Fancy Arrays: Fun and Useful for Analytical Data Marts.

ABSTRACT

The common notion is that the usage of fancy multidimensional arrays makes SAS programs difficult to read and understand. The truth is that they make work easier when used for the aggregation of variables from Analytical Data Marts, which are naturally organized as rectangular tables. This paper is intended to show how these scary creatures can be fun and easy to use.

INTRODUCTION

Multidimensional arrays are useful when your application requires the usage of data organized in rectangular tables (Excel spreadsheets for instance) with subgroups that can be defined as an additional 3rd dimension (or subs of subgroups for the 4th dimension, etc.). Arrays make it simple to initialize and directly access any element (variable) of a table for update or data manipulation.

The paper will illustrate how to use multidimensional arrays for Analytical Data Marts (Marketing and Medical Research) to aggregate data from transactional tables into different categories of Analytical Data organized in tables.

In Analytical Data Marts we usually want to group variables in 2-3 dimensional tables by defined indexes (table dimensions) which correspond to group categories such as product, media, membership status, duration since open date, etc. 2-3 dimensional tables defined as Multidimensional arrays in SAS programs reduce the amount of code that you have to write in order to initialize or set variables to constant values, access different variables in the table according to the variable categories (which can be mapped with PROC Format to the row and column of the variable location in the table: table indexes), and ultimately make your SAS program look extremely compact and easy to read (obviously if you understand Multidimensional Array processing).

However I would not recommend using Multidimensional Arrays just to make your program look fancy and cool.

FUN BUILDING ANALYTICAL DATA MARTS USING FANCY ARRAYS

It makes perfect sense to use fancy arrays for building Analytical Data Marts (which usually include hundreds of transformed variables derived from the aggregation of Transactional Tables).

We’ll define an Analytical Data Mart as a collection of transformed variables derived from Transactional Historical Data Aggregation usually coming from a company’s Data Warehouse or the Transactional Data Base.

Designing an Analytical Data Mart includes the following steps:
- Exploratory Data Analysis of Transactional Historical Data Tables (such as Marketing Promotional Data), which represents the Historical Data for a long period of time (1-5+ years);
- Definition of analytically significant and meaningful groups for transaction variable aggregation;
- Design of Data Mart transformed variables and methods of Data aggregation;
- Design variable naming convention for Data Mart variables (properly designed naming convention helps: easy identify variable groups, avoid variable redundancy, classify hundreds of transformed variables in well defined categories);
- Design the mapping of group variables into table indexes (Set of Formats);
- Coding and testing of Data Mart Aggregation process.

We’ll show how to design the Analytical Data Mart using examples of 2 and 3 dimensional tables of transformed variables.

Let’s start by defining 2-dimensional table of the customer’s Marketing Transactional Table aggregation by Marketing Media and Duration since the Account Open Date. Here is a Marketing Transactional Table layout:

<table>
<thead>
<tr>
<th>T_CUSTID</th>
<th></th>
<th>Customer ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_MEDIA</td>
<td></td>
<td>Marketing Media</td>
</tr>
<tr>
<td>T_OPENDT</td>
<td></td>
<td>Account Open Date</td>
</tr>
<tr>
<td>T_STATUS</td>
<td></td>
<td>Account Status (A-Active, C-Cancel)</td>
</tr>
</tbody>
</table>

Now we’ll define the Analysis End (Cut) Date, which is usually a Transactional table’s query pull date from Data Warehouse or the end date in query if analysis supposed to be done for the past period. The Analysis Cut Date could be a variable date if the analysis defined as an up to 12 months since the account’s open date:

| G_CUT_DT   |        | Analysis Cut Date    |

Let’s assume that we already input and validated the table’s data and defined significant groups for Marketing Media as:

- DM - Direct Marketing
- TM - Telemarketing
- ON - Online Marketing
- OT - Other Marketing Channels

And Duration from Account Open Date till Analysis Cut Date as:

<table>
<thead>
<tr>
<th>Duration</th>
<th></th>
<th>months</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-91 days</td>
<td></td>
<td>0-3 months</td>
</tr>
<tr>
<td>92-182 days</td>
<td></td>
<td>3-6 months</td>
</tr>
<tr>
<td>183-273 days</td>
<td></td>
<td>6-9 months</td>
</tr>
<tr>
<td>274-365 days</td>
<td></td>
<td>9-12 months</td>
</tr>
<tr>
<td>366+ days</td>
<td></td>
<td>12+ months</td>
</tr>
</tbody>
</table>

Now we can define the Data Mart’s transformed variable table:
It’s time to design the Variable’s Naming Convention. We’ll use 8-byte long variable names for several reasons; one of them is that some of the modeling software packages require the creation of Xport files in which 8-byte variable names have to be exported from SAS Datasets like in SAS V6.

