Paper 33

Building a 'Pre-subsetting' User Interface for Multidimensional Reports

Dan J. Kirpes, Fireman's Fund Insurance Companies, Novato, CA

ABSTRACT

The scope of this paper is to introduce and give an example of a user interface (SAS/AF® software) subsetting a multidimensional database (MDDB) prior to presenting it to the Multidimensional Report. It will show how to develop a FRAME using list boxes to define the subsets and show the passing of SCL list information to the SAS/EIS® software Multidimensional Report via a method override.

Skill Level: Advanced programmers

INTRODUCTION

In recent years, a new reporting tool for business decision support has evolved. The multidimensional data base is a storage technique that allows summarized data to be stored by hierarchies (classified by ranking order). This storage method has allowed businesses to quickly query the data using graphical interface software. This software allows drilling down/up in the predefined hierarchies. The SAS/EIS® software product provides both MDDB table creation and multidimensional reporting ability.

PROBLEM

When using SAS/MDDB™ and SAS/EIS® software version 6.12, it is possible to quickly build a MDDB and a multidimensional report. The biggest problem is reporting too much data; data that is not appropriate or needed. One question that arises when building the MDDB and the reports is how many MDDB's should be created. My approach was to keep everything as simple as possible by developing only one MDDB per logical data grouping and not by hierarchy. I was constrained to a monthly static MDDB and I was not able to develop the MDDB dynamically prior to each report. A number of reports were created from this one MDDB, but the problem was that the reports contained a lot of data, too much data. One way to down size the scope of a report is to use the option to "Subset by Dimension" while "displaying" the multidimensional report. This approach works well, but it is preferred to make the necessary selections (subsets) in advance of displaying the report.

SOLUTION

My approach to solving this dilemma of too much data in the report was to build a graphical user interface using a FRAME entry. This FRAME or selection criteria window used "push buttons" and "list boxes" to make the necessary selections within the classes that had been defined in the MDDB. The list boxes were populated with data from the MDDB. Once the selections were made, the information was stored in SCL lists. The address of the selection list was then passed to the multidimensional report so that the subsetting would take place prior to displaying the report. A method override for the multidimensional report was used to facilitate the subsetting.

SUBSETTING OPTIONS FOR MULTIDIMENSIONAL REPORTS

1. The first option is to create many multidimensional databases and multidimensional reports that are appropriate to your needs. This option requires a lot of work to develop the mddbs and the reports.
2. A popular option is to dynamically subset the detail data, create a new mddb, and display the report. This requires that the detail data is readily available and that SAS/MDDB™ software is licensed in the appropriate environment. A disadvantage may be that the response to display the report may be slow.
3. Another option is to display the multidimensional report and then to "Subset by Dimension". When the report first appears, it presents you with all of the data. You can right-mouse click in the report to pop open a menu list. The "Subset by Dimension" option allows you to select the dimension variable and then select the appropriate variable value. Click OK to run the report.
4. Another option is to "Subset by Dimension" BEFORE presenting the report. This can be easily accomplished by developing an SCL entry to override the '_POSTINIT_' method. This paper includes the program POSTINIX.SCL to register in the multidimensional report and "When to Run" is set to AFTER. This option works very well and is very easy to implement.
5. The final option consists of (a) a fully developed mddb, (b) a multidimensional report, and a custom user interface. The custom user interface executes before calling the multidimensional report. This paper provides the details for implementing this user interface.

CUSTOM PRE-SUBSETTING USER INTERFACE:

The user interface starts with a report menu selection for the requested report. The hierarchy, mddb file, and multidimensional report information are passed to the SELECT.FRAME program. The SELECT program can be used for any database, hierarchy, or report. This FRAME entry is modeled after the "Subset by Dimension" frame. It uses 3 list boxes: dimension, available, and selected. The frame also includes 5 push buttons: add, remove, reset, run report, and quit. The dimension list box is populated with the CLASS variables from the MDDB. The 'available' list box is populated with the values from a selected class variable. By selecting one or more values in the 'available' list box and then pressing the 'add' push button, the values are added to the 'selected' list box. If necessary, the selected variables can be moved back to the available list box.

During this processing of 'available' and 'selected' values, the program creates a new mddb table and checks for new available values. This checking process ensures that a valid report will be created.

