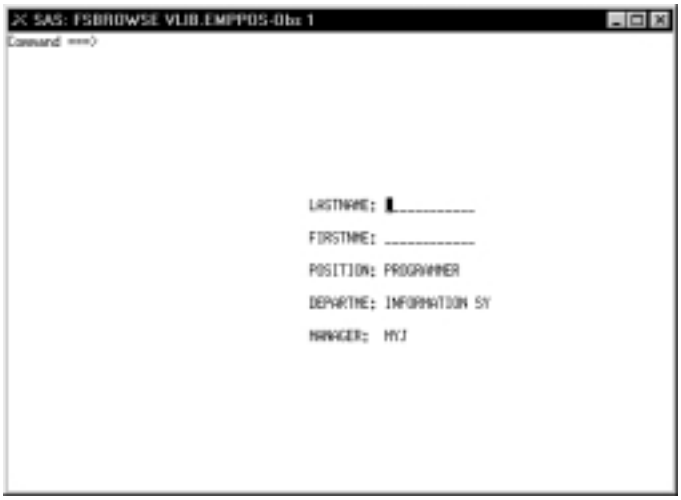
To scroll through the data from within a SAS/FSP window, use the FORWARD and BACKWARD commands. To end your browse or edit session, issue the END command.

Using the FSBROWSE Procedure

The FSBROWSE procedure enables you to look at SYSTEM 2000 data but not to change them. To use PROC FSBROWSE, submit the following SAS code:

```
proc fsbrowse data=vlib.emppos;
run;
```

The FSBROWSE procedure retrieves one observation from a SYSTEM 2000 database at a time. The following display shows the first observation of an employee's data described by the VLIB.EMPPOS view descriptor. (The view descriptor contains a SYSTEM 2000 ordering-clause to order the data by last name, which is missing for the first observation; that is, an employee has not yet been hired for the position.) To browse each observation, use the FORWARD and BACKWARD commands.



Using the FSEDIT Procedure

The FSEDIT procedure enables you to update SYSTEM 2000 data described by a view descriptor, if you have been granted the appropriate SYSTEM 2000 update authorities. For example, in the previous display, the LASTNAME and FIRSTNME values are missing in the first observation. You can add values to these items by using PROC FSEDIT.

```
proc fsedit data=vlib.emppos;
run;
```

PROC FSEDIT retrieves one observation at a time like PROC FSBROWSE. To edit data in the display, simply enter your changes. For example, for this observation, enter the value 'Adkins' for LASTNAME and 'Mary' for FIRSTNME.

To end your editing session, issue the END command. If you wanted to cancel an edit, you would issue the CANCEL command, but you must do this before you scroll to another observation. Once you scroll, the change is committed.

Using the FSVIEW Procedure

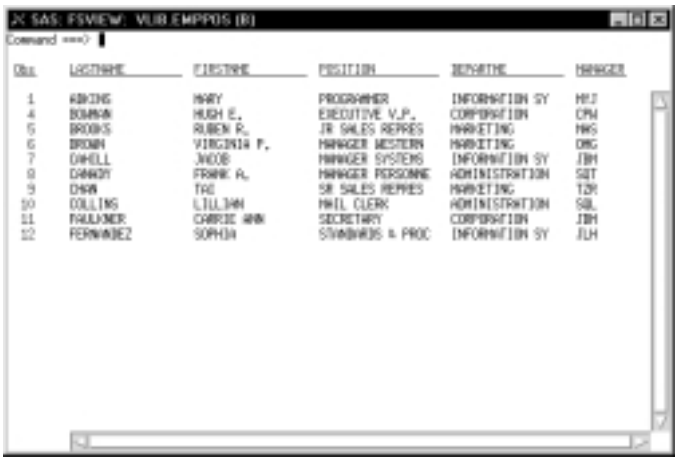
The FSVIEW procedure also enables you to browse or update SYSTEM 2000 data using a view descriptor, depending on how you invoke the procedure.

