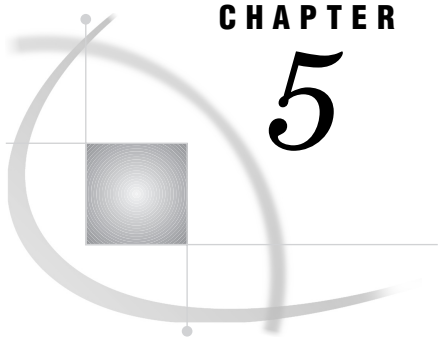


CHAPTER

5

Browsing and Updating IMS-DL/I Data

Introduction

The SAS/ACCESS interface to IMS-DL/I enables you to browse and update your IMS-DL/I data directly from a SAS session or program. This chapter shows you how to use SAS procedures to review and update IMS-DL/I data that is described by SAS/ACCESS view descriptors. The examples in this chapter use the view descriptors VLIB.CUSTINFO and VLIB.CHCKACCT. See Appendix 2 for definitions of all the view descriptors referenced in this chapter. Appendix 2 also includes the IMS-DL/I database and SAS data files and data sets.

To browse or update IMS-DL/I data, you must use a Program Specification Block (PSB) that contains a Program Communication Block (PCB) with the level of access desired. You need to have this desired level of access to the database, to the segments in that database, and to the fields in those segments. The types of access that a PCB can allow include

G	get
I	insert
R	replace
D	delete
A	all

Refer to Chapter 2, “Understanding IMS-DL/I Essentials,” on page 11 and “Program Specification Block” on page 23 for more information on accessing IMS-DL/I data.

READ, WRITE, ALTER, or PW passwords can be assigned to a view descriptor, access descriptor, PROC SQL view, DATA step view, or SAS data file. See Chapter 6, "ACCESS Procedure Reference," on page 93 and "SAS System Passwords for SAS/ ACCESS Descriptors" on page 96 for more information on assigning passwords.

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 IMS-DL/I data described by a view descriptor from within a SAS/FSP procedure.

You can use any of three SAS/FSP procedures: FSBROWSE, FSEDIT, and FSVIEW. The FSBROWSE and FSEDIT procedures display one observation at a time, while the FSVIEW procedure produces multiple observations in a tabular format, similar to the PRINT procedure. PROC FSVIEW enables you both to browse and update IMS-DL/I data, depending on which option you choose. The FSBROWSE, FSEDIT, or FSVIEW procedures can only be used with data accessed by a view descriptor, PROC SQL view, DATA step view, or SAS data file; you cannot reference an access descriptor with any SAS procedure or in the SAS DATA step.

Note: The formats assigned by the ACCESS procedure are by default used as informats by the SAS/FSP procedures when you add or update a path of data. △

Using the FSBROWSE Procedure

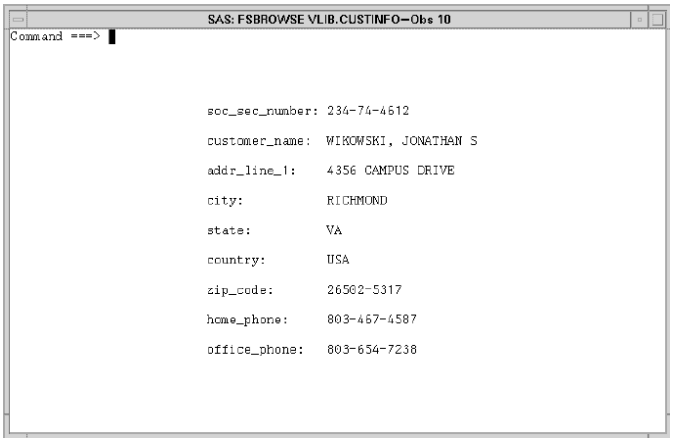
The FSBROWSE procedure enables you to look at IMS-DL/I data but does not allow you to change them. To use PROC FSBROWSE, submit the following SAS statements in the PROGRAM EDITOR window:

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

The FSBROWSE procedure retrieves observations from an IMS-DL/I database one at a time.

Display 5.1 on page 70 shows the last observation of the customers' data described by the VLIB.CUSTINFO view descriptor. To browse each observation, issue the FORWARD and BACKWARD commands. Because a view descriptor can describe only one path of data in an IMS-DL/I database, you can browse observations in one path of data only.

Display 5.1 Browsing IMS-DL/I Data in the FSBROWSE Window



For more information on the FSBROWSE procedure, see "The FSBROWSE Procedure" in *SAS/FSP Software: Usage and Reference*.

Note: Accessing observations by observation number is not supported for IMS-DL/I view descriptors within the FSBROWSE procedure; a WHERE command can be used to view a subset of the data. △

Using the FSEDIT Procedure

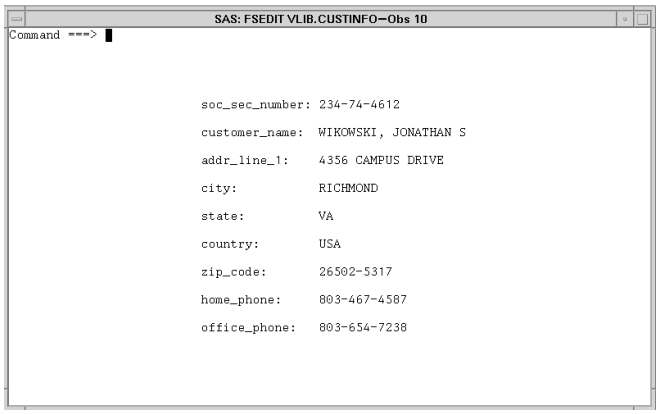
The FSEDIT procedure enables you to update the IMS-DL/I data described by a view descriptor if the view descriptor specifies in your PSB a PCB that allows you the appropriate level of update access (insert, replace, delete, or all) for the database segments. For example, if the area codes used in HOME_PHONE and OFFICE_PHONE are incorrect for Richmond, you can correct them with the FSEDIT procedure.

To use PROC FSEDIT, submit the following statements from the PROGRAM EDITOR window:

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

An FSEDIT window appears that looks like the FSBROWSE window. Scroll to the observation you want, or enter a WHERE statement to display the correct observation. You can then add or further update the information about customer **JONATHAN S. WIKOWSKI**, as shown in Display 5.2 on page 71.

Display 5.2 Updating Information in the FSEDIT Window



For more information on the FSEDIT procedure, see "The FSEDIT Procedure" in *SAS/FSP Software: Usage and Reference*.

Using the FSVIEW Procedure

The FSVIEW procedure also enables you to browse or update IMS-DL/I data using a view descriptor, depending on how you invoke the procedure.