Pressing the 'Run Report' push button calls the multidimensional report. The hierarchy and the 'selected ' SCL list are used to create a pre-subsetted report. Drill-downs are still possible in this report.

The GETAVAIL.SCL is code that is used to override the '_get_available_values_' method and is executed from within the SELECT program to determine the new available values after remaking the mddb table.

The OBJLAB.SCL is code that is used to override the '_object_label_' method. This method determines the titles within the multidimensional report.

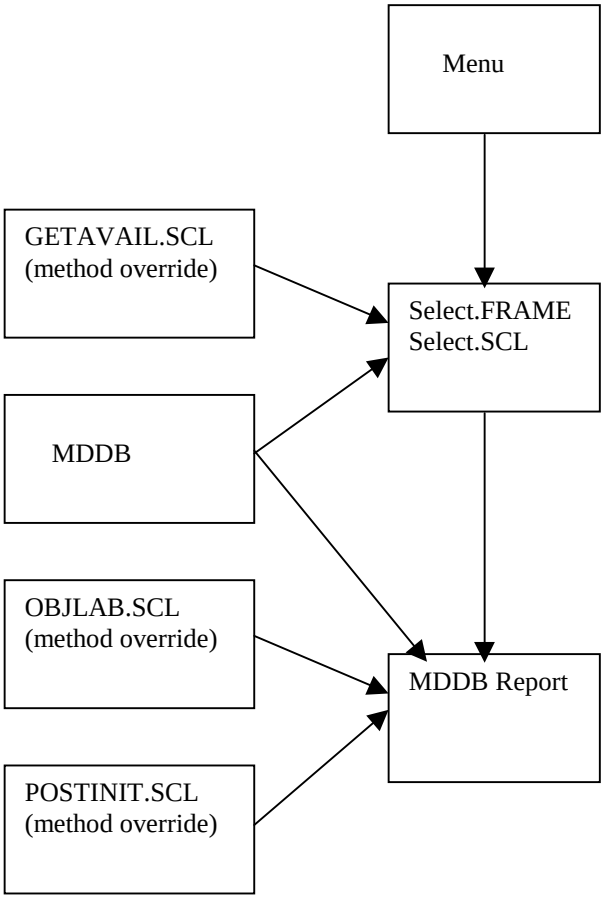
The POSTINIT.SCL code overrides the '_postinit_' method of the multidimensional report. This code executes prior to displaying the multidimensional report. The hierarchy and subset lists are used to prepare the report.

PROCESS FLOW FOR MDDB SUBSETTING

The following chart shows the process flow for subsetting a MDDB prior to displaying a multidimensional report. As can be

seen in the chart, the MDDB provides data to both the selection program and to the report. Also, the chart shows the relationship to the method overrides.
Following the flow chart is the program code and method overrides.

SUBSETTING PROCESS FLOW CHART



GETAVAIL.SCL

```
getavail: method current_dim $ wherelist
valueslist 8
          optional=totals_flag $ pdsname $
17;
   call
super(_self_,'_get_available_values_',
current_dim,wherelist,
        valueslist,totals_flag);
   return;
endmethod;
```

OBJLAB.SCL

```
*************************************;
*  Override OBJLAB method, When to run AFTER
*;
*************************************;
length title1 title2 title3 title4 $ 80;
objlab:
  method optional= cmdline $ rc 8;
  frameid=getnitemn(_self_,'_FRAME_');
  call send(frameid,'_get_widget_',
'obj1',objectid);
  titleid=getnitemn(objectid,'title');
  call send(titleid,'_is_hidden_',hidden);
  if hidden then return;
  call send(titleid,'_get_widget_',
'title1',title1id);
  call send(titleid,'_get_widget_',
'title2',title2id);
  call send(titleid,'_get_widget_',
'title3',title3id);
  call send(titleid,'_get_widget_',
'title4',title4id);
  call send(title1id,'_get_text_',title1);
  modelid=getnitemn(_self_,'modelid');
  sublist=makelist();
  call send(modelid,'_get_subsets_',
sublist);
  where='';
  do i=1 to listlen(sublist);
     catvarlist=getiteml(sublist,i);
     namelist=nameitem(sublist,i);
     cat_str = '';
     if listlen(catvarlist)>0 then do;
        do j=1 to listlen(catvarlist);

value=left(getitemc(catvarlist,j));
           if j=1 then cat_str=trim(value);
           else cat_str= cat_str|| ',' ||
trim(value);
        end;
     end;
     if listlen(catvarlist) then do;
        where=where||' '||namelist||'='
           ||cat_str;  /* <-- tmb added */
     end;
  end;
  title2 = where;
   call send(title2id,'_set_text_', title2);
   if length(where) >64 then do;
      title3 = substr(where,65);
      call send(title3id,'_set_text_',
title3);
   end;
   else do;
      title3 = ' ';
      call send(title3id,'_set_text_',
title3);
   end;
   if length(where) >128 then do;
      title4 = substr(where,129);
      call send(title4id,'_set_text_',
title4);
   end;
   else do;
      title4 = ' ';
      call send(title4id,'_set_text_',
title4);
   end;
  rc=dellist(sublist);
endmethod;
```

