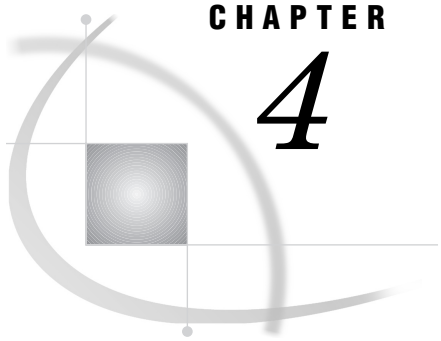


CHAPTER

4

Using IMS-DL/I Data in SAS Programs

Introduction

An advantage of the SAS/ACCESS to IMS-DL/I interface view engine is that it enables the SAS System to read and write IMS-DL/I data directly from SAS programs without having to code DL/I calls. This chapter presents examples using IMS-DL/I data described by view descriptors as input data for SAS programs. Throughout the examples, the SAS terms *variable* and *observation* are used instead of the IMS-DL/I terms *field* and *segment* because this chapter illustrates using SAS procedures and the DATA step. The examples include charting data using the Version 7 SQL procedure to combine data from various sources, and updating a Version 6 SAS data file with data from IMS-DL/I.

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 information on assigning passwords.

Appendix 2 includes definitions of all the view descriptors referenced in this chapter. Appendix 2 also includes the IMS-DL/I database data, SAS data files, and a DB2 table used in some of the examples.

Charting Data

GCHART procedure programs work with data described by view descriptors just as they do with other SAS data sets. The following example creates a horizontal bar chart

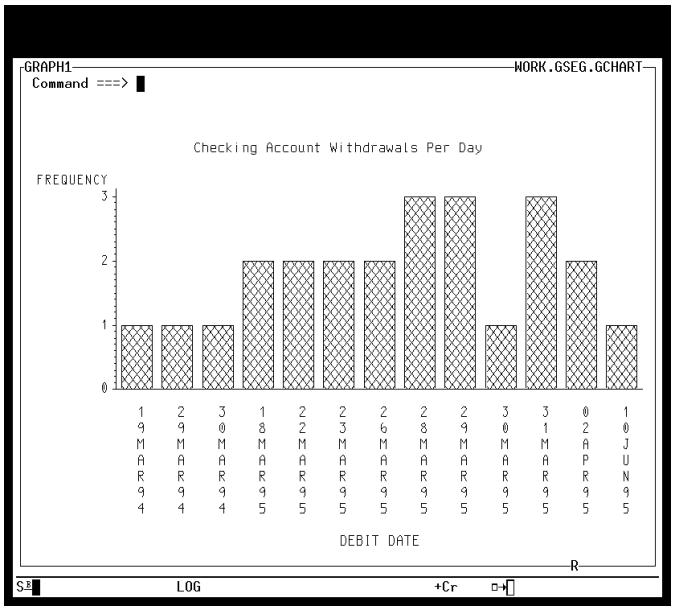
of the number of checking account withdrawals per day. This example uses the view descriptor VLIB.CDBTDATE to describe the CHCKDEBT segment of the ACCTDBD database:

```
options nodate linesize=132;
goptions device=chardrvw;

proc gchart data=vlib.cdbtdate;
   vbar check_date / discrete;
   title2 'Checking Account Withdrawals Per Day';
run;
```

Display 4.1 on page 48 shows the output for this example. CDBTDATE represents the date of each checking account withdrawal; the number of checking account withdrawals is represented by the length of the bar. For more information on the GCHART procedure, see *SAS Language Reference: Concepts* and the *SAS Procedures Guide*.

Display 4.1 Vertical Bar Chart of Checking Account Withdrawals



If you have SAS/GRAPH software, you can create colored block charts, plots, and other graphics based on IMS-DL/I data. See *SAS/GRAPH Software: Reference* for more information on the kinds of graphics that you can produce with this SAS product.

Calculating Statistics

You can also execute statistical procedures using IMS-DL/I data. This section shows examples using the FREQ, MEANS, and RANK procedures.

Using the FREQ Procedure

Suppose you want to find the percentages of your accounts in each city where you have a bank so that you can decide where to increase your marketing. The following

example calculates the percentages of customers for each city appearing in the IMS-DL/I database ACCTDBD using the view descriptor VLIB.CUSTINFO:

```
options nodate linesize=80;

proc freq data=vlib.custinfo;
   table city;
   title2 'Cities in the ACCTDBD Database';
run;
```

Output 4.1 on page 49 shows the one-way frequency table that this example generates.