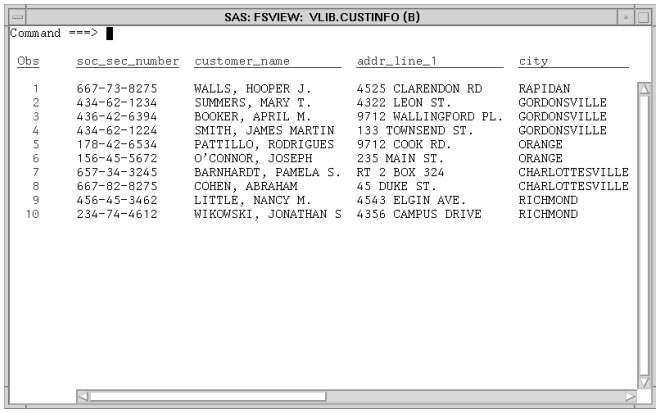
Using the FSVIEW Procedure to Browse IMS-DL/I Data

To browse IMS-DL/I data in a tabular format, submit the following PROC FSVIEW statements in the PROGRAM EDITOR:

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

Browse mode is the default for the FSVIEW procedure. The submitted statements produce the window shown in Display 5.3 on page 72.

Display 5.3 Browsing IMS-DL/I Data in the FSVIEW Window



Obs	soc_sec_number	customer_name	addr_line_1	city
1	667-73-8275	WALLS, HOOPER J.	4525 CLARENDON RD	RAPIDAN
2	434-62-1234	SUMMERS, MARY T.	4322 LEON ST.	GORDONSVILLE
3	436-42-6394	BOOKER, APRIL M.	9712 WALLINGFORD PL.	GORDONSVILLE
4	434-62-1224	SMITH, JAMES MARTIN	133 TOWNSEND ST.	GORDONSVILLE
5	178-42-6534	PATTILLO, RODRIGUES	9712 COOK RD.	ORANGE
6	156-45-5672	O'CONNOR, JOSEPH	235 MAIN ST.	ORANGE
7	657-34-3245	BARNHARDT, PAMELA S.	RT 2 BOX 324	CHARLOTTESVILLE
8	667-82-8275	COHEN, ABRAHAM	45 DUKE ST.	CHARLOTTESVILLE
9	456-45-3462	LITTLE, NANCY M.	4543 ELGIN AVE.	RICHMOND
10	234-74-4612	WIKOWSKI, JONATHAN S	4356 CAMPUS DRIVE	RICHMOND

Using the FSVIEW Procedure to Update IMS-DL/I Data

To edit the IMS-DL/I data in a tabular format, you must add the EDIT or MODIFY option to the PROC FSVIEW statement, as shown here:

```
proc fsview data=vlib.custinfo edit;
run;
```

Note: The CANCEL command in the FSVIEW window does *not* cancel your changes; it ends the browse or edit session. △

Specifying a SAS WHERE Statement While Browsing or Editing

If the IMS-DL/I engine can generate SSAs from the WHERE statement, it then retrieves a subset of the IMS-DL/I data. If the engine cannot generate SSAs from the WHERE statement, the WHERE statement is passed to the SAS System for processing. You can also use a SAS WHERE command to retrieve a subset of the IMS-DL/I data after you have invoked one of the SAS/FSP procedures using the PROC statements.

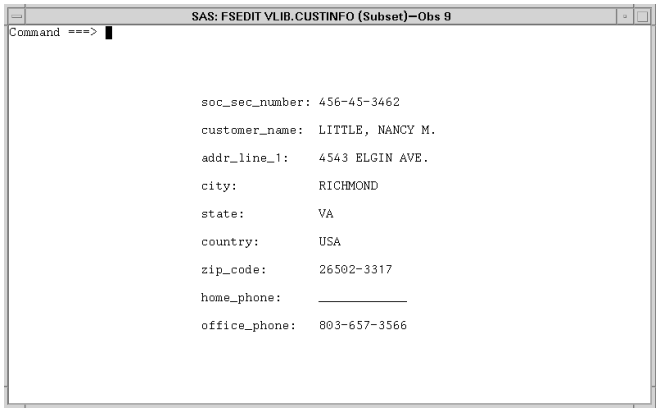
If you use a SAS WHERE *statement,* only the observations specified by that SAS WHERE statement are available. The other observations are not available until you exit the procedure. This is called a *permanent WHERE clause.*

If you use the SAS WHERE *command,* you can clear the command to make all the observations available. This is called a *temporary WHERE clause.*

In the following example, the FSEDIT procedure uses a SAS WHERE statement to retrieve a subset of customers from Richmond. Display 5.4 on page 73 shows the FSEDIT window after the statements have been submitted.

```
proc fsedit data=vlib.custinfo;
   where city='RICHMOND';
run;
```

Display 5.4 Submitting a SAS WHERE Statement While Invoking FSEDIT



Only the two observations with a CITY value of **RICHMOND** are retrieved for editing; you must scroll forward to see the second observation. The word **(Subset)** appears after VLIB.CUSTINFO in the window title to remind you that the data retrieved are a subset of the data described by the view descriptor. You can then edit each observation by typing over any incorrect information. Issue the END command to end your editing session. If you want to cancel changes to an observation, you can issue the CANCEL command before you scroll to another observation. Once you scroll, the changes are saved.

You can also enter a SAS WHERE command to display a subset of your data. A SAS WHERE command is a SAS WHERE expression that you enter on the command line. To begin the FSEDIT procedure, submit the following commands in the PROGRAM EDITOR:

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

Display 5.5 on page 73 shows what the FSEDIT display looks like when the following command-line command is entered within the FSEDIT window:

```
where city='RICHMOND'
```

Display 5.5 Entering a SAS WHERE Command in an FSEDIT Window

Only the two observations with a CITY value of **RICHMOND** are retrieved for editing; you must scroll forward to see the second observation. You can then edit each observation, as described earlier.

Although these examples have shown how to use a SAS WHERE statement and command with the FSEDIT procedure, you can use a SAS WHERE statement and command in the same way with the FSBROWSE and FSVIEW procedures. For more information on the SAS WHERE statement, refer to *SAS Language Reference: Dictionary*. For more information on the SAS WHERE command within the SAS/FSP procedures, refer to *SAS/FSP Software: Usage and Reference*.

Scrolling with SAS/FSP Procedures

Scrolling through data using the FSEDIT, FSBROWSE, or FSVIEW procedures is different when you are using view descriptors instead of SAS data files. While the FORWARD command works identically in both cases, the BACKWARD command does not.