POSTINIT.SCL

```
length _method_  $ 40;
length msg       $ 200;
length hierarchy $ 40;
length val $16 format $13 label str $80  s
tablename $40;
postinit:
   method;
*************************************;
* Override POSTINIT method, When to run AFTER
*;
*************************************;
   modelid  = getnitemn(_self_,'MODELID');
   hierlist = makelist();
   call send(modelid,'_get_hierarchies_l_',
hierlist);
*************************************;
*   Specify HIERARCHY                *;
```

```
*****************************************;
   hiername  = symget('hiername');
   hierlist2 = getniteml(hierlist,hiername);
*****************************************;
*   Set the subsets list                *;
*****************************************;
   sellist = symget('sellist');
   sublist = makelist();
   sublist = copylist(sellist,'RECURSIVELY');
*****************************************;
* display the report                    *;
*****************************************;
   call
   send(modelid,'_set_level_and_subset_',
   hierlist2,sublist);
endmethod;
```

SELECT.SCL

```
%eis; %eismeth;
*****************************************;
*  the EIS.SAS macro in SAS\EIS\SASMACRO *;
*  needs to be changed to the following: *;
*****************************************;
/*  %macro eismeth ;
    length kbname $ 8 kbdsid 8 applds $ 8
    appldsid 8 text1 text2 $ 200 num1 num2 8
    %mend; */

entry mddbfile $ 17 hiername $ 40 mddbrpt $
40;
length text         $ 16;
length value        $ 40;
length tblexist     $ 1;
INIT:
   control always;
   _msg_ = 'Please make selections.';
*****************************************;
* get dimensions and copy to list box   *;
*****************************************;
   classid =
loadclass('sashelp.eis.emddb_m.class');
   mddbid  = instance(classid);
   call send(mddbid,'_SET_MDDB_TABLE_',
mddbfile);
   hlist = makelist();
   call send(mddbid,'_GET_HIERARCHIES_',
hlist);
   dimsubid = getniteml(hlist,'CLASS');
   dim_lbox = copylist(dimsubid,'N');
   n = listlen(dim_lbox);
*****************************************;
* Initialize method override            *;
*****************************************;
   call send(mddbid,'_set_instance_method_',
'_get_available_values_',

'sugi.sugiappl.getavail.scl','getavail','over
ride');
*****************************************;
*  make lists                           *;
*****************************************;
   avl_lbox = makelist();
   sel_lbox = makelist();
   link reset;
RETURN;

*****************************************;
*   set the values for the list boxes   *;
*****************************************;
RESET:
   listid   = makelist();
   avllist  = makelist();
   sellist  = makelist();
*****************************************;
*  get the class values from the MDDB   *;
*  create list with original values     *;
*****************************************;
   do i = 1 to n;
      var = getitemc(dimsubid,i);
      a_list = makelist();
      call send(mddbid,
'_GET_CLASS_VALUES_',var,a_list);
      a_list = delitem(a_list,-1);
      rc=insertl(listid,a_list,-1,var);
   end;
*****************************************;
*  get the class values from the MDDB   *;
*  create list with available values    *;
*****************************************;
   do i = 1 to n;
      var = getitemc(dimsubid,i);
      b_list = makelist();
      call send(mddbid,'_GET_CLASS_VALUES_',
var,b_list);
      b_list = delitem(b_list,-1);
      rc=insertl(avllist,b_list,-1,var);
   end;
   *****************************************;
*  get the class values from the MDDB   *;
*  create list with selected values     *;
*****************************************;
   do i = 1 to n;
      var = getitemc(dimsubid,i);
      c_list = makelist();
      call send(mddbid,'_GET_CLASS_VALUES_',
var,c_list);
      c_list = delitem(c_list,-1);
      rc=insertl(sellist,c_list,-1,var);
   end;
   *****************************************;
*  Clear values in select list          *;
*****************************************;
   do i = 1 to n;
      text = getitemc(dimsubid,i);
      selsubid = getniteml(sellist,text);
      rc = clearlist(selsubid);
   end;
   *****************************************;
*  initialize list boxes                *;