Output 4.1 Frequency Table for Variable CITY

The SAS System
Cities in the ACCTDBD Database

CITY

CITY	Frequency	Percent	Cumulative Frequency	Cumulative Percent
CHARLOTTESVILLE	2	20.0	2	20.0
GORDONSVILLE	3	30.0	5	50.0
ORANGE	2	20.0	7	70.0
RAPIDAN	1	10.0	8	80.0
RICHMOND	2	20.0	10	100.0

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

Using the MEANS Procedure

In your analysis of recent accounts, suppose that you also want to determine some statistics by customer. In the following example, PROC MEANS is used to generate the mean debit amount for each customer (including the number of observations (N) and the number of missing values (NMISS)):

```
proc sort data=vlib.trans out=mydata.trandata;
   by soc_sec_number;
run;

options nodate linesize=80;

proc means data=mydata.trandata mean
   sum n nmiss maxdec=0;
   by soc_sec_number;
   var check_debit_amount;
   title2 'Mean Debit Amount Per Customer';
run;
```

Output 4.2 on page 50 shows the output for this example.

In the example, the view descriptor VLIB.TRANS selects CUSTOMER, CHCKACCT, and CHCKDEBT segment data from the IMS-DL/I database ACCTDBD. Since the ACCTDBD database is an HDAM and therefore is not indexed, the data described by the view descriptor must be sorted before using PROC MEANS. The sorted data are

stored in a SAS data file called MYDATA.TRANDATA, which is then used as input to PROC MEANS.

If your database is indexed, you can use a SAS BY statement for the indexed field so that data from this database are returned as if they were sorted. Database access methods HIDAM, HISAM, and SHISAM are indexed. If your database is not indexed, you need to sort the IMS-DL/I data before using the MEANS procedure with a BY statement. Because you cannot sort data in an IMS-DL/I database, you must use the OUT= option to extract data from the database so that you can pass it to the MEANS procedure. Since the ACCTDBD database is an HDAM and therefore is not indexed, the data described by the view descriptor must be sorted before using PROC MEANS with a BY statement for SOC_SEC_NUMBER.

Note: You can store the sorted data in a temporary data set if space is a concern. △

Note: If the view descriptor describes a path of data that includes segments from multiple hierarchical levels, the parent segment information is repeated for each SAS observation. This can cause misleading statistical results. To avoid misleading results, perform mathematical operations using only the data in the segment at the lowest hierarchical level. You can also avoid misleading results by creating a view descriptor that describes only the data in the segment at the lowest hierarchical level. △

Output 4.2 Statistics on Customer Debit Amounts

```
                          The SAS System
                    Mean Debit Amount Per Customer

-------------------------- SOC_SEC_NUMBER=156-45-5672 --------------------------

                          The MEANS Procedure

         Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

                                                        N
                  Mean             Sum     N    Miss
                ------------------------------------------
                    27             110     4        0
                ------------------------------------------


-------------------------- SOC_SEC_NUMBER=178-42-6534 --------------------------

         Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

                                                        N
                  Mean             Sum     N    Miss
                ------------------------------------------
                    26              26     1        0
                ------------------------------------------


-------------------------- SOC_SEC_NUMBER=234-74-4612 --------------------------

         Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

                                                        N
                  Mean             Sum     N    Miss
                ------------------------------------------
                     .               .     0        1
                ------------------------------------------


-------------------------- SOC_SEC_NUMBER=434-62-1224 --------------------------

         Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

                                                        N
                  Mean             Sum     N    Miss
                ------------------------------------------
                   162            1620    10        0
                ------------------------------------------
```

The SAS System

Mean Debit Amount Per Customer

---------- SOC_SEC_NUMBER=434-62-1234 ----------

The MEANS Procedure

Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

Mean	Sum	N	N Miss
.	.	0	1

---------- SOC_SEC_NUMBER=436-42-6394 ----------

Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

Mean	Sum	N	N Miss
.	.	0	1

---------- SOC_SEC_NUMBER=456-45-3462 ----------

Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

Mean	Sum	N	N Miss
66	263	4	0

---------- SOC_SEC_NUMBER=657-34-3245 ----------

Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

Mean	Sum	N	N Miss
.	.	0	1

---------- SOC_SEC_NUMBER=667-73-8275 ----------

The MEANS Procedure

Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

Mean	Sum	N	N Miss
355	1065	3	2

---------- SOC_SEC_NUMBER=667-82-8275 ----------

Analysis Variable : CHECK_DEBIT_AMOUNT CHECK_DEBIT_AMOUNT

Mean	Sum	N	N Miss
.	.	0	1

For more information on PROC MEANS, see *SAS Language Reference: Concepts* and the *SAS Procedures Guide*.