Scrolling backward with SAS/FSP procedures can be slow when you are working with a large database, particularly when you are looking at a path of data in a record near the end of the database. To scroll backward through an IMS-DL/I database, the IMS-DL/I engine must read forward in the database from the beginning until it reaches the observation preceding the one that is displayed when the BACKWARD command was issued. For example, suppose the view defines 5,000 observations, and the current observation is 3,400. To scroll backward to observation 3,399, the FSEDIT procedure must sequentially read observations 1 through 3,398. This can be expensive and time consuming.

Inserting and Deleting Segments with SAS/FSP Procedures

Inserting and deleting database segments with SAS/FSP procedures is also different when you are using view descriptors rather than SAS data files.

You can use the FSEDIT and FSVIEW procedures to insert segments into one path of an IMS-DL/I database on which a view descriptor is based, assuming you are using a PCB that allows you insert access to the database segments. There are two ways to add a new segment to an IMS-DL/I database using SAS/FSP procedures:

- To insert one path of data, type **ADD** on the command line and press ENTER. You can then enter an entire path of data, which the IMS-DL/I engine inserts in the database using a path call.
- To insert a path of data under an existing parent segment, use a WHERE statement or scroll to the parent segment under which you want to insert the path of data. If there are no child segments under the parent segment, enter the path of data and press ENTER. The IMS-DL/I engine inserts the new segments under the existing parent segment. If child segments do exist, display one of the paths of data and type the new data over the old path of data, making sure that you change the key field value in the segments to be inserted. The IMS-DL/I engine then inserts the new segment.

If the view descriptor you are using does not include all the variables defined in the access descriptor for the segment to be inserted, low values (hexadecimal zeroes) are placed in those fields in the new segment occurrence inserted into the database. For more information on inserting segments when the SAS observations contain missing values, see “Handling Missing Values” on page 134 in Chapter 7, “Advanced User Topics for the SAS/ACCESS Interface View Engine for IMS-DL/I,” on page 129. Refer to *SAS/FSP Software: Usage and Reference* for more information on how to use the ADD and

DUP commands in the FSEDIT procedure and the AUTOADD and DUP commands in the FSVIEW procedure.

When the DELETE command is used while the FSEDIT or FSVIEW procedure is referencing a view descriptor, the lowest-level existing database segment referenced in the view descriptor is removed permanently from the IMS-DL/I database. Refer to *SAS/FSP Software: Usage and Reference* for more information on this command.

CAUTION:

If you delete segments using a view descriptor that references only the upper hierarchical level segments in the database, any children of these segments will also be deleted, even though those child segments are not included in the view descriptor. △

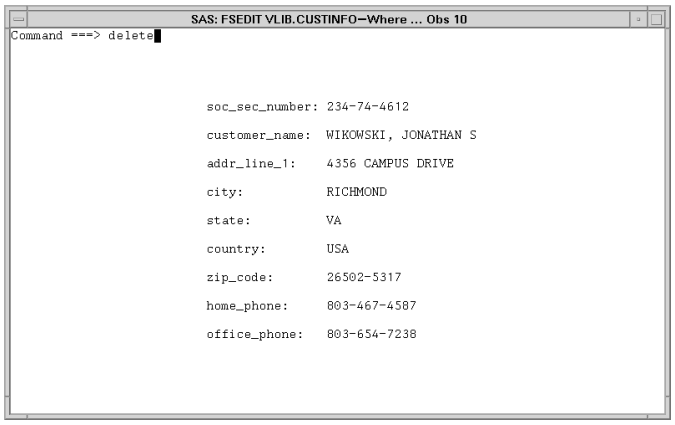
For example, consider a database consisting of a root segment, a child segment under the root, and another child segment under that child. If you delete a segment in that database using a view descriptor that references only the root and one child, the DELETE command will delete the entire path of data below the root segment. There are two ways you can delete an entire database record:

- Use the DELETE command with a view descriptor that references the root segment only.
- Use the DELETE command multiple times with a view descriptor that references an entire path of data in the database. Each time you use the DELETE command, only the lowest existing segment in the path is deleted.

See “Delete Processing” on page 146 in Chapter 7, “Advanced User Topics for the SAS/ACCESS Interface View Engine for IMS-DL/I,” on page 129 for more information on deleting segments.

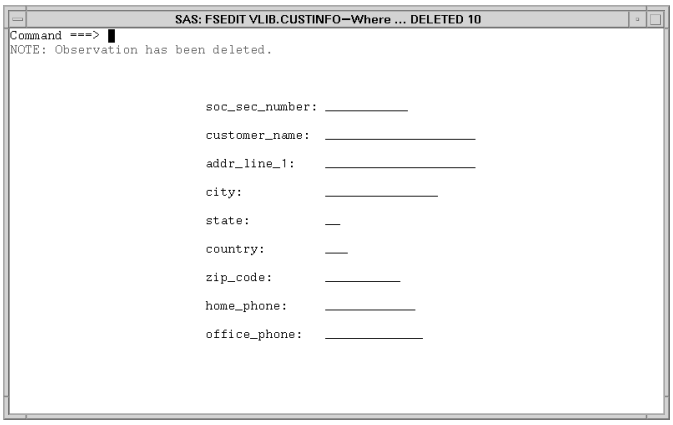
The following example illustrates how to use the DELETE command in the FSEDIT procedure. Suppose you want to edit the IMS-DL/I data described by VLIB.CUSTINFO to eliminate customers who have closed their bank accounts. If you are using a PCB that allows you delete authority, you can perform this function by using the FSEDIT procedure from the ACCESS window or with a PROC FSEDIT statement. Scroll forward to the observations to be deleted and enter **`DELETE`** on the command line, as shown in Display 5.6 on page 75.

Display 5.6 Deleting an IMS-DL/I Segment in an FSEDIT Window



The DELETE command deletes this root segment from the IMS-DL/I database described by VLIB.CUSTINFO and any child segments under it, and displays a message to that effect, as shown in Display 5.7 on page 76.

Display 5.7 Using the DELETE Command in the FSEDIT Window



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

Browsing and Updating with the SQL Procedure

The SQL procedure enables you to retrieve and update data from IMS-DL/I databases. You can retrieve and browse IMS-DL/I data by specifying a view descriptor in the SQL procedure's SELECT statement.

To update the data, you can specify view descriptors in the SQL procedure's INSERT, DELETE, and UPDATE statements. The view descriptor specified can access data from only one IMS-DL/I database path. You must use a PCB that allows you the appropriate level of access (insert, replace, delete, or all) for the segments that you want to update before you can edit the IMS-DL/I data.

The following list summarizes these SQL procedure statements:

SELECT retrieves, manipulates, and displays data from IMS-DL/I databases. A SELECT statement is usually referred to as a query because it queries the database for information.