*****************************************;
   call notify('DIMENSN','_get_last_sel_',
row,is_sel,text);
   if row gt 0 then
      call
notify('DIMENSN','_unselect_all_');
   call notify('AVAIL','_delete_all_');
   call notify('SELECT','_delete_all_');
   call notify('AVAIL','_gray_');
   call notify('SELECT','_gray_');
RETURN;

*****************************************;
*   determine values for list boxes     *;
*****************************************;
DIMENSN:
*****************************************;
* get selected values for selected class
*;
*****************************************;
   call notify('DIMENSN','_get_value_',
valueid);
   textid = getniteml(valueid,'text');
   dimvalue = getitemc(textid,1);
*****************************************;
* populate Available list box with values *;
*****************************************;
   call notify('AVAIL','_delete_all_');
   call
notify('AVAIL','_get_value_',availid);
   avsubid = getniteml(availid,'all');
   rc = clearlist(avsubid);
   avllstid = getniteml(avllist,dimvalue);
```

```
   do i = 1 to listlen(avllstid);
      text = getitemc(avllstid,i);
      call notify('AVAIL','_ADD_',text,-1);
   end;
   call notify('AVAIL','_ungray_');
*****************************************;
*  populate Select list box with values *;
*****************************************;
   call notify('SELECT','_delete_all_');
   call
notify('SELECT','_get_value_',selectid);
   selsubid = getniteml(selectid,'all');
   rc = clearlist(selsubid);
   sellstid = getniteml(sellist,dimvalue);
   do i = 1 to listlen(sellstid);
      text = getitemc(sellstid,i);
      call notify('SELECT','_ADD_',text,-1);
   end;
   call notify('SELECT','_ungray_');
RETURN;

*****************************************;
*  move items from available to selected
*;
*****************************************;
ADD:
*****************************************;
*   populate Selected list box with values
*;
*****************************************;
   call notify('AVAIL','_get_value_',
selectid);
   avlsubid = getniteml(selectid,'TEXT');
   do i = 1 to listlen(avlsubid);
      value = getitemc(avlsubid,i);
      call notify('SELECT','_ADD_',value,-1);
   end;
*****************************************;
*  remove selection from Available list box
*;
*****************************************;
   call notify('AVAIL','_get_maxrow_',
max_row);
   do i = max_row to 1 by -1;
      call notify('AVAIL','_issel_',i,
issel);
      if issel gt 0 then do;
         call notify('AVAIL','_get_text_',
i,value);
         call notify('AVAIL',
'_delete_text_',value);
      end;
   end;
   link snapshot;
RETURN;

*****************************************;
*  move items from selected to available*;
*****************************************;
REMOVE:
*****************************************;
* populate Available list box with values*;
*****************************************;
   call notify('SELECT','_get_value_',
selectid);
   avlsubid = getniteml(selectid,'TEXT');
   do i = 1 to listlen(avlsubid);
      value = getitemc(avlsubid,i);
      call notify('AVAIL','_ADD_',value,-1);
   end;
*****************************************;
* remove selection from Selected list box*;
*****************************************;
   call notify('SELECT','_get_maxrow_',
max_row);
   do i = max_row to 1 by -1;
      call notify('SELECT','_issel_',
i,issel);
      if issel gt 0 then do;
         call notify('SELECT',
'_get_text_',i,value);
         call notify('SELECT',
'_delete_text_',value);
      end;
   end;
   link snapshot;
RETURN;

SNAPSHOT:
*****************************************;
*  take a snapshot of the available listbox
*;
*  and save the values in the avllist.  *;
*****************************************;
   avlsubid = getniteml(avllist,dimvalue);
   rc = clearlist(avlsubid);
   call notify('AVAIL','_get_maxrow_', rows);
   if rows gt 0 then do;
      do i = 1 to rows;
         call notify('AVAIL','_get_text_',
i,text);
         rc = insertc(avlsubid,text,-1);
      end;
   end;
   *****************************************;
*  take a snapshot of the selected listbox
*;
*  and save the values in the sellist.  *;
*****************************************;
   selsubid = getniteml(sellist,dimvalue);