Using the FSVIEW Procedure to Browse SYSTEM 2000 Data

To browse SYSTEM 2000 data in a listing format, submit the code:

```
proc fsview data=vlib.emppos;
run;
```

Browse mode is the default for PROC FSVIEW. Notice that a (B) for browse follows the view descriptor's name. Also notice that the name Mary Adkins appears, reflecting the update you made using the FSEDIT procedure in the previous example.



Using the FSVIEW Procedure to Update SYSTEM 2000 Data

To edit SYSTEM 2000 data in a listing format, you must add the MODIFY option to the PROC FSVIEW statement, as shown in the following code:

```
proc fsview data=vlib.emppos modify;
run;
```

The same window as shown in the previous display appears, except the window title contains an (E), not a (B). *SAS/FSP Software: Usage and Reference* discusses in detail how to edit data using the FSVIEW procedure. Note that the CANCEL command in the FSVIEW window does not cancel your changes, whether you have scrolled or not.

Specifying a WHERE Clause While Browsing or Editing

You can specify a SAS WHERE statement with the SAS/FSP procedure statements to specify conditions that subset the retrieved SYSTEM 2000 data. You can also use a SAS WHERE command to do the same thing after you have invoked one of the SAS/FSP procedures.

Keep in mind that it is more efficient to use a WHERE clause rather than a subsetting IF statement. The interface view engine translates a WHERE clause into SYSTEM 2000 conditions and passes the conditions to SYSTEM 2000 software, connecting them by default using a Boolean AND, to any SYSTEM 2000 where-clause included in the view descriptor. Unlike a SYSTEM 2000 where-clause stored in a view descriptor, however, a SAS WHERE clause is restricted to items contained in the view descriptor. (A SYSTEM 2000 where-clause can reference items contained in a view descriptor and items contained in the access descriptor that the view descriptor is based on.)

Specifying selection criteria, whether in the form of a SAS WHERE clause or a SYSTEM 2000 where-clause, works essentially like filters. That is, more data goes into the clauses than comes out. Using the SAS/ACCESS interface, you can pass data through more than one filter, with each filter doing part of the subsetting. This is called successive filtering.

Sometimes, the interface view engine cannot translate all the SAS WHERE clause conditions into SYSTEM 2000 conditions. In such cases, the engine subsets what it can. As partially-filtered records are passed back to the SAS System, the SAS System automatically reapplies the entire WHERE clause as a second filter. (This is referred to as *post-processing*.) See “Using a SAS WHERE Clause for Selection Criteria” on page 134 for more information on what SAS WHERE clause conditions can and cannot be translated.

In some procedures, such as the FSEDIT procedure, you can continue to apply more filters by using the SAS WHERE command on the command line. Each time you enter another clause, the process of combining and filtering conditions is repeated. The engine decides what conditions it can handle, connects them by default with an AND to the prior conditions, sends them to SYSTEM 2000 software for the first (sometimes only) filtering, and then tells the SAS System to do any final filtering as required. For more information on using a SAS WHERE clause with a view descriptor, see "Using a SAS WHERE Clause for Selection Criteria" on page 134.

In the following example, the subset of retrieved employees are those from the Corporation Department, that is, the executives. An example of the FSEDIT window after the code has been submitted follows.

```
proc fsedit data=vlib.emppos;
   where departme='CORPORATION';
run;
```

Eleven observations with a DEPARTME value of CORPORATION are retrieved for editing. Note that the word (Subset) appears after VLIB.EMPPOS in the window title to remind you that the data retrieved are a subset of the data described by the view descriptor.

Had you subset the data from within the procedure with the command:

```
where departme='CORPORATION'
```

the results would be identical except the window title would say WHERE ... instead of (Subset) to indicate that a filter had been applied.

Although these examples have shown establishing a WHERE clause with the FSEDIT procedure, you can also establish WHERE clauses when using the FSBROWSE and FSVIEW procedures. For more information on the SAS WHERE statement, see *SAS Language Reference: Dictionary* and *SAS Language Reference: Concepts*. For more information on the WHERE command within the SAS/FSP procedures, refer to *SAS/FSP Software: Usage and Reference*.

Inserting and Deleting Data Records

When you do an insertion or a deletion for a SYSTEM 2000 database using a SAS/FSP procedure, you must be aware that the updates have the potential of affecting more than one data record in the database.