DELETE deletes segments from an IMS-DL/I database.

INSERT inserts segments in an IMS-DL/I database.

UPDATE updates the data values in an IMS-DL/I database.

If you want to use the SQL procedure to join or access more than one IMS-DL/I database, you must use a PSB in your view descriptors that includes a PCB for each database to be accessed. Each view descriptor to be joined must use the same PSB. If you join two view descriptors that reference different paths in the same database, each view descriptor must reference in the PSB (that refers to the same database) a different PCB by using the **PCB Index** field. That is, to access the same database using different view descriptors in any SAS procedure, you must include multiple PCBs for that database.

When using PROC SQL, notice that the data are displayed in the SAS OUTPUT window in display manager mode and written to the SASLIST DDname in batch mode, interactive line mode, and noninteractive mode. This procedure displays output data automatically without the PRINT procedure and executes without a RUN statement when an SQL procedure statement is submitted.

Retrieving and Updating with the SQL Procedure

Note: The following PROC SQL examples assume the ACCTDBD database has not been updated by the earlier SAS/FSP examples. △

You can use the SELECT statement to browse IMS-DL/I data described by a view descriptor. The query in the following example retrieves all the observations in the IMS-DL/I ACCTDBD database that are described by the VLIB.CUSTINFO view descriptor.

```
options linesize=132;

proc sql;
title2 'IMS-DL/I Data Retrieved by a PROC SQL query';
select * /* An asterisk means select all variables */
   from vlib.custinfo;
```

The OPTIONS statement is used to reset the default output width to 132 columns. Output 5.1 on page 77 displays the query's output. Note that PROC SQL displays labels, which are the IMS-DL/I item names. In Version 7, the item names are also the SAS variable names, as shown here.

Output 5.1 IMS-DL/I Data Retrieved by a PROC SQL Query

```
                                        The SAS System
                                IMS-DL/I Data Retrieved by a PROC SQL query

SOC_SEC_
NUMBER        CUSTOMER_NAME                          ADDR_LINE_1                  ADDR_LINE_2
CITY                          STATE  COUNTRY            ZIP_CODE    HOME_PHONE    OFFICE_PHONE
-------------------------------------------------------------------------------------------------------------------
667-73-8275   WALLS, HOOPER J.                                                     4525 CLARENDON RD
RAPIDAN                       VA     USA                22215-5600  803-657-3098  803-645-4418

434-62-1234   SUMMERS, MARY T.                                                     4322 LEON ST.
GORDONSVILLE                  VA     USA                26001-0670  803-657-1687

436-42-6394   BOOKER, APRIL M.                                                     9712 WALLINGFORD PL.
GORDONSVILLE                  VA     USA                26001-0670  803-657-1346

434-62-1224   SMITH, JAMES MARTIN                                                  133 TOWNSEND ST.
GORDONSVILLE                  VA     USA                26001-0670  803-657-3437

178-42-6534   PATTILLO, RODRIGUES                                                  9712 COOK RD.
ORANGE                        VA     USA                26042-1650  803-657-1346  803-657-1345

156-45-5672   O'CONNOR, JOSEPH                                                     235 MAIN ST.
ORANGE                        VA     USA                26042-1650  803-657-5656  803-623-4257

657-34-3245   BARNHARDT, PAMELA S.                                                 RT 2 BOX 324
CHARLOTTESVILLE               VA     USA                25804-0997  803-345-4346  803-355-2543

667-82-8275   COHEN, ABRAHAM                                                       2345 DUKE ST.
CHARLOTTESVILLE               VA     USA                25804-0997  803-657-7435  803-645-4234

456-45-3462   LITTLE, NANCY M.                                                     4543 ELGIN AVE.
RICHMOND                      VA     USA                26502-3317  803-657-3566

234-74-4612   WIKOWSKI, JONATHAN S.                                                4356 CAMPUS DRIVE
RICHMOND                      VA     USA                26502-5317  803-467-4587  803-654-7238
```

You can specify a WHERE clause as part of the SQL procedure's SELECT statement to retrieve a subset of the database data. The following example displays a list of customers who have accounts with the Richmond branch of the bank:

```
title2 'IMS-DL/I Data Retrieved by a WHERE Statement';
   select *
      from vlib.custinfo
      where city='RICHMOND';
```

Notice that the PROC SQL statement is not repeated in this query. With the SQL procedure, you do not need to repeat the PROC SQL statement unless you submit another SAS procedure, a DATA step, or a QUIT statement between PROC SQL statements. Output 5.2 on page 78 displays the customers of the Richmond branch who are described by VLIB.CUSTINFO.

Output 5.2 IMS-DL/I Data Retrieved Using a WHERE Statement

```
                                    The SAS System
                          IMS-DL/I Data Retrieved Using a WHERE Statement

SOC_SEC_
NUMBER        CUSTOMER_NAME                          ADDR_LINE_1                  ADDR_LINE_2
CITY                        STATE  COUNTRY             ZIP_CODE    HOME_PHONE    OFFICE_PHONE
-------------------------------------------------------------------------------------------------------------------------
456-45-3462   LITTLE, NANCY M.                                                    4543 ELGIN AVE.
RICHMOND                    VA     USA                 26502-3317  803-657-3566

234-74-4612   WIKOWSKI, JONATHAN S.                                               4356 CAMPUS DRIVE
RICHMOND                    VA     USA                 26502-5317  803-467-4587  803-654-7238
```

You can use the UPDATE statement to update the data in an IMS-DL/I database as was done earlier in this chapter using the FSEDIT procedure. Remember that when you reference a view descriptor in an SQL procedure statement, you are updating the IMS-DL/I data described by the view descriptor, not the view descriptor itself. Use the WHERE clause to position the IMS-DL/I engine on the database segment to be updated by specifying values for the key fields of parent segments.

The following UPDATE statements update the values that are contained in the last observation of VLIB.CUSTINFO:

```
update vlib.custinfo
   set zip_code = '27702-3317'
   where soc_sec_number = '234-74-4612';

update vlib.custinfo
   set addr_line_2 = '151 Knox St.'
   where soc_sec_number = '234-74-4612';

title2 'Updated Data in IMS-DL/I ACCTDBD Database';
select *
  from vlib.custinfo
  where soc_sec_number = '234-74-4612';
```

The SELECT statement in this example retrieves and displays the updated data in Output 5.3 on page 78. (Because you are referencing a view descriptor, you use the SAS names for items in the UPDATE statement; the SQL procedure displays the variable labels as stored in the view.)

