Integrating New Applications Tools in Version 6
Jill Sarisky, SAS Institute Inc., Cary, NC
John McIntyre, SAS Institute Inc., Cary, NC

ABSTRACT
This paper describes an application that integrates new features of Release 6.06 SAS software. Access to database tables is discussed, focusing on the ACCESS procedure. Combining data from database tables with SAS data sets by using the REPORT procedure is then highlighted. The interactive report writer, the REPORT procedure, is discussed, showing how data can be grouped and analyzed in various ways. A user application is then presented using screen control language (SCL), an enhancement to SAS/AF and SAS/FSP software.

INTRODUCTION
Release 6.06 of SAS software offers increased flexibility in storing and accessing data, new reporting abilities, and greater power in building user applications. Multi-Engine Architecture, new for Release 6.06, provides the mechanism for accessing a variety of different types of files as if the files were SAS data sets. Data stored in various databases can be accessed directly by the SAS System once the appropriate engine is used. A new reporting procedure, the REPORT procedure, provides a full-screen interactive report writer. A new programming language, screen control language, provides statements and functions to control program flow and to interact with elements of screens.

ACCESSING DIFFERENT DATA FILE TYPES
With Multi-Engine Architecture, data files in various databases (such as DB2, ORACLE, and SQL/DS) can be accessed by the SAS System as if they were SAS data sets. To do this, the appropriate database engine must be selected. The ACCESS procedure, new for Release 6.06, provides the interface for this.

Suppose a grocery store headquarters has food items in a DB2 table. Here is a portion of the table's contents:

<table>
<thead>
<tr>
<th>DEPT</th>
<th>PROD CODE</th>
<th>UNIT PRICE</th>
<th>ITEM NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAKERY</td>
<td>0123267856</td>
<td>$1.39</td>
<td>LIGHT BREAD</td>
</tr>
<tr>
<td>BAKERY</td>
<td>065423456</td>
<td>$1.60</td>
<td>WHOLE WHEAT BREAD</td>
</tr>
<tr>
<td>BAKERY</td>
<td>067763211</td>
<td>$1.33</td>
<td>SANDWICH ROLLS</td>
</tr>
<tr>
<td>BAKERY</td>
<td>045324986</td>
<td>$1.20</td>
<td>BAGELS</td>
</tr>
<tr>
<td>MEAT</td>
<td>4324086535</td>
<td>$0.69</td>
<td>CHICKEN WINGS</td>
</tr>
<tr>
<td>MEAT</td>
<td>4865409765</td>
<td>$1.55</td>
<td>CHICKEN PARTS</td>
</tr>
<tr>
<td>MEAT</td>
<td>462678089</td>
<td>$2.59</td>
<td>CUBE STEAK</td>
</tr>
<tr>
<td>MEAT</td>
<td>4523769053</td>
<td>$5.99</td>
<td>RIB EYE STEAK</td>
</tr>
<tr>
<td>PRODUCE</td>
<td>2087654645</td>
<td>$0.53</td>
<td>CARROTS</td>
</tr>
<tr>
<td>PRODUCE</td>
<td>2145086542</td>
<td>$0.78</td>
<td>CELERY</td>
</tr>
<tr>
<td>PRODUCE</td>
<td>2965423456</td>
<td>$0.39</td>
<td>POTATOES</td>
</tr>
<tr>
<td>PRODUCE</td>
<td>2756231446</td>
<td>$0.82</td>
<td>LEAF LETTUCE</td>
</tr>
</tbody>
</table>

You describe this table to the SAS System by defining the following special files called SAS/ACCESS descriptor files:

- master descriptor file contains information about the DB2 table you want to access. It contains the table name, the column names, and the data formats.
- subdescriptor file (also called View) defines a subset of data from the table described by the master descriptor file. You choose this subset by selecting particular columns and specifying the subsetting criteria that the data must meet.

Subdescriptor files provide a security feature by allowing some columns to be "invisible." If certain columns are not defined in the subdescriptor, the user has no way of knowing they exist. Several subdescriptors can be defined for each master descriptor file, providing a way to give different users different columns to access.

Subdescriptors also can save time for users who are interested in only a subset of a large file. Accessing the view prevents users from continually coding statements to omit unwanted columns. Also, the view can contain a DB2 WHERE clause, which omits unwanted rows.