   rc = clearlist(selsubid);
   call notify('SELECT','_get_maxrow_',
rows);
   if rows gt 0 then do;
      do i = 1 to rows;
         call notify('SELECT',
'_get_text_',i,text);
         rc = insertc(selsubid,text,-1);
      end;
   end;
   link newtable;
RETURN;

NEWTABLE:
*****************************************;
*  rebuild where clause                 *;
*****************************************;
   wherelst=makelist();
   wherelst =
copylist(sellist,'RECURSIVELY');
*****************************************;
*  create class list                    *;
*****************************************;
   classlst=makelist();
   classlst = copylist(dimsubid);
*****************************************;
*  delete table if it exists            *;
*****************************************;
   call send(mddbid,'_table_exist_','NEW',
tblexist);
   if tblexist = 'Y' then
      call send(mddbid,'_delete_table_',
'NEW');
*****************************************;
*  make new table                       *;
*****************************************;
   call send(mddbid,'_make_table_',classlst,
'NEW',wherelst,'Y','N');
*****************************************;
*  clear available list                 *;
*****************************************;
   do i = 1 to n;
      text = getitemc(dimsubid,i);
```

```
      avlsubid = getniteml(avllist,text);
      rc = clearlist(avlsubid);
   end;
*****************************************;
*  get subset values and                *;
*  recreate available list              *;
*****************************************;
   do i = 1 to n;
      avlvalue = getitemc(dimsubid,i);
      values=makelist();
      call
send(mddbid,'_get_available_values_',
avlvalue,wherelst,values,'N',mddbfile);
      avlsubid = getniteml(avllist,avlvalue);
      do j = 1 to listlen(values);
         value = getitemc(values,j);
         rc = insertc(avlsubid,value,-1);
      end;
   end;
*****************************************;
*  remove values from the available list*;
*  that are in the selected list        *;
*****************************************;
   do i = 1 to n;
      avlclass = getiteml(avllist,i);
      selclass = getiteml(sellist,i);
      do j = listlen(avlclass) to 1 by -1;
         avlvalue = getitemc(avlclass,j);
         do k = 1 to listlen(selclass);
            selvalue = getitemc(selclass,k);
            if avlvalue = selvalue then
               avlsubid =
delitem(avlclass,j);
         end;
      end;
   end;
RETURN;

RUNRPT:
   %global sellist hiername;
   call symput('hiername',hiername);
   call symput('sellist',sellist);
*****************************************;
* create multidimensional report using
POSTINIT method override *;
*****************************************;
   call display('sashelp.eis.runeis.program',
&eparmlst,mddbrpt,rc);
RETURN;

QUIT:
   link term;
RETURN;

TERM:
   rc = dellist(listid);
   rc = dellist(avllist);
   rc = dellist(sellist);
   rc = dellist(a_list);
   rc = dellist(b_list);
   rc = dellist(c_list);
   rc = dellist(avl_lbox);
   rc = dellist(sel_lbox);
   rc = dellist(dim_lbox);
   rc = dellist(hlist);
   rc = dellist(wherelst);
   rc = dellist(classlst);
   _status_ = 'H';
RETURN;
```

POSTINIX.SCL (OPTION 4 FROM PAGE 1)

```
length _method_  $ 40;
length msg       $ 200;
length hierarchy $ 40;
length val $16 format $13 label str $80
 s tablename $40;

postinit:
   method;
*****************************************;
*  Override POSTINIT method, When to run
AFTER                                    *;
*****************************************;
   modelid=getnitemn(_self_,'MODELID');
   call send(modelid,'_select_subset_');
endmethod;
```

CONCLUSION

By subsetting the MDDB prior to displaying the multidimensional report, you present only the relevant or needed data. All of this code has been assembled into a demonstration application and is available upon request.

ACKNOWLEDGMENTS

The author would like to express appreciation to the Full Screen Products Division of SAS Technical Support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Dan J. Kirpes
Fireman's Fund Insurance Company
777 San Marin Drive
Novato, CA 94998
Work Phone: (415) 899-3561
Fax: (415) 899-6048
Email: dkirpes@ffic.com