Output 5.3 IMS-DL/I Data Updated with the UPDATE Statement

```
                                        The SAS System
                                Updated Data in IMS-DL/I ACCTDBD Database

SOC_SEC_
NUMBER        CUSTOMER_NAME                          ADDR_LINE_1                    ADDR_LINE_2
CITY                        STATE  COUNTRY             ZIP_CODE    HOME_PHONE    OFFICE_PHONE
-------------------------------------------------------------------------------------------------------------------------
234-74-4612   WIKOWSKI, JONATHAN S.                                                  151 Knox St.
RICHMOND                    VA     USA                 27702-3317  803-467-4587  803-654-7238
```

Inserting and Deleting with the SQL Procedure

You can use the INSERT statement to add segments to an IMS-DL/I database or use the DELETE statement to remove segments from an IMS-DL/I database, as you did earlier in this chapter with the FSEDIT procedure. When inserting children under a parent segment, you must indicate the key values of the parent segments in the SET= statement. Use a view descriptor describing the entire path of data down to the lowest segment to be inserted. In the following example, the root segment that contains the value **234-74-4612** for the SOC_SEC_NUMBER variable is deleted from the ACCTDBD database. Note that any child segments that exist under the parent segment in this example will also be deleted.

```
options linesize=132;

  proc sql;
  delete from vlib.custinfo
     where soc_sec_number = '234-74-4612';

  title2 'Observation Deleted from IMS-DL/I
           ACCTDBD Database';
  select *
     from vlib.custinfo;
```

The SELECT statement then displays the data for VLIB.CUSTINFO in Output 5.4 on page 79.

Output 5.4 IMS-DL/I Data with an Observation Deleted

```
                                        The SAS System
                              Observation Deleted from IMS-DL/I ACCTDBD Database

SOC_SEC_
NUMBER        CUSTOMER_NAME                          ADDR_LINE_1                  ADDR_LINE_2
CITY                        STATE  COUNTRY             ZIP_CODE    HOME_PHONE    OFFICE_PHONE
-------------------------------------------------------------------------------------------------------------------
667-73-8275   WALLS, HOOPER J.                                                     4525 CLARENDON RD
RAPIDAN                     VA     USA                 22215-5600  803-657-3098  803-645-4418

434-62-1234   SUMMERS, MARY T.                                                     4322 LEON ST.
GORDONSVILLE                VA     USA                 26001-0670  803-657-1687

436-42-6394   BOOKER, APRIL M.                                                     9712 WALLINGFORD PL.
GORDONSVILLE                VA     USA                 26001-0670  803-657-1346

434-62-1224   SMITH, JAMES MARTIN                                                  133 TOWNSEND ST.
GORDONSVILLE                VA     USA                 26001-0670  803-657-3437

178-42-6534   PATTILLO, RODRIGUES                                                  9712 COOK RD.
ORANGE                      VA     USA                 26042-1650  803-657-1346  803-657-1345

156-45-5672   O'CONNOR, JOSEPH                                                     235 MAIN ST.
ORANGE                      VA     USA                 26042-1650  803-657-5656  803-623-4257

657-34-3245   BARNHARDT, PAMELA S.                                                 RT 2 BOX 324
CHARLOTTESVILLE             VA     USA                 25804-0997  803-345-4346  803-355-2543

667-82-8275   COHEN, ABRAHAM                                                       2345 DUKE ST.
CHARLOTTESVILLE             VA     USA                 25804-0997  803-657-7435  803-645-4234

456-45-3462   LITTLE, NANCY M.                                                     4543 ELGIN AVE.
RICHMOND                    VA     USA                 26502-3317  803-657-3566
```

CAUTION:

Use a WHERE clause in a DELETE statement in the SQL procedure. If you omit the WHERE clause from the DELETE statement in the SQL procedure, you delete the lowest level segment for each database path that is defined by the view descriptor in the IMS-DL/I database. If the view descriptor describes only the root segment, the entire database will be deleted. △

For more information on the SQL procedure, see the *SAS Guide to the SQL Procedure: Usage and Reference*.

Updating Data with the MODIFY Statement

The MODIFY statement extends the capabilities of the DATA step by enabling you to modify IMS-DL/I data accessed by a view descriptor or a SAS data file without creating an additional copy of the file. To use the MODIFY statement with a view descriptor, you must have update privileges defined in the PCB associated with the view, even if your program doesn't intend to modify the data.

You can specify either a view descriptor or a SAS data file as the data set to be opened for update by using the MODIFY statement. In the following example, the data set to be opened for update is the view descriptor VLIB.CUSTINFO, which describes data in the IMS-DL/I sample database ACCTDBD. See Appendix 2 for the code used to generate this view descriptor and the access descriptor MYLIB.ACCOUNT. The updates made to VLIB.CUSTINFO will be used to change the data in the ACCTDBD database.

In order to update VLIB.CUSTINFO, you create a SAS data set, MYDATA.PHONENUM, to supply transaction information.

The MODIFY statement updates the ACCTDBD database with data from the MYDATA.PHONENUM data set in the following example:

```
data vlib.custinfo
     work.phoneupd (keep=soc_sec_number home_phone
          office_phone)
     work.nossnumb (keep=soc_sec_number home_phone
          office_phone);
  modify vlib.custinfo mydata.phonenum;
  by soc_sec_number;
  select (_iorc_);
     when (%sysrc(_sok))
  /* soc_sec_number found in ACCTDBD        */
       do;
         replace vlib.custinfo;
         output phoneupd;
       end;
     when (%sysrc(_dsenmr))
  /* soc_sec_number not found in ACCTDBD    */
       do;
         _error_=0;
         output nossnumb;
  /* stores misses in NOSSNUMB      */
       end;
     otherwise
  /* traps unexpected outcomes      */
       do;
         put 'Unexpected error condition:
                _iorc_ = ' _iorc_;
         put 'for SOC_SEC_NUMBER=' soc_sec_number
             '. Data step continuing.';
         _error_=0;
       end;
  end;
run;
```

For each iteration of the DATA step, the SAS System attempts to read one observation (or record) of the ACCTDBD database as defined by VLIB.CUSTINFO, based on SOC_SEC_NUMBER values supplied by MYDATA.PHONENUM. If a match on SOC_SEC_NUMBER values is found, the current segment data in ACCTDBD are replaced with the updated information in MYDATA.PHONENUM, then SOC_SEC_NUMBER, HOME_PHONE and OFFICE_PHONE are stored in the PHONEUPD data file. If the SOC_SEC_NUMBER value supplied by MYDATA.PHONENUM has no match in VLIB.CUSTINFO, the transaction information is written to the data file NOSSNUMB.