The first step in establishing the SAS/ACCESS interface to DB2 is to associate a libname with the library in which you want to store master and subdescriptor files. After you have allocated the data set, submit this LIBNAME statement from the PROGRAM EDITOR window of the SAS Display Manager System:

```
libname dba 'access':v6detll.os.sas.access';
```

You now invoke PROC ACCESS to create a master descriptor of your table by submitting

```
proc access;
run;
```

PROC ACCESS runs in full-screen mode. You see the Data Services window, shown in Screen 1, listing the files associated with your libnames. Since this is your first time using PROC ACCESS, the only file listed is your profile catalog. After you create master and subdescriptor files, they are listed on this window.
on the command line.

The DB2 Master Descriptor Identification Panel then appears, as shown in Screen 2. You fill in the AUTHORIZATION ID (the user id who created the table), the TABLE NAME, and the SSID fields. The ASSIGN NAMES field is NO, meaning that you enable users to create views with their own names and formats instead of letting PROC ACCESS construct suitable SAS names for any DB2 names that are too long to be SAS names. LIBRARY and MEMBER are filled in from the PROC ACCESS statement. TYPE is ACCESS, indicating a master descriptor.

Screen 2 DB2 Master Descriptor Identification Panel

After you press ENTER, the DB2 Master Descriptor Display Panel appears (see Screen 3). Use this panel to define columns in the DB2 table. All the columns are listed.

Screen 3 DB2 Master Descriptor Display Panel

The field FUNC is available to designate columns as nondisplay. Any column marked with an N (nondisplay) is not shown on the Sub Descriptor Display Panel.

Type END on the command line to save this master descriptor and return to the DB2 Master Descriptor Identification Panel, and then type END again to return to the Data Services window, Screen 4, which now lists the Master Descriptor just created.

Screen 4 Data Services Window Listing Master Descriptor

Now that you have created the master descriptor file, continue by creating the subdescriptor file, which is used to access the data. To do this, place your cursor on the blank line in front of the Master Descriptor HEADQ in library DBACCESS, type a C (for Create), and press ENTER.

The DB2 Sub Descriptor Display Panel appears (see Screen 5). You enter the library and member names. A subdescriptor is member type VIEW. (You have an option to extract the DB2 data into a SAS data set by filling in the output SAS data set library and member.) Select the columns for your view by entering S in each FUNC field. Note that only three columns have been selected. Since you elected to assign your own SAS names, type in the three names.

Screen 5 DB2 Sub Descriptor Display Panel

Type END on the command line to save this subdescriptor and then return to the Data Services window. Now you can use this subdescriptor as input to many SAS procedures.

After you have created a subdescriptor file, you can browse it, as you would browse a SAS data set. Screen 6 shows the Data Services window. Type B (for Browse) in front of the subdescriptor PRICES and then press ENTER.
Use the DATA step to build a SALES data set containing the daily sales for one grocery store. Here is a listing of SALES, which you save in the SASLIB library:

<table>
<thead>
<tr>
<th>PROCODE</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0123267856</td>
<td>14</td>
</tr>
<tr>
<td>4865408765</td>
<td>8</td>
</tr>
<tr>
<td>2143086542</td>
<td>14</td>
</tr>
<tr>
<td>4324086535</td>
<td>10</td>
</tr>
<tr>
<td>0451234986</td>
<td>8</td>
</tr>
<tr>
<td>2756023146</td>
<td>7</td>
</tr>
<tr>
<td>0656423456</td>
<td>21</td>
</tr>
<tr>
<td>4642676009</td>
<td>12</td>
</tr>
<tr>
<td>086743211</td>
<td>17</td>
</tr>
<tr>
<td>2087664964</td>
<td>22</td>
</tr>
<tr>
<td>2965423456</td>
<td>11</td>
</tr>
<tr>
<td>4523789653</td>
<td>8</td>
</tr>
</tbody>
</table>

You can list your data set with a new SAS procedure, the SQL procedure (Structured Query Language), which allows you to do the following:

- Retrieve data from SQL database tables. Use the SELECT statement for this.
- Create database tables and modify existing tables.
- Create views (SELECT statements that are stored as text and executed at run time).

SQL tables are very similar to SAS data sets. SAS data sets have an inherent ordering and concept of an nth row; for example, you can specify a FIRST and LAST by-variable in a data set. A SAS data set and an SQL table are both relational files arranged in two-dimensional tables consisting of rows and columns. The SQL term row corresponds to an observation in a SAS data set; the SQL term column is much like a variable in a SAS data set.

