

Cubes on the Cheap

By Jimmy DeFoor
Brierley and Partners

Cubes are hot items in the OnLine Analytical Processing (OLAP) environment. The functionality is usually quite costly to acquire, but equivalent content with reasonably good navigational capabilities can be created using linked HTMLs generated by SAS Data Steps and simple Put _ODS_ statements. This paper explains how to create such 'cubes' and provides examples of the code that would create them.

A cube is a multi-level view of the *dimensions* of a data set. A multi-level view is one in which statistical measures exist for each combination of the dimensions of the data. For example, if the dimensions of sales data for a grocery store chain were Year, Month, Store, and Household, a cube would have statistical measures such as sums, averages, and medians for sales at every combination of those dimensions.

The different levels of a cube are created by collapsing one or more dimensions and then calculating statistical measures on the other dimensions. For example, Year is collapsed and statistical measures are calculated on sales for each store, household and month for all years. Year and Month are then both collapsed and statistics are calculated on sales at each store and household for all years and all months. This process is repeated until all combinations have been evaluated.

The number of levels in a cube is calculated using the exponential function: 2^D , where D is the number of dimensions. For example, four dimensions create 16 levels; three dimensions, 8 levels; two, 4; one, 2.

Proc Summary generates these multi-level views in one pass of the input data set. By default, it generates sums for every level, but other measures can be requested, including maxima, minima, P10, P90, etc. Each level is given a value generated by the _Type_ field. This value is based upon the order of the dimensions in the Class statement and the rule that a zero (0) is assigned to a field when it is collapsed and a one (1) when it is not.

If the order of the dimensions were Year, Month, Store, and Household, Proc Summary would assign a value of 0000 to the level that had all dimensions collapsed and a value of 1111 to the dimension which had no fields collapsed. It would assign 0011 to the dimension that represented a collapse of Year and Month and a 1000 to the dimension that represented a collapse of Month, Store, and Household.

When a field is collapsed, Proc Summary sets that field to blank. Creating a user format and adding it to the Proc Summary will change the blanks to 'Collapsed' in the output data set.

```
Proc Format;
  Value $Blank
    ' ' = 'Collapsed'
;
Run;
*;
Proc Summary data = DataforCube;
  Class Year Month Store Household; /* Dimension fields */
  Var Sales Visits; /* Measure fields */
  Output out= DataforCubeStats(drop=_freq_) /* drop record counts */
    sum = Sales Visits
    mean = AvgSales AvgVisits
    median = P50Sales P50Visits
  ;
  Format Year Month Store Household $Blank.0; /* Change Blank to Collapsed */
Run;
```

Proc Summary doesn't write out the `_Type_` field in the binary format. Instead, it converts it to a decimal value. A value of 1111 becomes $8 + 4 + 2 + 1$ or 15. A value of 0011 becomes $0 + 0 + 2 + 1$ or 3. A value of 1011 becomes $8 + 0 + 2 + 1$ or 11. This calculation can be done in a Do loop:

```
TypeValue = 0;
Pos       = 0;
Dimension = 4;
Start     = Dimension - 1;
Do J = Start to 0 by -1;
  Pos = Pos + 1;
  PosValue = Input(Substr(_Type_,Pos,1),3.0);
  TypeValue = TypeValue + 2**(J*PosValue);
End;
_Type_ = Put(TypeValue,4.0);
```

Knowing this, a programmer can reconstruct the original value of `_Type_` and place it into a new field, `BinaryType`, which can be used to separate the output of Proc Summary into a data set for each level.

```
Array Binary (0:3) p0 p1 p2 p3;
Array Exponent (0:3) e0 e1 e2 e3
              (0 1 2 3); /* values of fields in array */
Remainder = _Type_;
/* Start with highest exponent and evaluate downward. */
Do J = 3 to 0 by -1;
  BinaryValue = 2**Exponent(J);
  If Remainder ge BinaryValue then
    do;
      Binary(J) = '1';
      Remainder = Remainder - BinaryValue;
    end;
  else
    Binary(J) = '0';
  end;
End;
/* Create Binary Type field by concatenating array values */
Binarytype = P3||P2||P1||P0;
```

After separate data sets are built for each `BinaryType`, an HTML is created for each using the ODS Html statement and `Put_ODS_`. The cube is built by linking the HTMLs to a Table of Contents and to each other.

The links are constructed using the anchor HTML tag, `<a`, and the hyperlink tag, `href`.

```
<a href=Type0001.html> "0001" </a>
```

The text inside the quotes is underlined in the HTML. When the text is clicked, the browser opens the file referenced by the href.

Here is an example of a Table of Contents with links to each report. Not all levels (_Types_) are shown. The underlined values contain the links.