To further understand this type of processing, be aware that for each iteration of the DATA step (that is, each execution of the MODIFY statement), MYDATA.PHONENUM is processed sequentially. For each iteration, the current value of SOC_SEC_NUMBER is used to attach a WHERE clause to a request for an observation from VLIB.CUSTINFO as defined by the view. The engine then tries to generate a retrieval request with a qualified SSA from the WHERE clause. If the engine generates a qualified SSA, a GET UNIQUE call is issued, and data defined by the view are accessed directly. If the engine cannot generate a qualified SSA from the WHERE clause, a

sequential pass of the database is required for each transaction observation in MYDATA.PHONENUM.

Print the PHONEUPD data file to see the SOC_SEC_NUMBER items that were updated. The output is shown in Output 5.5 on page 82:

```
/*  Print data set named phoneupd  */
proc print data=work.phoneupd nodate;
  title2 'SSNs updated.';
run;
```

Output 5.5 Contents of the PHONEUPD Data File

```
                          The SAS System
                          SSNs updated.

          SOC_SEC_
   OBS    NUMBER          HOME_PHONE      OFFICE_PHONE

   1     667-73-8275    703-657-3098    703-645-4418
   2     434-62-1234    703-645-441
   3     178-42-6534    703-657-1346    703-657-1345
   4     156-45-5672    703-657-5656    703-623-4257
   5     657-34-3245    703-345-4346    703-355-5438
   6     456-45-3462    703-657-3566    703-645-1212
```

Print the NOSSNUMB data set to see the SOC_SEC_NUMBER items that were not updated. The output produced by the following code is shown in Output 5.6 on page 82:

```
/*  Print data set named nossnumb  */
proc print data=work.nossnumb nodate;
   title2 'SSNs not updated.';
run;
```

Output 5.6 Contents of the NOSSUNUMB Data File

```
                        The SAS System
                        SSNs not updated.

           SOC_SEC_
      OBS  NUMBER          HOME_PHONE      OFFICE_PHONE

      1   416-41-3162    703-657-3166    703-615-1212
```

Updating SAS Files with IMS-DL/I Data

You can update a SAS data file or data set with IMS-DL/I data that are described by a view descriptor just as you can update a SAS data file with data from another SAS data file.

Suppose you have a SAS data set, MYDATA.BIRTHDAY, that contains employee ID numbers, last names, and birthdays. (See Appendix 2 for a description of MYDATA.BIRTHDAY.) You want to update this data set with data described by

VLIB.EMPBDAY, a view descriptor that is based on the IMS-DL/I EMPLINF2 database. To perform this update, enter the following SAS statements:

```
libname vlib 'sas-data-library';
libname mydata 'sas-data-library';
options nodate;

/*-------------------------------------------*/
/*  Update the BIRTHDAY SAS data set         */
/*  with data from IMS-DL/I                  */
/*  EMPLINF2 database                        */
/*-------------------------------------------*/
options linesize=80;
proc sort data=mydata.birthday;
  by employee_id;
run;

proc print data=mydata.birthday;
  title2 'Sorted SAS Data Set MYDATA.BIRTHDAY';
run;

proc print data=vlib.empbday;
  title2 'Data Described by VLIB.EMPBDAY';
run;

data mydata.newbday;
  update mydata.birthday vlib.empbday;
  by employee_id;
run;

proc print data=mydata.newbday;
  title2 'SAS Data Set MYDATA.NEWBDAY';
run;
```

The EMPLINF2 database is a HIDAM database whose root segment is sequenced by the key field EMPID, so when the UPDATE statement references the view descriptor VLIB.EMPBDAY, the data is presented to the SAS System for updating in sorted order by EMPLOYEE_ID. However, the SAS data set MYDATA.BIRTHDAY must be sorted before the update because the UPDATE statement expects both the original file and the transaction file to be sorted by the same BY variable.

Output 5.7 on page 83, Output 5.8 on page 84, and Output 5.9 on page 84 show the results of the print procedures.

Output 5.7 Data Set to be Updated, MYDATA.BIRTHDAY, in EMPID Order

```
                The SAS System
       Sorted SAS Data Set MYDATA.BIRTHDAY

          employee_
  OBS        id        last_name     birthday

  1         1005       Knapp         06OCT38
  2         1024       Mueller       17JUN53
  3         1078       Gibson        23APR36
  4         1247       Garcia        04APR54
```

Output 5.8 IMS-DL/I Data Described by the View Descriptor VLIB.EMPBDAY

The SAS System
Data Described by VLIB.EMPBDAY

OBS	EMPLOYEE_ ID	LAST_NAME	FIRST_NAME	BIRTHDAY	PHONE_ EXTENSION
1	1001	Waterhouse	Clifton P.	01JAN48	X5109
2	1002	Bowman	Hugh E.	14JUL31	X5901
3	1003	Salazar	Yolanda	12DEC40	X5169
4	1004	Knight	Althea	09APR50	X5218
5	1005	Knapp	Patrice R.	04OCT37	X5012
6	1006	Garrett	Olan M.	23JAN35	X5208
7	1007	Brown	Virgina P.	24MAY46	X5258
8	1008	Hernandez	Jesse L.	26MAR33	X5448
9	1009	Jones	Michael Y.	21MAY31	X5713
10	1010	Smith	Janet F.	07AUG47	X5621
11	1011	Van Hotten	Gwendolyn	13SEP42	X5311
12	1012	Quintero	Pedro	21FEB48	X5348
13	1015	Scholl	Madison A.	19MAR45	X5419
14	1017	Waggonner	Merrilee D.	27APR36	X5914
15	1020	Rudd	Fred	.	
16	1024	Mueller	Patsy	17JUN52	X5822
17	1031	Chan	Tai	04JUL46	X5331
18	1049	Fernandez	Sophia	11SEP44	X5847
19	1050	Ameer	David	10OCT51	X5495
20	1062	Littlejohn	Fannie	17MAY54	X5653
21	1067	Cahill	Jacob	25DEC40	X5042
22	1071	Canady	Frank A.	19NOV41	X5406
23	1074	Millsap	Joel B.	12JUN36	X5224
24	1077	Gibson	Teddy B.	23APR46	X5703
25	1078	Gibson	George J.	23APR46	X5703
26	1083	Savage	William D.	20JAN53	X5505
27	1086	Schmidt	Penny	19FEB27	X5822
28	1092	Polanski	Ivan L.	11OCT47	X5621
29	1101	Nathaniel	Darryl	21MAR44	X5544
30	1105	Faulkner	Carrie Ann	17AUG51	X5417
31	1112	Jones	Rita M.	24DEC48	X5271
32	1119	Goodson	Alan F.	21JUN50	X5512
33	1120	Reid	David G.	15AUG45	X5369
34	1123	Freeman	Leopold	09FEB35	X5604
35	1133	Williamson	Janice L.	19MAY52	X5802
36	1139	Seaton	Gary	03OCT56	X5545
37	1145	Juarez	Armando	28MAY47	X5987
38	1156	Reed	Kenneth D.	05JAN55	X5307
39	1161	Richardson	Travis Z.	30NOV37	X5325
40	1213	Johnson	Bradford	15APR54	X5446
41	1217	Rodriguez	Romualdo R.	09FEB29	X5874
42	1219	Kaatz	Freddie	21JUN57	X5387
43	1234	Shropshire	Leland G.	04SEP49	X5616
44	1238	Throckmort	Stewart Q.	04AUG31	X5391
45	1247	Garcia	Francisco	05MAY55	X5348
46	1261	Collins	Lillian	01MAY51	X5616
47	1265	Slye	Leonard R.	18DEC60	X5123
48	1266	Redfox	Richard B.	04APR44	X5386
49	1272	Smith	Garland P.	05APR54	X5415
50	1313	Smith	Jerry Lee	13SEP42	X5169
51	1327	Brooks	Ruben R.	25FEB52	X5347
52	1900	Smith	John	.	