Using the RANK Procedure

You can also use more advanced statistical procedures on IMS-DL/I data. The following example uses the RANK procedure to rank checking account deposits by amount. It also assigns the variable name CRDRANK to the new item created by the RANK procedure, extracts and sorts the data, and prints the sorted output data. The view descriptor VLIB.CREDITS describes the CUSTOMER, CHCKACCT, and CHCKCRDT segments in the ACCTDBD database.

```
proc rank data=vlib.credits out=mydata.rankcred;
   var check_credit_amount;
   ranks crdrank;
run;

proc sort data=mydata.rankcred;
   by crdrank;
run;

options nodate linesize=132;

proc print data=mydata.rankcred;
   title2 'Deposits in Ascending Order';
run;
```

Output 4.3 on page 53 shows the result of this example.

Output 4.3 Ranking of Checking Account Balances

The SAS System
Deposits in Ascending Order

OBS	SOC_SEC_ NUMBER	CHECK_ACCOUNT_ NUMBER	CHECK_ CREDIT_ AMOUNT	CHECK_ CREDIT_ DATE	CHECK_ CREDIT_ TIME	CHECK_ CREDIT_ DESC
1	436-42-6394	345620135872	50.00	02APR95	12:16:34	ACH DEPOSIT
2	456-45-3462	345620134522	50.00	05APR95	12:14:52	ACH DEPOSIT
3	156-45-5672	345620123456	100.00	01APR95	12:24:34	ATM DEPOSIT
4	667-82-8275	382957492811	100.00	16APR95	09:21:14	ACH DEPOSIT
5	434-62-1224	345620134663	120.00	28MAR95	10:26:45	ACH DEPOSIT
6	657-34-3245	345620131455	230.00	04APR95	14:24:11	ACH DEPOSIT
7	434-62-1234	345620104732	400.00	02APR95	10:23:46	ACH DEPOSIT
8	234-74-4612	345620113263	672.32	31MAR95		ATM DEPOSIT
9	178-42-6534	745920057114	1300.00	12JUN95	14:34:12	ACH DEPOSIT
10	434-62-1224	345620134564	1342.42	22MAR95	23:23:52	ACH DEPOSIT
11	667-73-8275	345620145345	1563.23	31MAR95	15:42:43	MAIN ST BRANCH DEPOSIT
12	667-73-8275	345620154633	1563.23	31MAR95	15:42:43	BAD ACCT_NUM

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

Selecting and Combining Data

A great majority of SAS programs select and combine data from various sources. The method you use depends on the configuration of the data. The next three examples show you how to select and combine data using two different methods: the SET statement used in a DATA step and the SQL procedure. When choosing between these methods, you should first read the performance considerations discussed in Chapter 7, "Advanced User Topics for the SAS/ACCESS Interface View Engine for IMS-DL/I," on page 129.

Using the WHERE Statement

Suppose you had two view descriptors, VLIB.CHKCRD and VLIB.CHKDEB, that contain information about the checking accounts of customers. The view descriptor VLIB.CHKCRD describes the checking credit data in the CUSTOMER, CHCKACCT, and CHCKCRDT segments, and the view descriptor VLIB.CHKDEB describes the checking debit data in the CUSTOMER, CHCKACCT, and CHCKDEBT segments. You could use the SET statement to concatenate the data in these files and create a SAS data file that contains information on checking account transactions by customer. Since you are accessing the same database more than once, you need to reference the same PSB in both view descriptors, but use different PCB index values, where each value references an ACCTDBD PCB that is sensitive to the segments defined in the view. In this example, VLIB.CHKCRD uses a PCB index value of 2, and VLIB.CHKDEB uses a PCB index value of 3 in the ACCUPSB PSB.

The PROC SORT statement orders the accounts by Social Security number and checking account number.

```
data chktrans (keep=soc_sec_number
  check_account_number trantype date amount);
   length trantype $ 6;
   format date date9. amount dollar12.2;
   set vlib.chkcrd(in=crd) vlib.chkdeb(in=dbt);
   where check_balance>0;
   if crd then do;
      trantype='Credit';
      date=check_credit_date;
      amount=check_credit_amount;
   end;
   else if dbt then do;
      trantype='Debit';
      date=check_debit_date;
      amount=check_debit_amount;
   end
run;

proc sort;
   by soc_sec_number check_account_number;
run;

options nodate linesize=80;

proc print data=chktrans;
   by soc_sec_number;
   var check_account_number trantype date amount;
```

```
title2 'Checking Account Transactions by SSN';
run;
```

In the SAS WHERE statement, be sure to use the IMS-DL/I item name as the search criteria when VALIDVARNAME=V7 and the SAS variable name when VALIDVARNAME=V6. This is a Version 7 example. Output 4.4 on page 56 shows the result of the new temporary SAS data file WORK.CHKTRANS.