Each of 8 bytes in a variable name for a sizable Data Mart has its own meaning, but for our example we’ll define only 4 bytes:

1st byte is a Data Mart’s Table Indicator: C - for Customer Table;
2nd byte is a variable’s type: N – Numeric Total, D – Duration, B – Binary, etc.;
6th byte is a Media Group: D – DM, T – TM, O – Online, M – Other;
8th byte is Duration Group: 1 – 0-3, 2 – 3-6, 3 – 6-9, 4 – 9-12, 5 – 12+ months;
Unused bytes are _.

Now we can easily distinguish variables in the Data Mart’s Media by Duration table by using variable names:

<table>
<thead>
<tr>
<th>Media /Duration</th>
<th>0-3 mnths</th>
<th>3-6 mnths</th>
<th>6-9 mnths</th>
<th>9-12 mnths</th>
<th>12+ mnths</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM</td>
<td>CN_D_1</td>
<td>CN_D_2</td>
<td>CN_D_3</td>
<td>CN_D_4</td>
<td>CN_D_5</td>
</tr>
<tr>
<td>TM</td>
<td>CN_T_1</td>
<td>CN_T_2</td>
<td>CN_T_3</td>
<td>CN_T_4</td>
<td>CN_T_5</td>
</tr>
<tr>
<td>ON</td>
<td>CN_O_1</td>
<td>CN_O_2</td>
<td>CN_O_3</td>
<td>CN_O_4</td>
<td>CN_O_5</td>
</tr>
<tr>
<td>OT</td>
<td>CN_M_1</td>
<td>CN_M_2</td>
<td>CN_M_3</td>
<td>CN_M_4</td>
<td>CN_M_5</td>
</tr>
</tbody>
</table>

The Data Mart’s Table of transformed variables is defined and we’ll create formats to map the Table’s row and column groups into indexes (row and column numbers):

```
Proc format;
  Value $mediaf  “DM”=”1”
       “TM”=”2”
       “ON”=”3”
       Other=”4”
;
  Value opndurf  0-91   =”1”
                  92-182 =”2”
                  183-273 =”3”
                  274-365 =”4”
                  366-high=”5”
                  Other  =”0”
;
Run;
```

HANDLING DATA MART TABLES WITH FANCY ARRAYS
In SAS, the Data Mart Customer Table will be defined as a 2-dimensional array with a Macro list of variables:

```
%macro c_totmkt;
CN___D_1 CN___D_2 CN___D_3 CN___D_4 CN___D_5
CN___T_1 CN___T_2 CN___T_3 CN___T_4 CN___T_5
CN___O_1 CN___O_2 CN___O_3 CN___O_4 CN___O_5
CN___M_1 CN___M_2 CN___M_3 CN___M_4 CN___M_5
%mend  c_totmkt;
```

Now we’ll define table’s dimensions and Analysis Cut Date:

```
%let g_meddim=4;
%let g_durdim=5;
%let g_cut_dt="07aug2010"d;
```

In the Data Step we’ll define the 2-dimensional array:

```
Array c_tmkt{& g_meddim,& g_durdim} %c_totmkt;
```

And initialize it using a nested do loop for the aggregation of totals:

```
Do j=1 to & g_meddim;
   Do k=1 to & g_durdim;
      c_tmkt{j,k}=0;
   end;
end;
```

In the Transaction Table’s aggregation, each observation can add only 1 to 1 of 20 variables of Data Mart’s Customer Table. The table index setup points out (access) the variable that has to be aggregated:

```
_i_med=put(upcase(t_media, $mediaf.))+0;
_i_dur=put(&g_cut_dt - t_opendt, opndurf.);
```

The next statement validates the open date’s duration (transactions with open date after Analysis Cut Date won’t be aggregated) and access directly the variable in Data Mart’s table for aggregation:

```
If _i_dur>=0 then c_tmkt{_i_med, _i_dur}+1;
```