Output 5.9 Data in the New Data Set MYDATA.NEWBDAY

The SAS System
SAS Data Set MYDATA.NEWBDAY

OBS	employee_ id	last_name	birthday	FIRST_NAME	PHONE_ EXTENSION
1	1001	Waterhouse	01JAN48	Clifton P.	X5109
2	1002	Bowman	14JUL31	Hugh E.	X5901
3	1003	Salazar	12DEC40	Yolanda	X5169
4	1004	Knight	09APR50	Althea	X5218
5	1005	Knapp	04OCT37	Patrice R.	X5012
6	1006	Garrett	23JAN35	Olan M.	X5208
7	1007	Brown	24MAY46	Virgina P.	X5258
8	1008	Hernandez	26MAR33	Jesse L.	X5448
9	1009	Jones	21MAY31	Michael Y.	X5713
10	1010	Smith	07AUG47	Janet F.	X5621
11	1011	Van Hotten	13SEP42	Gwendolyn	X5311
12	1012	Quintero	21FEB48	Pedro	X5348
13	1015	Scholl	19MAR45	Madison A.	X5419
14	1017	Waggonner	27APR36	Merrilee D.	X5914
15	1020	Rudd	.	Fred	
16	1024	Mueller	17JUN52	Patsy	X5822
17	1031	Chan	04JUL46	Tai	X5331
18	1049	Fernandez	11SEP44	Sophia	X5847
19	1050	Ameer	10OCT51	David	X5495
20	1062	Littlejohn	17MAY54	Fannie	X5653
21	1067	Cahill	25DEC40	Jacob	X5042
22	1071	Canady	19NOV41	Frank A.	X5406
23	1074	Millsap	12JUN36	Joel B.	X5224
24	1077	Gibson	23APR46	Teddy B.	X5703
25	1078	Gibson	23APR46	George J.	X5703
26	1083	Savage	20JAN53	William D.	X5505
27	1086	Schmidt	19FEB27	Penny	X5822
28	1092	Polanski	11OCT47	Ivan L.	X5621
29	1101	Nathaniel	21MAR44	Darryl	X5544
30	1105	Faulkner	17AUG51	Carrie Ann	X5417
31	1112	Jones	24DEC48	Rita M.	X5271
32	1119	Goodson	21JUN50	Alan F.	X5512
33	1120	Reid	15AUG45	David G.	X5369
34	1123	Freeman	09FEB35	Leopold	X5604
35	1133	Williamson	19MAY52	Janice L.	X5802
36	1139	Seaton	03OCT56	Gary	X5545
37	1145	Juarez	28MAY47	Armando	X5987
38	1156	Reed	05JAN55	Kenneth D.	X5307
39	1161	Richardson	30NOV37	Travis Z.	X5325
40	1213	Johnson	15APR54	Bradford	X5446
41	1217	Rodriguez	09FEB29	Romualdo R.	X5874
42	1219	Kaatz	21JUN57	Freddie	X5387
43	1234	Shropshire	04SEP49	Leland G.	X5616
44	1238	Throckmort	04AUG31	Stewart Q.	X5391
45	1247	Garcia	05MAY55	Francisco	X5348
46	1261	Collins	01MAY51	Lillian	X5616
47	1265	Slye	18DEC60	Leonard R.	X5123
48	1266	Redfox	04APR44	Richard B.	X5386
49	1272	Smith	05APR54	Garland P.	X5415
50	1313	Smith	13SEP42	Jerry Lee	X5169
51	1327	Brooks	25FEB52	Ruben R.	X5347
52	1900	Smith	.	John	

Appending Data with the APPEND Procedure

You can append data described by SAS/ACCESS view descriptors and PROC SQL views to SAS data files and vice versa. You can also append data from one view descriptor to the data from another.

In the following example, two branch managers have kept separate records on customers' checking accounts. One manager has kept records in the CUSTOMER and CHCKACCT segments of the IMS-DL/I database ACCTDBD, described by the view descriptor VLIB.CHCKACCT. The other manager has kept records in a Version 7 SAS data set, MYDATA.CHECKS. Due to a corporate reorganization, the two sources of data must be combined so that all customer data are stored in the IMS-DL/I database ACCTDBD. A branch manager can use the APPEND procedure to perform this task, as the following example demonstrates.

The data described by the VLIB.CHCKACCT view descriptor and the data in the SAS data set MYDATA.CHECKS are displayed in Output 5.10 on page 86 and Output 5.11 on page 86.

```
options linesize=120;

   proc print data=vlib.chckacct;
      title2 'Data Described by VLIB.CHCKACCT';
   run;

   proc print data=mydata.checks;
      title2 'Data in MYDATA.CHECKS Data Set';
   run;
```

Note: To use PROC APPEND, you must use a view descriptor that describes the entire path of data from the root segment down to the level where you want to append data. If a parent segment already exists with a key value equal to that specified in the input data set, the IMS-DL/I engine inserts the remaining path of data under the parent segment. △