Output 4.4 WORK.CHKTRANS Data File Created Using a SAS WHERE Statement

Checking Account Transactions by SSN

SOC_SEC_NUMBER=156-45-5672

OBS	CHECK_ACCOUNT_NUMBER	TRANTYPE	DATE	AMOUNT
1	345620123456	Credit	01APR1991	$100.00
2	345620123456	Debit	28MAR1991	$13.29
3	345620123456	Debit	31MAR1991	$32.87
4	345620123456	Debit	02APR1991	$50.00
5	345620123456	Debit	31MAR1991	$13.42

SOC_SEC_NUMBER=178-42-6534

OBS	CHECK_ACCOUNT_NUMBER	TRANTYPE	DATE	AMOUNT
6	745920057114	Credit	12JUN1991	$1,300.00
7	745920057114	Debit	10JUN1991	$25.89

SOC_SEC_NUMBER=234-74-4612

OBS	CHECK_ACCOUNT_NUMBER	TRANTYPE	DATE	AMOUNT
8	345620113263	Credit	31MAR1991	$672.32
9	345620113263	Debit	.	.

SOC_SEC_NUMBER=434-62-1224

OBS	CHECK_ACCOUNT_NUMBER	TRANTYPE	DATE	AMOUNT
10	345620134564	Credit	22MAR1991	$1,342.42
11	345620134564	Debit	18MAR1991	$432.87
12	345620134564	Debit	18MAR1991	$19.23
13	345620134564	Debit	22MAR1991	$723.23
14	345620134564	Debit	22MAR1991	$82.32
15	345620134564	Debit	26MAR1991	$73.62
16	345620134564	Debit	26MAR1991	$31.23
17	345620134564	Debit	29MAR1990	$162.87
18	345620134564	Debit	29MAR1991	$7.12
19	345620134564	Debit	31MAR1991	$62.34
20	345620134663	Credit	28MAR1991	$120.00
21	345620134663	Debit	28MAR1991	$25.00

SOC_SEC_NUMBER=434-62-1234

OBS	CHECK_ACCOUNT_NUMBER	TRANTYPE	DATE	AMOUNT
22	345620104732	Credit	02APR1991	$400.00
23	345620104732	Debit	.	.

Checking Account Transactions by SSN

SOC_SEC_NUMBER=436-42-6394

OBS	CHECK_ACCOUNT_ NUMBER	TRANTYPE	DATE	AMOUNT
24	345620135872	Credit	02APR1991	$50.00
25	345620135872	Debit	30MAR1990	.

SOC_SEC_NUMBER=456-45-3462

OBS	CHECK_ACCOUNT_ NUMBER	TRANTYPE	DATE	AMOUNT
26	345620134522	Credit	05APR1991	$50.00
27	345620134522	Debit	29MAR1991	$42.73
28	345620134522	Debit	29MAR1991	$172.45
29	345620134522	Debit	30MAR1991	$38.23
30	345620134522	Debit	02APR1991	$10.00

SOC_SEC_NUMBER=657-34-3245

OBS	CHECK_ACCOUNT_ NUMBER	TRANTYPE	DATE	AMOUNT
31	345620131455	Credit	04APR1991	$230.00
32	345620131455	Debit	.	.

SOC_SEC_NUMBER=667-73-8275

OBS	CHECK_ACCOUNT_ NUMBER	TRANTYPE	DATE	AMOUNT
33	345620145345	Credit	31MAR1991	$1,563.23
34	345620145345	Debit	19MAR1990	.
35	345620145345	Debit	23MAR1991	$820.00
36	345620145345	Debit	23MAR1991	$52.00
37	345620145345	Debit	28MAR1991	$193.00
38	345620154633	Credit	31MAR1991	$1,563.23
39	345620154633	Debit	.	.

SOC_SEC_NUMBER=667-82-8275

OBS	CHECK_ACCOUNT_ NUMBER	TRANTYPE	DATE	AMOUNT
40	382957492811	Credit	16APR1991	$100.00
41	382957492811	Debit	.	.