The whole program that creates the Customer Data Mart table is so small that it can be listed in the paper like a fragment:

```
Data custmkt.mkttbl(keep=t_custid %c_totmkt);
Length %c_totmkt 4;
Retain %c_totmkt;
```
Array c_tmkt{& g_meddim, & g_durdim} %c_totmkt;
Set tranmkt.thans end=eof;
By t_custid;
If first.t_custid then do;
   Do j=1 to & g_meddim;
      Do k=1 to & g_durdim;
      c_tmkt{j,k}=0;
   end;
end;
_i_med=put(upcase(t_media, $mediaf.))+0;
_i_dur=put(&g_cut_dt - t_opendt, opndurf.)+0;
If _i_dur>=0 then c_tmkt{_i_med, _i_dur}+1;
If last.t_custid then do;
   Output custmkt.mkttbl;
End;
Label   cn___d_1="# of direct mail accounts opened 3 mns prior to cut dt"
   cn___d_2="# of direct mail accounts opened 3-6 mns prior to cut dt"
; 
Run;

We purposely skipped comments before lines of code that have to be listed prior to some statements to explain each program’s fragment.

A couple of necessary comments:
- A length statement has been used for Data Mart variables in order to decrease output observation length (reasonable total number of accounts cannot be bigger than 99 and length 4 is a bit excessive – 3 would be enough, to count some outliers);
- We assumed that the transaction table (tranmkt.thans SAS dataset) was previously sorted by t_custid. The common aggregation technique was used with first. and last. pointers for ordered SAS DS in SAS Data Step.

3-DIMENSIONAL DATA MART’S TABLE – NO PROBLEM

We have successfully created the program that aggregates the 2-dimensional table. Now let’s extend the Data Mart’s table to a 3-dimensional table (with block of rows defining 1st dimension, a row defining the 2nd dimension and a column defining the 3rd dimension). For an additional dimension we’ll use the customer’s status group.

We’ll add an additional byte to the variable naming convention:
- 1st byte is a Data Mart’s Table Indicator: C - for Customer Table;
- 2nd byte is a variable’s type : N – Numeric Total, D – Duration, B – Binary, etc.;
- 4th byte is an Account Status : A - Active, C – Cancel, _ - Any Status
- 6th byte is a Media Group : D – DM, T – TM, O – Online, M – Other;
- 8th byte is Duration Group : 1 – 0-3, 2 – 3-6, 3 – 6-9, 4 – 9-12, 5 – 12+ months;
Unused bytes are _.

Now the Data Mart’s Customer Marketing 3-dimensional table will look like:

<table>
<thead>
<tr>
<th>Status/Media /Duration</th>
<th>0-3 mnths</th>
<th>3-6 mnths</th>
<th>6-9 mnths</th>
<th>9-12 mnths</th>
<th>12+ mnths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>CN_A_D_1</td>
<td>CN_A_D_2</td>
<td>CN_A_D_3</td>
<td>CN_A_D_4</td>
<td>CN_A_D_5</td>
</tr>
<tr>
<td>TN</td>
<td>CN_A_T_1</td>
<td>CN_A_T_2</td>
<td>CN_A_T_3</td>
<td>CN_A_T_4</td>
<td>CN_A_T_5</td>
</tr>
<tr>
<td>ON</td>
<td>CN_A_O_1</td>
<td>CN_A_O_2</td>
<td>CN_A_O_3</td>
<td>CN_A_O_4</td>
<td>CN_A_O_5</td>
</tr>
<tr>
<td>OT</td>
<td>CN_A_M_1</td>
<td>CN_A_M_2</td>
<td>CN_A_M_3</td>
<td>CN_A_M_4</td>
<td>CN_A_M_5</td>
</tr>
<tr>
<td>Cancel</td>
<td>DA</td>
<td>DC</td>
<td>DD</td>
<td>DE</td>
<td>DF</td>
</tr>
<tr>
<td>TN</td>
<td>CN_C_T_1</td>
<td>CN_C_T_2</td>
<td>CN_C_T_3</td>
<td>CN_C_T_4</td>
<td>CN_C_T_5</td>
</tr>
<tr>
<td>ON</td>
<td>CN_C_O_1</td>
<td>CN_C_O_2</td>
<td>CN_C_O_3</td>
<td>CN_C_O_4</td>
<td>CN_C_O_5</td>
</tr>
<tr>
<td>OT</td>
<td>CN_C_M_1</td>
<td>CN_C_M_2</td>
<td>CN_C_M_3</td>
<td>CN_C_M_4</td>
<td>CN_C_M_5</td>
</tr>
<tr>
<td>Any Status</td>
<td>DN</td>
<td>DC</td>
<td>DD</td>
<td>DE</td>
<td>DF</td>
</tr>
<tr>
<td>TN</td>
<td>CN_T_1</td>
<td>CN_T_2</td>
<td>CN_T_3</td>
<td>CN_T_4</td>
<td>CN_T_5</td>
</tr>
<tr>
<td>ON</td>
<td>CN_O_1</td>
<td>CN_O_2</td>
<td>CN_O_3</td>
<td>CN_O_4</td>
<td>CN_O_5</td>
</tr>
<tr>
<td>OT</td>
<td>CN_M_1</td>
<td>CN_M_2</td>
<td>CN_M_3</td>
<td>CN_M_4</td>
<td>CN_M_5</td>
</tr>
</tbody>
</table>

The number of variables in the Data Mart’s table increased from 20 to 60.