To display the SALES data set with PROC SQL, use the SELECT statement, which retrieves data for the query, formats the selected rows, and prints to SAS Output. Here is the code you submit to ask for columns PROD CODE and QUANTITY. Note that in the FROM clause the table name is preceded by a libref (SAS data library reference), referring to the table's SAS data library. LINKING a table name with a library is called qualifying the table (name).

```
proc sql;
  select prodc ode, quantity from salis. sale s;
```

Note that a RUN statement is not required for PROC SQL. SQL statements are executed as soon as a semi-colon is encountered. Also, as long as you continue with SQL procedure statements, you do not have to repeat the PROC SQL statement. Screen 8 shows a display of the OUTPUT window.
Use the SELECT statement to specify columns in a table. To specify selected rows, use the WHERE clause, which instructs SQL to display only result rows that satisfy a condition. Here is your code to display only quantities greater than or equal to 14:

```
select product, quantity
from saaslib.sales
where quantity > 14
order by quantity;
```

Screen 9 shows your output.

Now you can proceed to combine this SAS data set with your database subdescriptor by matching values in columns that relate the two tables. This is called joining tables. You join the subdescriptor of the DB2 data set DBACCESS.PRICES (columns DEPARTMENT, PROD CODE, and UNITPRIC) with the SAS data set SASLIB.SALES (columns PROD CODE and QUANTITY). The column that is common to both tables is PROD CODE. Since both tables have a column for PROD CODE, you must qualify the column (precede the column name with the table name). To get all the information needed, the tables must be joined together by matching rows that have common product codes. In the WHERE expression below, PRICES.PROD CODE = SALES.PROD CODE performs the matching of these two columns:

```
select department, prices.prod code, unitpric, quantity
from dbaccess.prices, saaslib.sales
where prices.prod code = sales.prod code
order by department;
```

Screen 10 shows the output.

Calculations can be performed on numeric columns with arithmetic expressions. The available calculations are addition (+), subtraction (-), multiplication (*), and division (/). The query below displays the total sales for each store item, a product of unit price and quantity:

```
select department, prices.prod code, unitpric, quantity,
       unitpric * quantity
from dbaccess.prices, saaslib.sales
where prices.prod code = sales.prod code
order by department;
```

Screen 11 displays the output.

Note that the total sales column has no heading and that the decimal points are not aligned. This is the default display format for expressions in the SELECT clause. To enhance the appearance, add a column alias (a temporary name for a column) and format the output, allowing six numbers including the dollar sign before the decimal point and two numbers after.

```
select department, prices.prod code, unitpric, quantity,
       unitpric * quantity
from dbaccess.prices, saaslib.sales
where prices.prod code = sales.prod code
order by department;
```

Screen 11 Adding a Column
Screen 12 shows the output.

SCREEN 12 TOTSALES as Column Alias

The SELECT statements used so far create a display of your data in the OUTPUT window, but the format of the display is not saved. To store the display format, you can create an SQL view, a SELECT statement that is stored as text and executed at run time. Any SAS procedure can read an SQL view as if the view were a SAS data set. A view displays data derived from existing tables or other views. Views do not actually contain data, as tables do. This is advantageous because the composite tables can be changed without affecting the view. At run time, the view contains the current data in the tables.

Note that a PROC ACCESS view is different from a PROC SQL view. A PROC ACCESS view is a subset of data in a database table (DBZ, ORACLE, SQL/DS) whereas a PROC SQL view is derived from one or more tables (from a SAS or other data set). SQL views have a variety of uses. One use is to provide users with an alternate "view" of stored data. For instance, a group of users may only be concerned with data that pertains to their department. A view can be created that selects only the rows and columns of interest. The user retrieves data using the view name instead of the table name where the actual data are stored. This reduces the length of their SQL queries and shields them from unwanted information. A second use of views is to hide the details of large and complex joins. A third use is to code an often-used join in a view once and then refer to it many times in succeeding queries to get the most current data available. Finally, an Information Center can provide powerful views for the SAS user community, thereby making the need for all users to learn SQL details.

You define an SQL view with a CREATE statement. A permanently stored view is preceded by a libref referring to a SAS data library. The statements below create a permanent view named SASEDATA:

```sql
CREATE view saslib.salesdata as
  select departmt, prices.prodcode, unitprice, quantity,
          unitprice*quantity as totsales for department 1
from dbaccess.prices, saslib.sales
where prices.prodcode = sales.prodcode
order by departmt;
```

After you submit these statements, a message appears in the LOG window:

NOTE: SQL view saslib.salesdata has been defined.