When you insert a new observation, it can cause the insertion of more than one SYSTEM 2000 data record, depending on how many levels the new observation represents and depending on a comparison between the data being inserted and the data in the last observation read, if any. During an insert, levels having data different from the prior observation, if any, result in a data record insertion. Depending on how many fields you change, one or more records are inserted at the levels that have changed. If your application inserts records in a random fashion, for example you want to add a position record to one employee while looking at another employee, then you should specify a BY key in your view descriptor. See Appendix 2, “Advanced User Topics,” on page 127 for more information on inserting data records and on using a BY key to resolve ambiguous inserts.

When you delete an observation, the results are not obvious to you and may be difficult to predict. The interface view engine must handle deletes carefully to ensure that what you request to delete does not adversely affect another user of the database. When you issue the DELETE command, you can expect one of the following results:

- At the very least, the items in the bottom-level record of your view descriptor are set to null (missing).
- At the very most, all the data records in the observation are physically removed from the database.
- Between those two results, the interface view engine makes a case-by-case decision on each record in the view. If the record has descendants, it is not affected. If the record has no descendants, it is physically removed.

The following example illustrates using the DELETE command in the FSEDIT procedure. Suppose you want to edit the SYSTEM 2000 data described by VLIB.EMPPOS to delete the following observation. If you have been granted update authority, you can use the PROC FSEDIT statement. Scroll forward to the observation you want to delete, and issue the DELETE command on the command line, as shown in the following display.

The DELETE command processes the deletion and displays a message to that affect, as shown in the following display. There is no indication to you of what actions the interface view engine takes. Regardless of the actions, though, the observation you delete is no longer available for processing.

Even though it looks as if the entire observation is removed from the database, this is not the case in this example. The interface view engine sets the values for the bottom level items (POSITION, DEPARTME, and MANAGER) to missing; the records are not physically removed because the POSITION WITHIN COMPANY RECORD has descendant records that would be affected by a removal. The values for LASTNAME and FORENAME are not affected because they are at level 0 with descendant records. Also, values for other items in the POSITION WITHIN COMPANY record are not affected.

For more information on using the SAS/FSP procedures, see *SAS/FSP Software: Usage and Reference*.

Browsing and Updating with the SQL Procedure

The SAS System's SQL procedure enables you to retrieve and update data from SYSTEM 2000 databases. You can retrieve and browse SYSTEM 2000 data by specifying a view descriptor in a PROC SQL SELECT statement. To update the data, you can specify view descriptors in the PROC SQL INSERT, DELETE, and UPDATE statements. You must be granted update authority before you can edit SYSTEM 2000 data. Here is a summary of these PROC SQL statements:

DELETE
: deletes values from a SYSTEM 2000 database.

INSERT
: inserts values in a SYSTEM 2000 database.

SELECT
: retrieves and displays data from SYSTEM 2000 databases. A SELECT statement is usually referred to as a query because it queries the database for information.

UPDATE
: updates values in a SYSTEM 2000 database.

The query in the following example retrieves and displays values in the SYSTEM 2000 EMPLOYEE database that are described by the VLIB.EMPPOS view descriptor, assuming that the previous updates with the SAS/FSP procedures have occurred (that is, you added the values Mary and Adkins to the programmer position.) Notice that the SQL procedure prints the variable labels instead of the SAS variable names and the data are displayed in the SAS OUTPUT window. Notice also that the SELECT

statement executes without your using a RUN statement, because the SQL procedure displays output data automatically, without your using the PRINT procedure, and executes when you submit it.

```
proc sql;
   title 'SYSTEM 2000 Data Output Using
      a SELECT Statement';
select *
/* Asterisk indicates 'select all items' */
   from vlib.emppos;
```

Output 5.1 on page 50 displays the query's results.