The first line of the DATA step uses the KEEP= data set option. This option works with view descriptors just as it works with other SAS data sets; the KEEP= option specifies that you want only the listed variables included in the new SAS data file WORK.CHKTRANS, although you can use the other variables in the view descriptor within the DATA step. Note that the KEEP= option does not reduce the number of variables mapped by the view descriptor and, therefore, does not reduce the amount of data read by the engine.

When you reference a view descriptor in a SAS procedure or DATA step, it is more efficient to use a SAS WHERE statement than a subsetting IF statement because an IF statement does not reduce the amount of data read. A DATA step or SAS procedure passes the SAS WHERE statement to the interface view engine, which attempts to

create SSAs from the WHERE statement. If the engine can create the SSAs, it processes the SAS WHERE statement and returns to the SAS System only the data that satisfy the WHERE statement. Otherwise, all the data referenced by the view descriptor are returned to the SAS System for processing. Processing IMS-DL/I data using a WHERE statement that the IMS-DL/I engine can turn into SSAs reduces the amount of data read and retrieved by the engine. This improves engine performance significantly. For more information on how IMS-DL/I handles WHERE statements, see "Performance and Efficient View Descriptors" on page 122.

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

Using the SAS System's SQL Procedure

This section provides two examples of using the SAS System's SQL procedure on IMS-DL/I data. The SQL procedure implements the Structured Query Language (SQL) in Version 7 of the SAS System. The SQL procedure is a good way to perform SQL operations with IMS-DL/I, which by itself has no SQL capabilities. The first example illustrates using PROC SQL to combine data from three sources. The second example shows how to use the GROUP BY clause to create new items from data described by a view descriptor.

Combining Data from Various Sources

The SQL procedure provides another way to select and combine data. For example, suppose you have the following:

- a view descriptor, VLIB.CUSTACCT, based on the CUSTOMER and CHCKACCT segments of the IMS-DL/I database ACCTDBD.
- a SAS data file, MYDATA.CHGDATA, which contains checking account numbers and checking fees.
- MYDATA.BANKCHRG, a view descriptor based on data in a DB2 table that contains additional banking fees. (The MYDATA.BANKCHRG view descriptor has been created using the SAS/ACCESS interface to DB2.)

You can use PROC SQL to create a view that joins all these sources of data. When you use the PROC SQL view in your SAS program, the joined data are presented in a single output table. In this example, using the SAS WHERE or subsetting IF statements would not be an appropriate way of presenting data from various sources because you want to compare variables from several sources rather than simply merge or concatenate the data. For more information on the DB2 table used in this example, see Appendix 2.

CAUTION:

Accessing More Than One IMS-DL/I Database When you use PROC SQL to access more than one IMS-DL/I database, the view descriptors for each database must use the same PSB. In addition, a PCB must be included in that PSB for each database you want to access. If you are accessing the same database multiple times, each view descriptor must specify a different PCB using the PCB index field. △

Output 4.5 on page 59, Output 4.6 on page 60, and Output 4.7 on page 60 show the results of the PRINT procedure performed on the VLIB.CUSTACCT view descriptor (based on IMS-DL/I data), the MYDATA.BANKCHRG view descriptor (based on DB2 data), and the MYDATA.CHGDATA data file. The following code generates the output:

```
options nodate linesize=120;
```

```
   proc print data=vlib.custacct;
      title2 'Data Described by VLIB.CUSTACCT';
   run;

   options nodate linesize=80;

   proc print data=mydata.bankchrg;
      title2 'Data Described by MYDATA.BANKCHRG';
   run;


proc print data=mydata.chgdata;
      title2 'SAS Data File MYDATA.CHGDATA';
   run;
```

Output 4.5 Data Described by VLIB.CUSTACCT

The SAS System
Data Described by VLIB.CUSTACCT

OBS	SOC_SEC_ NUMBER	CUSTOMER_NAME	CHECK_ACCOUNT_ NUMBER
1	667-73-8275	WALLS, HOOPER J.	345620145345
2	667-73-8275	WALLS, HOOPER J.	345620154633
3	434-62-1234	SUMMERS, MARY T.	345620104732
4	436-42-6394	BOOKER, APRIL M.	345620135872
5	434-62-1224	SMITH, JAMES MARTIN	345620134564
6	434-62-1224	SMITH, JAMES MARTIN	345620134663
7	178-42-6534	PATTILLO, RODRIGUES	745920057114
8	156-45-5672	O'CONNOR, JOSEPH	345620123456
9	657-34-3245	BARNHARDT, PAMELA S.	345620131455
10	667-82-8275	COHEN, ABRAHAM	382957492811
11	456-45-3462	LITTLE, NANCY M.	345620134522
12	234-74-4612	WIKOWSKI, JONATHAN S.	345620113263