You can display the text of a query stored in a view definition by using the DESCRIBE statement. Here you ask for the SASEDATA view definition:

```sql
describe view saslib.salesdata;
```

In the log window, this message is displayed:

NOTE: SQL VIEW saslib.salesdata is defined as:
  select departmt, prices.prodcode, unitprice, quantity,
          unitprice*quantity as totsales for department 1
from dbaccess.prices, saslib.sales
where prices.prodcode = sales.prodcode
order by departmt;

This view can be used as input to many SAS procedures. (Note that the ORDER clause describes DEPARTMT as asc, meaning ascending order, which is the default. The ability to store the ORDER information is a feature of PROC SQL that is not available with many other SQL-based database management systems.

REPORTING FACILITY

Another new procedure, the REPORT procedure, allows creation of full-screen reports in an interactive environment. Suppose you have created a view of sales containing a week's sales for regional grocery stores. You can generate reports based on this view.

Submitting the following code produces the base report shown in Screen 13:

```sql
proc report is saslib.salesdata;
run;
```

SCREEN 13 PROC REPORT Base Report

The second line on the screen is a display of some frequently used PROC REPORT commands. Much of PROC REPORT is cursor driven. To activate a command, position the cursor on the command and then press ENTER (or, if using a mouse, click the mouse). The last three screen lines display eleven function keys set with PROC REPORT commands. If you are using a mouse, select a column by positioning the cursor over it and clicking. If you are not using a mouse, press the function key.

The middle lines list the base report, showing department, product code, total sales for that product, store location, and sale
date. You can scroll down to view sales for three stores for the week.

To customize your report, group sales first by department. Do this by setting the attribute of department to GROUP. Place your cursor on the heading DEPT and then press the function key for ATTR (Attribute). Screen 14 is a display of the Attribute window, partially overlaying the report.

Screen 14 Attribute Window for DEPT

Place your cursor on GROUP (under Usage), press ENTER, then move the cursor to End and press ENTER again. The Attribute window disappears, and you see Screen 15 - data grouped by department.

Screen 15 Report Grouped by DEPT

If you scroll down, you see each department's sales by each product code and store location, as well as the sale date.

Next, group sales by both department and product code. Place your cursor on PRODCODE, and press the function key for Attr. Screen 16 shows the Attribute window, this time for PRODCODE. Again choose usage of GROUP.

Screen 16 Attribute Window for PRODCODE

Screen 17 shows the report.

Screen 17 Data Grouped by DEPT and PRODCODE

If you scroll down, you see sales grouped first by department and then by product code for each store location and each sale date.

Another feature of the reporting facility is adding and deleting columns. To simplify your report, you can delete the column for sale date. Place your cursor on SALEDATE, press the function key for Delete, and the column is removed.

Continue by defining store location as a grouping variable.

Since the data are already grouped by store location, the report output does not change. You now would like to produce statistics for sales. This requires you to enter the desired statistic in the Attribute window for the variable TOTSALES. Place your cursor on TOTSALES and then press the function key for Attr. Screen 18 is your display.
Choose the usage of VALUE and fill in statistic with SUM to indicate that TOTSALES is to be summed by grouping variables. Update format from DOLLAR9.2 to DOLLAR10.2 to allow enough room for the totals. Your resulting report sums sales for each department, product code, and store location for all dates. $525.66 is the total weekly sales for the Cary store for product code 0056423456.

Finally, you would like to see grand totals for sales. Move the cursor to Summarize (on the command display line) and press ENTER. Screen 20 shows a Summary window partially overlaying your report.

Report structures can be stored and then reused with the same data sets and views or with others. PROC REPORT can be run in batch as well as interactively.

APPLICATION DEVELOPMENT FACILITY

Customized applications can be built that allow interaction with the SAS System. A powerful programming language, screen control language (SCL), provides routines and functions to manage screens that make up an application. SCL can be used in SAS/AF software PROGRAM screens and in SAS/FSP software FSEDIT and FSBROWSE procedures.

You can build an application that gives a report and a chart of the weekly sales view or both. Begin by invoking the BUILD procedure, which creates MENU, PROGRAM, and HELP screens that make up the application. These screens are placed in a catalog that you name when you invoke the BUILD procedure. Begin by submitting:

```sas
proc build catalog = saslib.appdev;
run;
```
You see the SAS/AF window that lists the catalog directory. The first screen to create is the main menu, shown in Screen 22. The (E) on the top line indicates that you are in EDIT mode.

**Screen 22 Main Menu Display**

You need to associate a screen to be called when each option is entered. This is done by entering the screen name on the attribute panel of the menu. Enter `attr` on the command line of the MENU entry, and fill in the screens to be called, as in Screen 23.