Output 5.1 SYSTEM 2000 Data Output Using a PROC SQL Query

SYSTEM 2000 Data Output Using a SELECT Statement

LAST NAME	FORENAME	POSITION TITLE	DEPARTMENT	MANAGER
ADKINS	MARY	PROGRAMMER	INFORMATION SY	MYJ
AMEER	DAVID	JR SALES REPRESE	MARKETING	VPB
AMEER	DAVID	SR SALES REPRESE	MARKETING	VPB
BOWMAN	HUGH E.	EXECUTIVE VICE-P	CORPORATION	CPW
BROOKS	RUBEN R.	JR SALES REPRESE	MARKETING	MAS
BROWN	VIRGINA P.	MANAGER WESTERN	MARKETING	OMG
CAHILL	JACOB	MANAGER SYSTEMS	INFORMATION SY	JBM
CANADY	FRANK A.	MANAGER PERSONNE	ADMINISTRATION	PRK
CHAN	TAI	SR SALES REPRESE	MARKETING	TZR
COLLINS	LILLIAN	MAIL CLERK	ADMINISTRATION	SQT
FAULKNER	CARRIE ANN	SECRETARY	CORPORATION	JBM

As in the SAS/FSP procedures, you can specify a WHERE clause as part of the SELECT statement to subset the observations you want to display. The following example requests the employees that are technical writers.

```
title 'SYSTEM 2000 Data Output Subset by a
   WHERE Clause';
select *
   from vlib.emppos
   where position='TECHNICAL WRITER';
```

Notice that the PROC SQL statement is not repeated in this query. With the SQL procedure, you do not need to repeat the PROC statement unless you use another SAS procedure or the DATA step between PROC SQL statements. Because you are referencing a view descriptor, you use the SAS names for items in the WHERE clause.Output 5.2 on page 51 displays the one employee who is a technical writer.

Output 5.2 SYSTEM 2000 Data Output Subset by a WHERE Clause

```
              SYSTEM 2000 Data Output Subset by a WHERE Clause

   LAST NAME   FORENAME    POSITION TITLE    DEPARTMENT      MANAGER
   ------------------------------------------------------------------
   GIBSON      MOLLY I.    TECHNICAL WRITER  INFORMATION SY  JC
```

You can use the UPDATE statement to update SYSTEM 2000 data. Remember that when you reference a view descriptor in a PROC SQL statement, you are not updating the view descriptor, but rather the SYSTEM 2000 data described by the view descriptor. Suppose that Mary Adkins, whom you previously added to the unfilled programmer position, decided to change her position from programmer to technical writer. You could update her position title and manager as follows:

```
update vlib.emppos
   set position='TECHNICAL WRITER'
   where lastname='ADKINS';
update vlib.emppos
   set manager='JC'
   where lastname='ADKINS';

   title 'Updated VLIB.EMPPOS View Descriptor';
select *
   from vlib.emppos;
```

Output 5.3 on page 51 displays the query's results.

Output 5.3 Updated VLIB.EMPPOS View Descriptor

```
                Updated VLIB.EMPPOS View Descriptor                  1

LAST NAME   FORENAME    POSITION TITLE    DEPARTMENT      MANAGER
------------------------------------------------------------------
ADKINS      MARY        TECHNICAL WRITER  INFORMATION SY  JC
AMEER       DAVID       SR SALES REPRESE  MARKETING       VPB
AMEER       DAVID       JR SALES REPRESE  MARKETING       VPB
BOWMAN      HUGH E.     EXECUTIVE VICE-P  CORPORATION     CPW
BROOKS      RUBEN R.    JR SALES REPRESE  MARKETING       MAS
BROWN       VIRGINA P.  MANAGER WESTERN   MARKETING       OMG
CAHILL      JACOB       MANAGER SYSTEMS   INFORMATION SY  JBM
CANADY      FRANK A.    MANAGER PERSONNE  ADMINISTRATION  PRK
CHAN        TAI         SR SALES REPRESE  MARKETING       TZR
COLLINS     LILLIAN     MAIL CLERK        ADMINISTRATION  SQT
FAULKNER    CARRIE ANN  SECRETARY         CORPORATION     JBM
```

You can use the INSERT statement to add values to a SYSTEM 2000 database or the DELETE statement to remove values as described by a view descriptor. In the following example, the values described by the view descriptor VLIB.EMPPOS for the employee with the last name of Adkins are deleted from the EMPLOYEE database:

```
delete from vlib.emppos
   where lastname='ADKINS';

   title 'Data Deleted from SYSTEM 2000 EMPLOYEE
     Database';
select *
 from vlib.emppos;
```

Output 5.4 on page 52 displays the query's results.