Now we’ll add a format to define 1st-dimension index:

```
Value $statusf
    “A”=”1”
    “C”=”2”
    Other=”0”
```

In SAS the Data Mart Customer Table will be defined as 3-dimensional array with a Macro list of variables:

```
%macro c_totmkt;
CN_A_D_1 CN_A_D_2 CN_A_D_3 CN_A_D_4 CN_A_D_5
CN_A_T_1 CN_A_T_2 CN_A_T_3 CN_A_T_4 CN_A_T_5
CN_A_O_1 CN_A_O_2 CN_A_O_3 CN_A_O_4 CN_A_O_5
CN_A_M_1 CN_A_M_2 CN_A_M_3 CN_A_M_4 CN_A_M_5
CN_C_D_1 CN_C_D_2 CN_C_D_3 CN_C_D_4 CN_C_D_5
CN_C_T_1 CN_C_T_2 CN_C_T_3 CN_C_T_4 CN_C_T_5
CN_C_O_1 CN_C_O_2 CN_C_O_3 CN_C_O_4 CN_C_O_5
CN_C_M_1 CN_C_M_2 CN_C_M_3 CN_C_M_4 CN_C_M_5
CN___D_1 CN___D_2 CN___D_3 CN___D_4 CN___D_5
CN___T_1 CN___T_2 CN___T_3 CN___T_4 CN___T_5
```
Now we'll define table’s dimensions and Analysis Cut Date:

%let g_st_dim=3;
%let g_meddim=4;
%let g_durdim=5;
%let g_cut_dt="07aug2010"d;

And define the 3-dimensional table as the 3-dimensional array:
Array c_tmkt{& g_st_dim , & g_meddim,& g_durdim} %c_totmkt;

The whole program to create the Customer Data Mart’s 3-dimensional table is still nice and small:

Data custmkt.mkttbl(keep=t_custid %c_totmkt);;
  Length %c_totmkt 4;
  Retain %c_totmkt;
  Array c_tmkt{& g_st_dim , & g_meddim,& g_durdim} %c_totmkt;
  Set tranmkt.thans end=eof;
  By t_custid;
  If first.t_custid then do;
    Do i=1 to & g_st_dim;
      Do j=1 to & g_meddim;
        Do k=1 to & g_durdim;
          c_tmkt{i, j,k}=0;
        end;
      end;
    end;
  end;
  _i_st=put(upcase(t_status, $statusf.))+0;
  _i_med=put(upcase(t_media, $mediaf.))+0;
  _i_dur=put(&g_cut_dt – t_opendt, opndurf.)+0;
  If _i_st>0 and _i_dur >=0 then do;
    c_tmkt{_i_st, _i_med, _i_dur}+1;
    c_tmkt{& g_st_dim, _i_med, _i_dur}+1;
  end;
  If last.t_custid then do;
    Output custmkt.mkttbl;
  end;
Label   cn_a_d_1="# of active dir mail accnts opened 3 mns prior to cut dt"
cn_a_d_2="# of active dir mail accnts opened 3-6 mns prior to cut dt"
..........................
The Data Step program barely increased in size after we extended the Data Mart to the 3-dimensional table. The number of calculated variables tripled, but the program’s code remains compact and easy to read thanks to the fancy and efficient array technique.

CONCLUSION

The design and development of Historical Data Marts can be efficiently structured and implemented by organizing Data Mart variables in tables with table dimensions that correspond to analyzed variable groups or categories. The Data Mart variable tables are then handled as multidimensional arrays in Data Mart Aggregation programs. The multidimensional array programming technique described in the paper creates compact, efficient and easily expandable Data Mart SAS Aggregation programs.

REFERENCES


Mary McDonald, UBS Financial Services: Fun with Fancy Arrays, Paper 158-29

TRADEMARK NOTICES

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. In the USA and other countries ® indicates USA registration.

AUTHOR CONTACT

The authors welcome comments, questions, corrections, and suggestions.

Val Volovik
Affinion Group
6 High Ridge Park
Stamford, CT. 06905
(203) 956-8979
vvolovik@affiniongroup.com