Output 4.6 Data Described by MYDATA.BANKCHRG

The SAS System
Data Described by MYDATA.BANKCHRG

OBS	ssn	accountn	chckchrg	atmfee	loanchrg
1	667-73-8275	345620145345	3.75	5.00	2.00
2	434-62-1234	345620104732	15.00	25.00	552.23
3	436-42-6394	345620135872	1.50	7.50	332.15
4	434-62-1224	345620134564	9.50	0.00	0.00
5	178-42-6534	.	0.50	15.00	223.77
6	156-45-5672	345620123456	0.00	0.00	0.00
7	657-34-3245	345620132455	10.25	10.00	100.00
8	667-82-8275	.	7.50	7.50	175.75
9	456-45-3462	345620134522	23.00	30.00	673.23
10	234-74-4612	345620113262	4.50	7.00	0.00

Output 4.7 Data in the SAS Data File MYDATA.CHGDATA

The SAS System
SAS Data File MYDATA.CHGDATA

OBS	account	charge
1	345620135872	$10
2	345620134522	$7
3	345620123456	$12
4	382957492811	$3
5	345620134663	$8
6	345620131455	$6
7	345620104732	$9

The following SAS statements select and combine data from these three sources to create a PROC SQL view, SQL.CHARGES. The SQL.CHARGES view retrieves checking fee information so that the bank can charge customers for checking services.

```
options nodate linesize=132;
libname sql 'SAS-data-library';

proc sql;
  create view sql.charges as
     select distinct custacct.soc_sec_number,
            custacct.customer_name,
            custacct.check_account_number,
            chgdata.charge,
            bankchrg.chckchrg,
            bankchrg.atmfee,
            bankchrg.loanchrg
     from vlib.custacct,
          mydata.bankchrg,
          mydata.chgdata
     where custacct.soc_sec_number=bankchrg.ssn and
       custacct.check_account_number=chgdata.account;
title2 'Banking Charges for the Month';
```

```
select * from sql.charges;
```

The CREATE statement incorporates a WHERE clause along with the SELECT clause. The last SELECT statement retrieves and displays the PROC SQL view SQL.CHARGES. To select all the items from the view, use an asterisk (*) in place of item names. When an asterisk is used, the order of the items displayed matches the order of the items as specified in the SQL.CHARGES view definition. Notice that PROC SQL prints the output automatically on the display using the IMS-DL/I item names instead of the SAS variable names. It also executes without a RUN statement when the procedure is submitted. Output 4.8 on page 61 shows the data described by the PROC SQL view SQL.CHARGES.

Output 4.8 Data Described by the PROC SQL View SQL.CHARGES

The SAS System
Banking Charges for the Month

SOC_SEC_ NUMBER	CUSTOMER_NAME	CHECK_ACCOUNT_ NUMBER	charge	chckchrg	atmfee	loanchrg
156-45-5672	O'CONNOR, JOSEPH	345620123456	$12	0.00	0.00	0.00
434-62-1224	SMITH, JAMES MARTIN	345620134663	$8	9.50	0.00	0.00
434-62-1234	SUMMERS, MARY T.	345620104732	$9	15.00	25.00	552.23
436-42-6394	BOOKER, APRIL M.	345620135872	$10	1.50	7.50	332.15
456-45-3462	LITTLE, NANCY M.	345620134522	$7	23.00	30.00	673.23
657-34-3245	BARNHARDT, PAMELA S.	345620131455	$6	10.25	10.00	100.00
667-82-8275	COHEN, ABRAHAM	382957492811	$3	7.50	7.50	175.75

Creating New Items with the GROUP BY Clause

It is often useful to create new items with summary or aggregate functions such as the SUM function. 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 PROC SQL to retrieve and manipulate data from the view descriptor VLIB.SAVEBAL, which is based on the CUSTOMER and SAVEACCT segments in the ACCTDBD database. When this query (as a SELECT statement is often called) is submitted, it calculates and displays the average savings account balance for each city.

```
options nodate linesize=80;

proc sql;
   title2 'Average Savings Balance Per City';
   select distinct city,
       avg(savings_balance) label='Average Balance'
          format=dollar12.2
   from vlib.savebal
   where city is not missing
   group by city;
```

Output 4.9 on page 61 shows the query's result.