Output 5.4 VLIB.EMPPOS Data with an Observation Deleted

```
          Data Deleted from SYSTEM 2000 EMPLOYEE Database               1

LAST NAME   FORENAME    POSITION TITLE    DEPARTMENT      MANAGER
-----------------------------------------------------------------------
ADKINS      MARY
AMEER       DAVID       SR SALES REPRESE  MARKETING       VPB
AMEER       DAVID       JR SALES REPRESE  MARKETING       VPB
BOWMAN      HUGH E.     EXECUTIVE VICE-P  CORPORATION     CPW
BROOKS      RUBEN R.    JR SALES REPRESE  MARKETING       MAS
BROWN       VIRGINA P.  MANAGER WESTERN   MARKETING       OMG
CAHILL      JACOB       MANAGER SYSTEMS   INFORMATION SY  JBM
CANADY      FRANK A.    MANAGER PERSONNE  ADMINISTRATION  PRK
CHAN        TAI         SR SALES REPRESE  MARKETING       TZR
COLLINS     LILLIAN     MAIL CLERK        ADMINISTRATION  SQT
FAULKNER    CARRIE ANN  SECRETARY         CORPORATION     JBM
```

CAUTION:

You must use the WHERE clause in the DELETE statement. If you omit the WHERE clause from a SQL procedure DELETE statement, you will delete all the data in the database accessed by the view descriptor. △

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

Appending Data with the APPEND Procedure

With the APPEND procedure, you can append data described by SAS/ACCESS view descriptors and PROC SQL views to SAS data files. You can also update the data described by a view descriptor by appending to it the data from another SAS data set.

For an append operation to be successful, the variables in the BASE=, or target, data set and the variables in DATA=, or source, data set must match, or you must use the

FORCE= option to concatenate the data sets. The FORCE= option causes the APPEND procedure to drop the extra variables and issues a warning.

You can append the data described by a view descriptor to a Version 6 or Version 7 data file and vice versa. For variables that use the longer Version 7 naming conventions or otherwise do not match, use the RENAME= data set option in the APPEND procedure to rename the variables.

Example of Appending Data to a SAS Data File

In the following example, two managers have kept separate employee telephone lists. The Marketing manager kept records in the SYSTEM 2000 EMPLOYEE database described by the view descriptor VLIB.EMPPHON. The Corporation manager kept records for the executive telephone list in a Version 6 SAS data file, MYDATA.CORPHON. The two sources must be combined to create a telephone list of both departments.

The data described by the view descriptor VLIB.EMPPHON and the data in the SAS data file MYDATA.CORPHON are displayed in Output 5.5 on page 53 and Output 5.6 on page 54.

```
proc print data=vlib.empphon;
   title 'Marketing Phone List';
run;

proc print data=mydata.corphon;
   title 'Corporation Phone List';
run;
```

Output 5.5 Data Described by VLIB.EMPPHON

Marketing Phone List 1

OBS	LASTNAME	FIRSTNME	PHONE
1	AMEER	DAVID	545 XT495
2	BROOKS	RUBEN R.	581 XT347
3	BROWN	VIRGINA P.	218 XT258
4	CHAN	TAI	292 XT331
5	GARRETT	OLAN M.	212 XT208
6	GIBSON	GEORGE J.	327 XT703
7	GOODSON	ALAN F.	323 XT512
8	JUAREZ	ARMANDO	506 XT987
9	LITTLEJOHN	FANNIE	219 XT653
10	RICHARDSON	TRAVIS Z.	243 XT325
11	RODRIGUEZ	ROMUALDO R	243 XT874
12	SCHOLL	MADISON A.	318 XT419
13	SHROPSHIRE	LELAND G.	327 XT616
14	SMITH	JERRY LEE	327 XT169
15	VAN HOTTEN	GWENDOLYN	212 XT311
16	WAGGONNER	MERRILEE D	244 XT914
17	WILLIAMSON	JANICE L.	218 XT802

