Overview

Using SAS/AF and SCL, this application leads the user through the creation of SQL queries. Push buttons are used to select the desired library. Selection lists facilitate the choice of one or more data sets and their associated variables. Aliases are assigned by default, but the user is given the option of changing them. The WHERE clause can be generated entirely via selection lists and push buttons. The same is true for HAVING, GROUP BY and ORDER BY clauses.

Once the user has created the SQL query, s/he can view it in the PREVIEW window, where it can be edited. Upon leaving the PREVIEW window, the query is submitted via push button. The query is automatically saved to the user's PDS. The user always has the option (via a push button) to pull in the last query saved and edit or submit it.

By default, the query generates a temporary VIEW. After the query has run, the user is asked if s/he wants to save the results, either as a permanent VIEW or as a data set. Permanent views are accessed via a selection list, any time the user wants to apply one.

The user does not need to know SQL. If they do, so much the better; if they do not, the system teaches it to them, while they develop real queries.

Environment

As part of the interactive SAS* environment for Intel Products Group Finance, users have access to a number of utilities. One of these is the SQL query generator. This facility can be used as a stand-alone report and data generator or as a means of extracting data from our large, complex Sales and Marketing or Manufacturing databases (indexed SAS data sets) to use with other end user utilities or with SAS/ASSIST*.

The other user utilities also have SAS/AF shells, using SCL. These include front ends for PROC PRINT, PROC TABULATE, PROC REPORT, PROC MEANS, data uploads and downloads, messaging and so on. The emphasis is on creating flexible tools for end users, to enable them to accomplish tasks that would otherwise require programmer involvement.

The Query Facility

When the user selects the SQL Query facility from a selection list of utilities, s/he is taken to the entry program of the application, at which point s/he is asked which library should serve as the storage facility for permanent views and data sets created. The library to use for input is also requested. Almost all responses can be given via push buttons. A PMENU allows access to the other utilities, as well as all of the canned applications the user is set up to run.

If the user enters a ? in the data set name(s) field, a data set selection window (using the specified input library) is opened, using DIRLIST. Once the library and data set(s) have been selected, choosing “SQL Queries” from the PMENU opens a selection list of available programs to run. This paper will concentrate on the first option (generating a SQL query with assistance). However, the other options should be noted:

Generate SQL Query (No Help)

Upon making this selection, the user is taken to a SAS/AF panel which has been set up to work as a code editor. Once s/he is satisfied, the code is written to the PREVIEW window, from which it can be submitted.

Edit Existing SQL Queries

Editing stored views involves allowing the user to change descriptions or to delete views, via an FSEDIT of the data set containing view names and descriptions. If the name is deleted from this data set, the view itself is deleted.
"Describe" Existing SQL Queries

To see the logic contained in a stored view, the user selects this option. This clears the SAS Log, sends the view logic to the Log, then writes the Log to a file and calls our output management utility to display it and print it.

Report From Existing SQL Query

Generating a report from a stored view involves opening a selection list window (using the data set described above), allocating libraries necessary to the view and asking the user if titles, totals, etc. are desired. SQL is used, for example, to open a selection list of numeric variables in the view, for report subtotals/totals.

When the user selects the option to get assistance in the creation of a query, the following SAS/AF panel is displayed. The push buttons shown result in one or a series of SAS/AF programs being called. Selecting the option to edit the last SQL query stored copies code from a PDS member (where it was automatically stored, previously) into the PREVIEW window, where the user can edit it.

Template For Creation of SQL Query

Run Cancel Help (Local) Help (SQL)

In this program, you will be constructing an SQL statement to produce a report. In some fields, a '?' will get help or open a selection list.

[Press ENTER here to edit last SQL statement stored ]
[Press ENTER here to select variables from data files]
[Press ENTER here to create columns, e.g. MATCOST-MKTCOST]
[Press ENTER here to create WHERE statement]

Group by (use with discretion, as grouping is expensive -- use alias):

[Press ENTER here to provide subsetting logic for GROUPS]

Order by (use alias):

Title:

Create SQL Query (Computed Columns)

Run Cancel Help (Local) Help (SQL)

In this program, you will be constructing an SQL statement to produce a report. If you are uncertain about any of the entries requested, place a '?' in the field, to get help or open a selection list window.

Create computed columns and name them, creating statements such as:

SUM(MATCOST) AS TOTMAT (to summarize a group)
MATCOST+MKTCOST/2 AS MATTOMKT
MATCOST-MKTCOST AS DELTA1

Selecting Variables For A Query

When the user selected one or more data sets, a PROC CONTENTS was run to generate a data set containing variable names and labels. This is displayed in a selection list, to allow the user to specify the columns for the query. The names of the data sets from which vars are actually selected are picked off and added to the FROM statement. When the desired vars are selected (up to 15), a screen displays the list of vars in DNAME.VARNAME format, with another column showing the default query column name (the var name, without the data set name prefix). The user can change any alias, then select RUN from the PMENU to return to the screen labeled "Template For Creation Of SQL Query" (see first screen, below).