Output 4.9 Data Retrieved by a PROC SQL Query

```
                The SAS System
        Average Savings Balance Per City

                                        Average
CITY                                    Balance
-----------------------------------------------
CHARLOTTESVILLE                       $1,673.35
GORDONSVILLE                          $4,758.26
ORANGE                                  $615.60
RAPIDAN                                 $672.63
RICHMOND                                $924.62
```

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

Updating a SAS Data File with IMS-DL/I Data

You can update a SAS data file with IMS-DL/I 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.

You can even perform updates when the file to be updated is a Version 6 data file with user-defined, 8-byte SAS variable names and the transaction data are from a Version 7 source. Version 7 uses generated SAS variable names of up to 32 bytes.

You have two choices when you update a Version 6 SAS data file with Version 7 data:

- operate Version 7 in default mode. Your Version 6 program will run, but WHERE processing will not be available.
- set the VALIDVARNAME SAS System option to V6 to operate in Version 6 mode. The V6 option offers functionality comparable to Version 6 of the interface view engine, including WHERE processing.

The VALIDVARNAME SAS System option lets you control what type of variable names will be used in a SAS session. It enforces the naming conventions by converting any nonconforming names to the necessary format. The default format is V7. When a Version 6 program is run under V7, the software replaces the 8-byte Version 6 variable names created with the SASNAME= (sn=) parameter with longer SAS variable names generated from the ITEM name. When a Version 7 program is run and VALIDVARNAME=V6, the longer variable names are truncated to 8 bytes. The conversion is permanent: when a conversion is made, the original names are not stored.

The following examples illustrate the situations in which each of the options is appropriate.

Example of VALIDVARNAME=V6

Suppose you have a Version 6 SAS data set, VER6.SSNUMS, which contains some customer names and Social Security numbers. You want to update this data set with data described by VLIB.SSNAME, a view descriptor based on the CUSTOMER segment of the IMS-DL/I database ACCTDBD. Since this will require you to first sort the data then create an output data set with the sorted data, this is a good situation for using VALIDVARNAME=V6.

To perform the update, you would enter the following SAS statements:

```
options validvarname=V6;
options nodate linesize=80;
libname ver6 'SAS-data-library';

proc sort data=ver6.ssnums;
   by ssnumb;
run;

proc print data=ver6.ssnums;
   title2 'VER6.SSNUMS Data File';
run;

proc sort data=vlib.ssname out=mydata.newnums;
   by ssnumb;
run;

proc print data=mydata.newnums;
   title2 'Data Described by MYDATA.NEWNUMS';
run;

data mydata.newnames;
   update ver6.ssnums mydata.newnums;
   by ssnumb;
run;

proc print data=mydata.newnames;
   title2 'MYDATA.NEWNAMES Data File';
run;
```

The new SAS data file MYDATA.NEWNAMES is a Version 6 data file stored in a Version 6 data library associated with the libref MYDATA.

Output 4.10 on page 63, Output 4.11 on page 64, and Output 4.12 on page 64 show the results of PRINT procedures for the original data file, the transaction data, and the updated data file.

Output 4.10 Data in the Data File to Be Updated, VER6.SSNUMS

The SAS System
VER6.SSNUMS Data File

OBS	SSNUMB	NAME
1	267-83-2241	GORDIEVSKY, OLEG
2	276-44-6885	MIFUNE, YUKIO
3	352-44-2151	SHIEKELESLAM, SHALA
4	436-46-1931	NISHIMATSU-LYNCH, CAROL

Output 4.11 Data Described by Updated Data File MYDATA.NEWNUMS

The SAS System
Data Described by MYDATA.NEWNUMS

OBS	SSNUMB	NAME
1	156-45-5672	O'CONNOR, JOSEPH
2	178-42-6534	PATTILLO, RODRIGUES
3	234-74-4612	WIKOWSKI, JONATHAN S.
4	434-62-1224	SMITH, JAMES MARTIN
5	434-62-1234	SUMMERS, MARY T.
6	436-42-6394	BOOKER, APRIL M.
7	456-45-3462	LITTLE, NANCY M.
8	657-34-3245	BARNHARDT, PAMELA S.
9	667-73-8275	WALLS, HOOPER J.
10	667-82-8275	COHEN, ABRAHAM