Output 5.10 Data Described by the VLIB.CHCKACCT View Descriptor

The SAS System
Data Described by VLIB.CHCKACCT

OBS	SOC_SEC_ NUMBER	CUSTOMER_NAME	CHECK_ACCOUNT_ NUMBER	CHECK_ DATE	CHECK_ BALANCE
1	667-73-8275	WALLS, HOOPER J.	345620145345	15MAR95	1266.34
2	667-73-8275	WALLS, HOOPER J.	345620154633	28MAR95	1298.04
3	434-62-1234	SUMMERS, MARY T.	345620104732	27MAR95	825.45
4	436-42-6394	BOOKER, APRIL M.	345620135872	26MAR95	234.89
5	434-62-1224	SMITH, JAMES MARTIN	345620134564	16MAR95	2645.34
6	434-62-1224	SMITH, JAMES MARTIN	345620134663	24MAR95	143.78
7	178-42-6534	PATTILLO, RODRIGUES	745920057114	10JUN95	1502.78
8	156-45-5672	O'CONNOR, JOSEPH	345620123456	27MAR95	463.23
9	657-34-3245	BARNHARDT, PAMELA S.	345620131455	29MAR95	1243.25
10	667-82-8275	COHEN, ABRAHAM	382957492811	03APR95	7302.06
11	456-45-3462	LITTLE, NANCY M.	345620134522	25MAR95	831.65

Output 5.11 Data in the MYDATA.CHECKS Data Set

The SAS System
Data in MYDATA.CHECKS Data Set

OBS	customer_name	soc_sec_ number	check_ account_ number	check_ balance	check_ date
1	COWPER, KEITH	241-98-4542	183352795865	862.31	25MAR95
2	OLSZEWSKI, STUART	309-22-4573	382654397566	486.00	02APR95
3	NAPOLITANO, BARBARA	250-36-8831	284522378774	104.20	10APR95
4	MCCALL, ROBERT	367-34-1543	644721295973	571.92	05APR95

The manager can combine the data from these two sources using the APPEND procedure, as shown in the following example:

```
proc append base=vlib.chckacct data=mydata.checks;
   run;

   proc print data=vlib.chckacct;
     title2 'Appended Data';
   run;

   proc sql;
     delete from vlib.account
     where soc_sec_number in( '241--98--4542'
                              '250--36--8831'
                              '309--22--4573'
                              '367--34--1543' )
   run;
```

The database type determines where the segments are inserted. In this case, the database type is not an indexed database type, so the data in MYDATA.CHECKS are intermixed with the data described by VLIB.CHCKACCT. Output 5.12 on page 87 displays the updated data described by the view descriptor, VLIB.CHCKACCT.

Output 5.12 Appended Data

The SAS System
Appended Data

OBS	SOC_SEC_ NUMBER	CUSTOMER_NAME	CHECK_ACCOUNT_ NUMBER	CHECK_ DATE	CHECK_ BALANCE
1	667-73-8275	WALLS, HOOPER J.	345620145345	15MAR95	1266.34
2	667-73-8275	WALLS, HOOPER J.	345620154633	28MAR95	1298.04
3	434-62-1234	SUMMERS, MARY T.	345620104732	27MAR95	825.45
4	250-36-8831	NAPOLITANO, BARBARA	284522378774	10APR95	104.20
5	241-98-4542	COWPER, KEITH	183352795865	25MAR95	862.31
6	436-42-6394	BOOKER, APRIL M.	345620135872	26MAR95	234.89
7	434-62-1224	SMITH, JAMES MARTIN	345620134564	16MAR95	2645.34
8	434-62-1224	SMITH, JAMES MARTIN	345620134663	24MAR95	143.78
9	178-42-6534	PATTILLO, RODRIGUES	745920057114	10JUN95	1502.78
10	367-34-1543	MCCALL, ROBERT	644721295973	05APR95	571.92
11	156-45-5672	O'CONNOR, JOSEPH	345620123456	27MAR95	463.23
12	657-34-3245	BARNHARDT, PAMELA S.	345620131455	29MAR95	1243.25
13	667-82-8275	COHEN, ABRAHAM	382957492811	03APR95	7302.06
14	456-45-3462	LITTLE, NANCY M.	345620134522	25MAR95	831.65
15	309-22-4573	OLSZEWSKI, STUART	382654397566	02APR95	486.00

Note: The APPEND procedure issues a warning message when a variable in the view descriptor does not have a corresponding variable in the input data set. △

The PROC SQL code deletes the appended data so the next PROC APPEND example will work without reinitializing the database.

You can use the APPEND procedure's FORCE option to force PROC APPEND to concatenate two data sets that have different variables or variable attributes.

The APPEND procedure also accepts a SAS WHERE statement to retrieve a subset of the data. In the following example, a subset of the observations from the DATA= data set is added to the BASE= data set.

```
proc append base=vlib.chckacct data=mydata.checks
  (where=(check_date >='26MAR95'd));
run;

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

Note that the WHERE data set option applies only to the DATA= data set. Output 5.13 on page 88 displays the results.

Output 5.13 Appended Data with a WHERE Data Set Option

The SAS System
Appended Data with a WHERE= Data Set Option

OBS	SOC_SEC_ NUMBER	CUSTOMER_NAME	CHECK_ACCOUNT_ NUMBER	CHECK_ DATE	CHECK_ BALANCE
1	667-73-8275	WALLS, HOOPER J.	345620145345	15MAR95	1266.34
2	667-73-8275	WALLS, HOOPER J.	345620154633	28MAR95	1298.04
3	434-62-1234	SUMMERS, MARY T.	345620104732	27MAR95	825.45
4	250-36-8831	NAPOLITANO, BARBARA	284522378774	10APR95	104.20
5	436-42-6394	BOOKER, APRIL M.	345620135872	26MAR95	234.89
6	434-62-1224	SMITH, JAMES MARTIN	345620134564	16MAR95	2645.34
7	434-62-1224	SMITH, JAMES MARTIN	345620134663	24MAR95	143.78
8	178-42-6534	PATTILLO, RODRIGUES	745920057114	10JUN95	1502.78
9	367-34-1543	MCCALL, ROBERT	644721295973	05APR95	571.92
10	156-45-5672	O'CONNOR, JOSEPH	345620123456	27MAR95	463.23
11	657-34-3245	BARNHARDT, PAMELA S.	345620131455	29MAR95	1243.25
12	667-82-8275	COHEN, ABRAHAM	382957492811	03APR95	7302.06
13	456-45-3462	LITTLE, NANCY M.	345620134522	25MAR95	831.65
14	309-22-4573	OLSZEWSKI, STUART	382654397566	02APR95	486.00

Note that the IMS-DL/I engine has no way to determine how large a database is. Therefore, if you use the APPEND procedure to add a database to itself, a loop can result. For more information on the APPEND procedure, see "The APPEND Procedure" in the *SAS Procedures Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Interface to IMS-DL/I Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. 316 pp.

SAS/ACCESS® Interface to IMS-DL/I Software: Reference, Version 8

ISBN 1–58025–548–5

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

1st printing, October 1999