If the user elects to create new columns, the screen at the bottom of the page is displayed. In building new column logic, the user can enter a '?' at any time, to open a selection list of variable names. For example, the user could type:

FILE.VAR1 / FILE.VAR2 - ?

and use the selection list to fill in the next variable name.
Creating The WHERE Clause

The creation of the WHERE clause can often be the most difficult for a user to handle, on a free form basis. The screen below is designed to allow the user to "fill in the blanks"; with help, including push buttons and a selection list of vars, to generate the structure and the content of the SQL WHERE clause.

By using parentheses and AND and OR push buttons, the user can create complex logic combinations with little effort. Selection lists exist for the variables in the data sets from which the query is generated and for operands used in the WHERE clause. Thus, the user gets as much or as little help as s/he requires.

The user can enter a ? anywhere on either of the two variable entries or on the operand entry to produce a selection list. The operand selection list uses a HELP entry as a selection list, as described in the SAS/AF manuals. The screen below shows a fully constructed WHERE clause (note the use of push buttons for logical expressions). Only the portion filled in by the user and push buttons which have been activated are shown.

Other Clauses

The GROUP BY and ORDER BY options on the template screen also allow the use of selection lists (displaying the aliases). The HAVING clause push button leads to a screen similar to that of the WHERE clause builder, to develop the subsetting logic for groupings.

Viewing The Query

Once the query is "built," the user presses the Run option on the PMENU to open a window with two push button options (aside from the ever present PMENU, which offers help and an exit). The two options are:

View/ Edit the SQL statement
Submit the SQL statement now

If the user selects the first option, s/he is taken to the PREVIEW window, where the query code is currently stored. The code was written to the PREVIEW window, incrementally, through the use of submit blocks (with no options on the submit statement). A commented paragraph precedes the query code, telling the user how to enhance the query by editing it. The PMENU also provides the user with the capability to write the code out to a file.

---

Create SQL Query (WHERE)
Run Cancel Help (Local) Help (SQL)

In this program, you will be constructing an SQL statement to produce a report. If you are uncertain about any of the entries requested, place a ? in the field, to get help or open a selection list window.

Construct the SQL where clause, below, creating statements such as:

\[
\begin{align*}
\text{MATCOST} &> \text{AVGPRICE AND (MKTCOST} > 0 \text{ OR Q1PJCOST} > 0) \\
\text{ITEMNBR LIKE '459123%'}
\end{align*}
\]

Where:

\[
\begin{align*}
\text{[ ]} & \text{ [AND] [OR] [ ]} \\
\text{[ ]} & \text{ [AND] [OR] [ ]} \\
\text{[ ]} & \text{ [AND] [OR] [ ]} \\
\text{[ ]} & \text{ [AND] [OR] [ ]} \\
\text{[ ]} & \text{ [AND] [OR] [ ]} \\
\text{[ ]} & \text{ [AND] [OR] [ ]}
\end{align*}
\]

---

Create SQL Query (WHERE)
Run Cancel Help (Local) Help (SQL)

In this program, you will be constructing an SQL statement to produce a report. If you are uncertain about any of the entries requested, place a ? in the field, to get help or open a selection list window.

Construct the SQL where clause, below, creating statements such as:

\[
\begin{align*}
\text{FILE1.CHARVAR LIKE 'ABCDE%'}
\end{align*}
\]

Where:

\[
\begin{align*}
\text{FILE2.NUMVAR > 500} & \text{ [AND] [OR]} \\
\text{FILE1.NUMVAR2 < 0} & \text{ [ ]}
\end{align*}
\]

---
Submitting The Query

After the user is through viewing and editing the query code, s/he returns to the SAS/AF screen from which s/he entered the PREVIEW window and selects the option to submit the query. In our environment, the results of the query are written to the user’s output file (using PROC PRINTTO), allowing the user to print the report to a mainframe or PC printer, download the report and so forth. After exiting the output manager, the user sees the screen below:

At this point, the user can save the results of the query, either as data or as a stored view. SCL is used to ensure that the name given by the user at this point does not already exist. At some later time, the user can select the permanent view for a report (see above). Since it is a view, s/he can be sure that the data will be current whenever it is used.