Output 4.12 Data in the Updated Data File MYDATA.NEWNAMES

The SAS System
MYDATA.NEWNAMES Data File

OBS	SSNUMB	NAME
1	156-45-5672	O'CONNOR, JOSEPH
2	178-42-6534	PATTILLO, RODRIGUES
3	234-74-4612	WIKOWSKI, JONATHAN S.
4	*267-83-2241*	*GORDIEVSKY, OLEG*
5	*276-44-6885*	*MIFUNE, YUKIO*
6	*352-44-2151*	*SHIEKELESLAM, SHALA*
7	434-62-1224	SMITH, JAMES MARTIN
8	434-62-1234	SUMMERS, MARY T.
9	436-42-6394	BOOKER, APRIL M.
10	*436-46-1931*	*NISHIMATSU-LYNCH, CAROL*
11	456-45-3462	LITTLE, NANCY M.
12	657-34-3245	BARNHARDT, PAMELA S.
13	667-73-8275	WALLS, HOOPER J.
14	667-82-8275	COHEN, ABRAHAM

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

Example of VALIDVARNAME=V7

The following is an example of a Version 7 update of data. The Version 7 data set, MYDATA.SSNUMS, is updated with data described by the view descriptor VLIB.SSNAME. Both the data in the data set and in the view descriptor are sorted by social security number before the output data set is used to update the existing data set.

To perform the update, you would enter the following statements:

```
proc sort data=mydata.ssnums;
   by soc_sec_number;
run;
```

```
proc print data=mydata.ssnums;
   title2 'MYDATA.SSNUMS Data Set';
run;

proc sort data=vlib.ssname out=mydata.newnums;
   by soc_sec_number;
run;

proc print data=mydata.newnums;
   title2 'Data Described by MYDATA.NEWNUMS';
run;

data mydata.newnames;
   update mydata.ssnums mydata.newnums;
   by soc_sec_number;
   run;

proc print data=mydata.newnames;
   title2 'MYDATA.NEWNAMES Data Set';
   run;
```

The new SAS data file MYDATA.NEWNAMES is a Version 7 data file that is stored in a Version 7 data library associated with the libref MYDATA. Output 4.13 on page 66, Output 4.14 on page 66, and Output 4.15 on page 67 show the results of the PRINT procedures for the original data file, the transaction data, and the updated data file.

Output 4.13 Data in the Data File to be Updated, MYDATA.SSNUMS

```
                The SAS System
            MYDATA.SSNUMS Data Set

            soc_sec_
OBS         number        customer_name

1      267-83-2241    GORDIEVSKY, OLEG
2      276-44-6885    MIFUNE, YUKIO
3      352-44-2151    SHIEKELESLAM, SHALA
4      436-46-1931    NISHIMATSU-LYNCH, CAROL
```

Output 4.14 Data Described by Updated Data File MYDATA.NEWNUMS

```
                The SAS System
           Data Described by MYDATA.NEWNUMS

           SOC_SEC_
  OBS       NUMBER          CUSTOMER_NAME

   1     156-45-5672    O'CONNOR, JOSEPH
   2     178-42-6534    PATTILLO, RODRIGUES
   3     234-74-4612    WIKOWSKI, JONATHAN S.
   4     434-62-1224    SMITH, JAMES MARTIN
   5     434-62-1234    SUMMERS, MARY T.
   6     436-42-6394    BOOKER, APRIL M.
   7     456-45-3462    LITTLE, NANCY M.
   8     657-34-3245    BARNHARDT, PAMELA S.
   9     667-73-8275    WALLS, HOOPER J.
  10     667-82-8275    COHEN, ABRAHAM
```

Output 4.15 Data in the Updated Data File MYDATA.NEWNAMES

The SAS System
MYDATA.NEWNAMES Data Set

OBS	soc_sec_ number	customer_name
1	156-45-5672	O'CONNOR, JOSEPH
2	178-42-6534	PATTILLO, RODRIGUES
3	234-74-4612	WIKOWSKI, JONATHAN S.
4	267-83-2241	GORDIEVSKY, OLEG
5	276-44-6885	MIFUNE, YUKIO
6	352-44-2151	SHIEKELESLAM, SHALA
7	434-62-1224	SMITH, JAMES MARTIN
8	434-62-1234	SUMMERS, MARY T.
9	436-42-6394	BOOKER, APRIL M.
10	436-46-1931	NISHIMATSU-LYNCH, CAROL
11	456-45-3462	LITTLE, NANCY M.
12	657-34-3245	BARNHARDT, PAMELA S.
13	667-73-8275	WALLS, HOOPER J.
14	667-82-8275	COHEN, ABRAHAM

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