Output 5.6 Data in MYDATA. CORPHON

Corporation Phone List 1

OBS	LASTNAME	FIRSTNME	PHONE
1	BOWMAN	HUGH E.	109 XT901
2	FAULKNER	CARRIE ANN	132 XT417
3	GARRETT	OLAN M.	212 XT208
4	KNAPP	PATRICE R.	222 XT 12
5	KNIGHT	ALTHEA	213 XT218
6	MILLSAP	JOEL B.	131 XT224
7	MUELLER	PATSY	223 XT822
8	NATHANIEL	DARRYL	118 XT544
9	SALAZAR	YOLANDA	111 XT169
10	WATERHOUSE	CLIFTON P.	101 XT109

You can combine the data described by these two sources using the APPEND procedure as follows:

```
proc append base=mydata.corphon data=vlib.empphon;
run;

proc sort data=mydata.corphon;
by lastname;

proc print data=mydata.corphon;
   title 'Corporation and Marketing Phone List';
run;
```

Output 5.7 on page 55 displays the data in the updated data file MYDATA.CORPHON. Notice that the combined data is sorted by last name. Also, since PROC PRINT was used to display the data, the variable names are used (for example, FIRSTNME), not the variable labels, which are the item names (for example, FORENAME), as with the SQL procedure.

Output 5.7 Appended Data

Corporation and Marketing Phone List 1

OBS	LASTNAME	FIRSTNME	PHONE
1	AMEER	DAVID	545 XT495
2	BOWMAN	HUGH E.	109 XT901
3	BROOKS	RUBEN R.	581 XT347
4	BROWN	VIRGINA P.	218 XT258
5	CHAN	TAI	292 XT331
6	FAULKNER	CARRIE ANN	132 XT417
7	GARRETT	OLAN M.	212 XT208
8	GARRETT	OLAN M.	212 XT208
9	GIBSON	GEORGE J.	327 XT703
10	GOODSON	ALAN F.	323 XT512
11	JUAREZ	ARMANDO	506 XT987
12	KNAPP	PATRICE R.	222 XT 12
13	KNIGHT	ALTHEA	213 XT218
14	LITTLEJOHN	FANNIE	219 XT653
15	MILLSAP	JOEL B.	131 XT224
16	MUELLER	PATSY	223 XT822
17	NATHANIEL	DARRYL	118 XT544
18	RICHARDSON	TRAVIS Z.	243 XT325
19	RODRIGUEZ	ROMUALDO R	243 XT874
20	SALAZAR	YOLANDA	111 XT169
21	SCHOLL	MADISON A.	318 XT419
22	SHROPSHIRE	LELAND G.	327 XT616
23	SMITH	JERRY LEE	327 XT169
24	VANHOTTEN	GWENDOLYN	212 XT311
25	WAGGONNER	MERRILEE D	244 XT914
26	WATERHOUSE	CLIFTON P.	101 XT109
27	WILLIAMSON	JANICE L.	218 XT802

The APPEND procedure also accepts a WHERE= data set option or a WHERE statement to subset which observations from the DATA= data set are added to the BASE= data set, as shown in the following example. (It is assumed that the MYDATA.CORPHON data file is in its original state before issuing the previous APPEND procedure.)

```
proc append base=mydata.corphon
  data=vlib.empphon(where=(lastname='AMEER'));
run;

proc print data=mydata.corphon;
  title2 'Appended Data with a WHERE= Data Set Option';
run;
```

Output 5.8 on page 56 displays the data when the observations appended to the BASE= data set are subset by the WHERE= data set option.