System Overview

Below is a list of the programs involved in the application, copied from the PROC BUILD window:

For example, the SQLQUERY program calls the SQLWHERE program, to generate the WHERE clause. The SQLWHERE program calls, in turn, the SQLSEL2 program, which provides the user with a selection list of variables.

A few examples of code can demonstrate the control you have.

At some later time, the user can select the permanent view for a report (see above). Since it is a view, s/he can be sure that the data will be current whenever it is used.

System Overview

Below is a list of the programs involved in the application, copied from the PROC BUILD window:

BUILD: DIRECTORY BUILD.PROSQL (E) Command = = = >

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEARLOG</td>
<td>Clear LOG For DESCRIBE Program</td>
</tr>
<tr>
<td>DATASETS</td>
<td>Publish Contents Of Data Sets</td>
</tr>
<tr>
<td>DESCRIBE</td>
<td>Write VIEW Description to SASPRT</td>
</tr>
<tr>
<td>EDQUERY</td>
<td>Edit SASUSER.QUERIES Data Set</td>
</tr>
<tr>
<td>PMENU</td>
<td>PMENU program for SQL Query Facility</td>
</tr>
<tr>
<td>PREVIEW</td>
<td>Preview SQL Statement For Editing</td>
</tr>
<tr>
<td>PRIMARY</td>
<td>Entry Program For SQL Query Facility</td>
</tr>
<tr>
<td>QUERIES</td>
<td>SQL Queries Selection List Program</td>
</tr>
<tr>
<td>RUNVIEW</td>
<td>Run a stored SQL view</td>
</tr>
<tr>
<td>SQLCOLS</td>
<td>Called to generate computed columns</td>
</tr>
<tr>
<td>SQLHAVING</td>
<td>Called to generate HAVING clause</td>
</tr>
<tr>
<td>SQLLD</td>
<td>Preview &quot;Old&quot; SQL Statement For Editing</td>
</tr>
<tr>
<td>SQLPREV</td>
<td>Base program for User Generated Query</td>
</tr>
<tr>
<td>SQLQUERY</td>
<td>Base program for SQL query generation</td>
</tr>
<tr>
<td>SQLRENAM</td>
<td>Called to rename columns to 8 chars</td>
</tr>
<tr>
<td>SQLSAVE</td>
<td>Called to Save/Delete SQL view</td>
</tr>
<tr>
<td>SQLSELECT</td>
<td>Called to select columns from tables</td>
</tr>
<tr>
<td>SQLSEL2</td>
<td>Called by SQLWHERE to select columns</td>
</tr>
<tr>
<td>SQLSEL3</td>
<td>Called by SQLHAVING to select columns</td>
</tr>
<tr>
<td>SQLSEL4</td>
<td>Called by SQLCOLS to select columns</td>
</tr>
<tr>
<td>SQLWHERE</td>
<td>Called to generate WHERE clause</td>
</tr>
<tr>
<td>VARLIST</td>
<td>Variable Selection List Program</td>
</tr>
</tbody>
</table>

One thing you notice in looking at this partial catalog directory is the number of instances of one program calling another.

```sas
BUILD: SOURCE SQLWHERE.PROGRAM (E)
0103 if upcase(p2)='LIKE' then do;
0104 if index(whr12,'%')=0 then
0105 whr12 = trim(compress(whr12,'%'));
0106 end;
/* WHEN USING 'LIKE' THE VALUE NEEDS TO BE /
* FOLLOWED BY THE /
*/ PERCENT SIGN, AS A WILD CARD */
0108 wclaus1=lp1ch | whr1 | '*' | a1 | '*' | whr2 |
| ' | andor1 | lp1ch; 0109 wclaus2=lp2ch | whr3 | '*' | d1 | '*' | whr4 |
| ' | andor2 | lp2ch; 0110 wclaus3=lp3ch | whr5 | '*' | g1 | '*' | whr6 |
| ' | andor3 | lp3ch; 0111 wclaus4=lp4ch | whr7 | '*' | l1 | '*' | whr8 |
| ' | andor4 | lp4ch; 0112 wclaus5=lp5ch | whr9 | '*' | m1 | '*' | whr10 |
| ' | andor5 | lp5ch; 0113 wclaus6=lp6ch | whr11 | '*' | p1 | '*' | whr12 |
| ' | andor6 | lp6ch; 0114 _status_ ='H'; 0115 RETURN;
/* WCLAUS1 -- WCLAUS6 WERE PASSED (EMPTY) TO THIS
*/ PROGRAM FROM THE SQLQUERY
/* PROGRAM. NOW THEY ARE /
*/ PASSED BACK WITH REAL VALUES */
0118 GETVARS;
0119 call display(‘sqlsel2.program’,dsid.varnam);
0120 RETURN;
/* THE CALLED PROGRAM DISPLAYS A SELECTION LIST /
*/ FROM WHICH VARIABLES ARE SELECTED AND /
*/ PASSED BACK TO THIS PROGRAM */
```