**Screen 23 Attribute Panel of Main Menu**

Press END until each panel is removed from the screen and the catalog directory is displayed.

Proceed by creating the PROGRAM screen to be called when the user selects the option to view a report of weekly sales. Enter `edit report.program` on the command line of the catalog directory screen to see Screen 24, the display panel of the PROGRAM screen.

**Screen 24 Blank Display Panel for PROGRAM Screen**

The next panel is what the user sees when a report of the sales data is to be generated. Enter instructions for the user on this panel, as shown in Screen 25.

**Screen 25 Display Panel for Report Program**

Enter the programming statements needed on the source panel of the PROGRAM screen. After entering the `SOURCE` command on the command line of the display panel, you have a display of the source panel, shown in Screen 26.
Use screen control language to write the source portion of your program. Programs have three parts, with these reserved labels:

**INIT** (initialization) prepares the environment (display panel) for the user to see. The tasks in this phase are completed before the display panel appears, so any initial values or messages set here are displayed when the application begins. Typical tasks in this part are initializing variables, importing values through macro variables, displaying initial messages, and opening data sets.

**MAIN** (execution) user interacts directly with the application to accomplish specific tasks. In this section, the user may be prompted for information, values may be verified, or data sets may be checked.

**TERM** (termination) application ends and housekeeping tasks such as closing files, resetting default values, and submitting final SAS statements are performed. The tasks in this phase are performed after the user issues an END or CANCEL command, so much of this action is transparent to the user.

SCL-reserved labels control when statements are executed by letting the applications developer selectively process user input. SCL statements actually provide the ability to have two-way communication during an application.

Screen 27 shows the source panel for your program.
Press END three times until you again see the catalog directory. 

To complete the application, create a program to display a chart of any week's sales data. Enter edit chart program on the command line of the catalog directory, and create your display panel, shown in Screen 29.

This display panel, when shown to the user, will look like the screen above, except that dashes will replace the field preceded by an ampersand (user field). You require the user to enter a week by entering

on the command line, which displays the attribute panel for this screen. Enter REQUIRED as an option for the field WEEK. You can access this user field in your SCL source panel. Screen 30 shows the source panel filled in. Note that user fields in SCL are not preceded by an ampersand, but, within a SUBMIT block, the ampersand is required.

The INIT section has no set-up for display, only a RETURN statement. The MAIN section creates a message for the user by setting the SCL system variable(MSG) to 'Press END to see chart for week of', followed by the date entered. This message is displayed on the message line (below the command line) on the PROGRAM window when it is redisplayed.

The TERM section submits the SAS statements to display a chart for the week requested.

Compile this program, set the parent to MAIN.MENU, and return to the catalog directory screen. 

To invoke our application, enter the AF command, which puts you into SAS/AF execution mode. The AF command replaces the DISPLAY procedure. This command can be entered on the command line of any display manager window or from the catalog directory. Enter

on the command line to display your main menu, shown in Screen 31.
When you press END, you see the base report for the week of January 8, 1989. You can now scroll through the base report or use any of the customizing features of PROC REPORT. After you have completed viewing the report, press END to return to the Main Menu. If you select option 2 to see a chart, you see Screen 34.

SCL has many more options, including:

- functions for data sets, files, variables, observations, user fields, commands, formatting, character strings, macros, selection lists, window control, PREVIEW window, legend display, and error diagnosis.

- features for DO GROUP control and extended tables

- ability to use some SAS statements and functions.

SCL can also be used with SAS/FSP software. You can add the power and flexibility of SAS DATA step programming to your FSEDIT application. You can perform tasks such as checking data, calculating values for fields based on user input, cross-validating values entered, and providing messages and help to the users.

CONCLUSION

Release 6.06 of SAS software offers powerful features for:

- accessing data in various databases by using the ACCESS procedure
- combining data from different sources using the SQL procedure
- generating reports using the REPORT procedure
- developing applications using the screen control language feature of SAS/AF and SAS/FSP software.

SAS, SAS/AF, SAS/FSP and SAS/ACCESS are registered trademarks of SAS Institute Inc., Cary, NC, USA.

ORACLE is a registered trademark of Oracle Corporation.

DB2 and SQL/DS are trademarks of IBM Corporation.

Screen 32 Display of Report Program

When you press END, you see the base report for the week of January 8, 1989. You can now scroll through the base report or use any of the customizing features of PROC REPORT. After you have completed viewing the report, press END to return to the Main Menu. If you select option 2 to see a chart, you see Screen 33.

Screen 33 Display of Chart Program

If you select the week of January 8, 1989, you see Screen 34.