Output 5.8 Appended Data with a WHERE= Data Set Option

Appended Data With a WHERE= Data Set Option 1

OBS	LASTNAME	FIRSTNME	PHONE
1	BOWMAN	HUGH E.	109 XT901
2	FAULKNER	CARRIE ANN	132 XT417
3	GARRETT	OLAN M.	212 XT208
4	KNAPP	PATRICE R.	222 XT 12
5	KNIGHT	ALTHEA	213 XT218
6	MILLSAP	JOEL B.	131 XT224
7	MUELLER	PATSY	223 XT822
8	NATHANIEL	DARRYL	118 XT544
9	SALAZAR	YOLANDA	111 XT169
10	WATERHOUSE	CLIFTON P.	101 XT109
11	AMEER	DAVID	545 XT495

Appending Data to a Version 7 Data File

Had the Corporation manager in the previous example kept his records in a Version 7 SAS data file named V7.CORPHON (see Output 5.9 on page 57) and used variable names longer than 8 characters, the data in VLIB.EMPPHON could be appended with the following code:

```
proc append base=v7.corphon
   (rename (firstname=firstnme))
   data=vlib.empphon;
run;

proc sort data=v7.corphon;
   by lastname;

proc print data=v7.corphon;
  title2 'Corporation and Marketing Phone List';
run;
```

Output 5.9 Data in V7.CORPHON

Corporation Phone List

Obs	lastname	firstname	phone
1	BOWMAN	HUGH E.	109 XT901
2	FAULKNER	CARRIE ANN	132 XT417
3	GARRETT	OLAN M.	212 XT208
4	KNAPP	PATRICE M.	222 XT 12
5	KNIGHT	ALTHEA	213 XT218
6	MILLSAP	JOEL B.	131 XT224
7	MUELLER	PATSY	223 XT822
8	NATHANIEL	DARRYL	118 XT544
9	SALAZAR	YOLANDA	111 XT169
10	WATERHOUSE	CLIFTON P.	101 XT109

In this example, the RENAME= data set option is used to reconcile a character-length discrepancy between the firstname variable in the V7 data file and the firstnme variable in the view descriptor.Output 5.10 on page 58 shows a portion of the data in the updated data file V7.CORPHON.

Output 5.10 Data in V7.CORPHON with Data from VLIB.EMPPHON Appended to It

Corporation and Marketing Phone List

Obs	lastname	firstnme	phone
1			
2	AMEER	DAVID	545 XT495
3	BOWMAN	HUGH E.	109 XT901
4	BOWMAN	HUGH E.	109 XT901
5	BROOKS	RUBEN R.	581 XT347
6	BROWN	VIRGINA P.	218 XT258
7	CAHILL	JACOB	435 XT426
8	CANADY	FRANK A.	153 XT406
9	CHAN	TAI	292 XT331
10	COLLINS	LILLIAN	166 XT616
11	FAULKNER	CARRIE ANN	132 XT417
12	FAULKNER	CARRIE ANN	132 XT417
13	FERNANDEZ	SOPHIA	414 XT847
14	FREEMAN	LEOPOLD	436 XT604
15	GARCIA	FRANCISCO	414 XT348
16	GARRETT	OLAN M.	212 XT208
17	GARRETT	OLAN M.	212 XT208
18	GIBSON	GEORGE J.	327 XT703
19	GIBSON	MOLLY I.	323 XT227
20	GOODSON	ALAN F.	323 XT512
21	HERNANDEZ	JESSE L.	444 XT448
22	JOHNSON	BRADFORD	244 XT446
23	JONES	MICHAEL Y.	434 XT713
24	JONES	RITA M.	127 XT271
25	JUAREZ	ARMANDO	506 XT987
26	KAATZ	FREDDIE	243 XT387
27	KNAPP	PATRICE M.	222 XT 12
28	KNAPP	PATRICE R.	222 XT123
29	KNIGHT	ALTHEA	213 XT218
30	KNIGHT	ALTHEA	213 XT218
31	LITTLEJOHN	FANNIE	219 XT653
32	MILLSAP	JOEL B.	131 XT224
33	MILLSAP	JOEL B.	131 XT224
34	MUELLER	PATSY	223 XT822
35	MUELLER	PATSY	223 XT822
36	NATHANIEL	DARRYL	118 XT544
37	NATHANIEL	DARRYL	118 XT544
38	POLANSKI	IVAN L.	506 XT621

Appending SAS Data to a View Descriptor

When appending SAS data to a view descriptor, you will not be able to sort the data unless you specify an output data file. To sort the data in the view descriptor, you would have to sort the SYSTEM 2000 database, which is not recommended.