BUILD: SOURCE SQLSEL2.PROGRAM (E)
00001 entry $dsid 8
00002 varnam $ 17;
/* PARAMETERS PASSED BY SQLWHERE PROGRAMS /
*/ TO BE RETURNED WITH USER'S SELECTIONS
00003 INIT;
00004 call set(dsid);
/* ASSOCIATES DATA SET VARS WITH SCREEN VARS */
00005 vnum = varnum($dsid,"column");
00006 call setrow(0,1,'x',"y");
/* SETS UP OPTIONS FOR EXTENDED TABLE */
00007 RETURN;
00008
00009 MAIN;
00010 RETURN;
00011
00012 TERM;
00013 row=selected(1);
Below is part of the ATTR for the window vars in the PREVIEW program, in which the user determines whether to view/edit the query code or run it. Some of the code from that program and another precede the ATTR display.

BUILD: SOURCE PREVIEW.PROGRAM (E)

00001 entry gonot 8;
00003 INIT:

00004 control always;
00005 gono=' ';
00006 gonot=' '; /* WINDOW VAR AND PARAM. SET TO MISSING */
00007 RETURN;

00008 MAIN:

00009 if gono='YES' then do;
00010 gonot='YES';
00011 goto TERM;
00012 end;
00013
00014 if gono='NO' then do;

00015 call wregion(1,1,24,80);
00016 rc=preview('edit',

'The following code is from the program which asks the user if the view which has just been executed should be saved and, if so, as what.

BUILD: SOURCE SQLSAVE.PROGRAM (E)

00001 entry qid 8;
00003 INIT:

00004 control asis always;
00005 call set(qid);
00006 RETURN;

00008 MAIN:

00009 if schoice='YES' and (qdesc='') or qname=' ' or v=' ') then do;

00010 mess='Enter 35 char description then name';
00011 vd='Enter D for Data, V for View';
00012 unprotected qdesc qname;

00013 if qdesc=' ' then cursor qdesc;
00014 else cursor qname;

00015 end;

00016 else if schoice='NO' then do;

00017 _status_ = 'C';
00018 goto TERM;

00019 end;

00020 else if word(l,'l')='nomore' then do;

00021 _statu5_= 'C';
00022 goto TERM;

00023 end;

00024 else if word(l,'l')='run' then do;

00025 if schoice='YES' and (qdesc='') and qname=' ' then do;

00026 then goto TERM;

00027 end;

00028 RETURN;

00030 TERM:

00031 if _status_ = 'C' then do;

00032 _status_ = 'R';
00033 RETURN;

00034 end;

00035 if schoice='YES' and qdesc=' ' then do;

00036 _msg_ = 'Enter a 35 char. description';
00037 cursor qdesc;

00038 _status_ = 'R';
00039 RETURN;

00040 end;

00041 else if schoice='YES' and qname=' ' then do;

00042 _msg_ = 'Enter an 8 char. name';
00043 cursor qname;

00044 _status_ = 'R';
00045 RETURN;

00046 end;
Conclusion

The SQL query facility is the most complex of a series of utilities written to allow the end user to be more productive, on her or his own. By developing such utilities, we have been able to reduce the number of canned systems that must be written and maintained by programmers.

In turn, we stress the need for the users to know their data. In canned applications, pushing buttons or typing menu selections is enough. The options are limited; the user cannot stray far off the track. Nor, however, can they utilize their own knowledge of their business and their data, in real time. With end user tools, such as the SQL query application, the user has great freedom, but that freedom will either be wasted or abused, unless the user understands the data she is working with.

The expansion of the number of such utilities is the direct result of the advent of Version 6 of the SAS system, as it provides tools which make possible projects which we simply would not touch with Version 5 of SAS. If SAS -- and SAS programmers -- are to compete with other tools, especially as we move from mainframes to micros and LANs, the development of end user tools must be a primary activity. The SAS Institute obviously recognizes this, with the development of SAS/ASSIST. As programmers in our own organizations, we must expand on that foundation, developing tools which give our users the same power and comfort level they are used to with off the shelf PC packages. Systems we develop must not only be efficient and effective, relative to our organizations' needs, they must be whizzy.

For More Information

The author may be contacted at:
Intel Corporation
5200 N.E. Elam Young Pkwy.
HF2-23
Hillsboro, OR 97124
(503) 696-7414

Trademarks

SAS, SAS/AF, SAS/ASSIST, SAS/FSP are registered trademarks or trademarks of the SAS Institute Inc. in the USA and other countries.