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

Browsing and Updating Using the QUEST Procedure

The QUEST procedure allows you to access a SYSTEM 2000 database directly, that is, without using a view descriptor. The procedure is basically a messenger for SYSTEM

2000 statements: when you submit a statement in the QUEST procedure, the SAS System scans the statement and passes it to SYSTEM 2000 software, which then executes it.

SYSTEM 2000 software includes an interactive language, also called QUEST, that is used for creating, browsing, updating, and managing SYSTEM 2000 databases. The QUEST procedure gives you full access to that language, either from interactive line-mode sessions or batch mode. In effect, when you submit the PROC QUEST statement, you start a SYSTEM 2000 session; when you submit the END statement, you end the session.

Because the QUEST language is interactive, SYSTEM 2000 software responds to each statement as soon as you submit it. Like the SQL procedure, you do not use a RUN statement to have the statements executed; you simply submit them.

The following example illustrates how to browse and update a SYSTEM 2000 database using the QUEST procedure.

Suppose management is considering a reorganization. They request a list of all managers. That information is available in the EMPLOYEE database, which can be accessed in Multi-User mode. First, submit the PROC QUEST statement.

```
proc quest s2kmode=m;
```

A message appears in the LOG window verifying that you have accessed SYSTEM 2000 software. Now, submit SYSTEM 2000 statements to specify your password for the database and to open the database.

```
user, demo;
data base name is employee;
```

Request a list of managers with the SYSTEM 2000 TALLY statement.

```
tally manager;
```

Output 5.11 on page 60 displays the list produced.

Output 5.11 TALLY Statement Output

```
****************************************
 ITEM-         MANAGER
****************************************
OCCURRENCES    VALUE
----------------------------------------
         1     AFG
         3     CPW
         2     FAC
         3     GVH
         5     HEB
         2     ILP
         4     JBM
         3     JC
         1     JFS
         2     JLH
         1     MAS
         3     MYJ
         4     OMG
         3     PQ
         3     PRK
         1     RMJ
         3     SQT
         4     TZR
         7     VPB
----------------------------------------
       19 DISTINCT VALUES
----------------------------------------
       55 TOTAL OCCURRENCES
----------------------------------------
```

Now, you could end the SYSTEM 2000 session and print your report.

```
exit;
```

Suppose that Olan Garrett, the Vice-President for Marketing, wants to make just one change in his department. He decides to have Jerry Smith report to a different manager. Again, use the QUEST procedure to access the EMPLOYEE database.

```
proc quest s2kmode=m;
user,demo; data base name is employee;
```

Request a list of all Marketing employees and their current managers with the SYSTEM 2000 LIST statement.

```
list employee number, last name, forename,
   manager,
  ordered by manager
  where department eq marketing at 1;
```

Output 5.12 on page 61 displays the list produced.

Output 5.12 LIST Statement Output

```
* EMPLOYEE NUMBER   LAST NAME    FORENAME              MANAGER
***
*            1313   SMITH        JERRY LEE             AFG
*            1217   RODRIGUEZ    ROMUALDO R            GVH
*            1077   GIBSON       GEORGE J.             GVH
*            1133   WILLIAMSON   JANICE L.             GVH
*            1327   BROOKS       RUBEN R.              MAS
*            1011   VAN HOTTEN   GWENDOLYN             OMG
*            1161   RICHARDSON   TRAVIS Z.             OMG
*            1007   BROWN        VIRGINIA P.           OMG
*            1017   WAGGONNER    MERRILEE D            TZR
*            1119   GOODSON      ALAN F.               TZR
*            1234   SHROPSHIRE   LELAND G.             TZR
*            1031   CHAN         TAI                   TZR
*            1050   AMEER        DAVID                 VPB
*            1145   JUAREZ       ARMANDO               VPB
*            1015   SCHOLL       MADISON A.            VPB
*            1062   LITTLEJOHN   FANNIE                VPB
```

After looking at the report, Olan Garrett decides to have Jerry Lee Smith report to Madison Scholl. The SYSTEM 2000 statement you would enter is as follows:

```
change manager eq mas* wh employee number eq 1313;
```

SYSTEM 2000 software issues a message that one record was selected for the change. To end the SYSTEM 2000 session, issue:

```
end;
```

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