Household Sales and Visits Data of Ajax Supermarkets

Table of Contents

Click Report to See That Level of the Cube

Report	Level	Year	Month	Store	Household
<u>0000</u>	0	Collapsed	Collapsed	Collapsed	Collapsed
<u>0001</u>	1	Collapsed	Collapsed	Collapsed	Expanded
<u>0010</u>	2	Collapsed	Collapsed	Expanded	Collapsed
<u>0011</u>	3	Collapsed	Collapsed	Expanded	Expanded
<u>0100</u>	4	Collapsed	Expanded	Collapsed	Collapsed
<u>1111</u>	15	Expanded	Expanded	Expanded	Expanded

Notice the entries under each dimension. They are either Collapsed or Expanded to indicate the summary level of that column. They are created by sub-stringing the Report field for the value of the dimension and writing 'Collapsed' if it is a zero (0) and 'Expanded' if it is a one (1). The Level field is created from _Type_ field that was output by Proc Summary. The Report field is created using the BinaryType field generated from the _Type_ field.

Each linked HTML will have the data for the level listed in that Report. For example, the 0100 Report would have monthly sales data over all years, all stores, and all households. In this example, it shows total, average, and P50 (median) sales for each month.

Household Sales and Visits Data of Ajax Supermarkets

Report 0100

Go to Table of Contents

Year	Month	Store	Household	TotalSales	AvgSales	P50Sales
Collapsed	January	Collapsed	Collapsed	10,000,000	2,000,000	1,800,000
Collapsed	February	Collapsed	Collapsed	11,000,000	2,200,000	1,900,000
Collapsed	March	Collapsed	Collapsed	12,000,000	2,400,000	2,200,000
Collapsed	April	Collapsed	Collapsed	11,000,000	2,200,000	2,000,000
Collapsed	May	Collapsed	Collapsed	12,000,000	2,400,000	2,100,000
Collapsed	June	Collapsed	Collapsed	13,000,000	2,600,000	2,300,000

A link back to the Table of Contents is placed in the header and footer of the report so that the user can easily move to any level of the cube. Though needed, that approach is insufficient. It does not give the user an intuitive method of expanding and collapsing dimensions. The 'Expand, Collapse' method is created by adding a table at the top and bottom of each report that takes the user to the 'opposite' view of each dimension. This is the table that would be used in the 0100 Report.

Year	Month	Store	Household
<u>Expand</u>	<u>Collapse</u>	<u>Expand</u>	<u>Expand</u>

The links for each column of the header and footer table are created by 'flipping' each digit one at a time, leaving the other digits unchanged. For example, the first digit of 0100 is for the Year dimension. Flipping the first digit to 1 creates a BinaryType of 1100. It shows expanded sales values for both Year and Month. Similarly, flipping only the second digit creates a BinaryType of 0000, which is the top level of all dimensions meaning all four dimensions have been collapsed.

Creating these links is done with the following code. It flips each digit and builds a ReportType of either expand or collapse. It then creates a value for each variable that contains the link and the ReportType. The resulting variables are written to a SAS data set that is used to create the header and footer for each report.

```

Array Fields (4) Field1 - Field4;
Start = 1;
Do J = 1 to 4; /* Evaluate each position of BinaryType */
  Linkref = Binarytype; /* copy BinaryType*/
  Binary = Substr(BinaryType,Start,1); /* examine one position of field */
  If Binary eq '0' then
    do;
      substr(LinkRef,Start,1) = '1'; /* Flip to a 1 if originally a 0 */
      ReportType = 'Expand';
    end;
  else
    do;
      substr(LinkRef,Start,1) = '0'; /* Else Flip to a 0 */
      ReportType = 'Collapse';
    end;
  Field(J) = '<a href=Type'||LinkRef||'.html> "'||ReportType||'" </a>';
  Start = Start + 1;
End;

```

Opening the data set generated in this step will show this content for each column.

```
'<a href=Type0100.html>"Expand" </a>';
```

As stated earlier, the browser opens the file named in the 'href=' after the double-quoted item is clicked, which is Expand in this example.

The data step that writes the header and footer values is very simple. It uses labels to create column names that match the columns of the report. It then tells ODS which variables will be in the table and writes the variables to the table using the default Put _ODS_ statement. That statement writes all variables at once, in the order defined in the File Print ODS statement.

```

Data _null_;
  Set HeaderFooter end=eof;
  Label Field1 = Year Field2 = Month Field3 = Store Field4 = Household;
  File Print ODS=(Variables=(Field1 - Field4));
  Put _ODS_;
run;

```

Each report HTML is constructed with three data steps. The first step writes the header table; the second, the report table; the third, the footer table. Writing to one HTML file three times requires the use of the 'mod' parameter on the file statement so that the tables are appended instead of overwritten. It also necessitates the use of the No_Top_Matter and No_Bottom_Matter options on the Body parameter of the ODS HTML statement to prevent HTML commands from being written to the file that would cause the browser to display separation lines between the tables.

```

Filename body "D:\Proj\ZDevelopment\Reports\Type0100.html" mod;
ODS HTML body=body (No_Top_Matter No_Bottom_Matter);

```

The ODS HTML statement specifies that HTML commands will be written to the filename referenced in the Body= parameter. The ODS HTML statement is also used to open and close the file, set the style template that will be used for the fonts in the file, and indicate whether frame or page files will be associated with the body file being written.

The three data steps that create the report HTMLs are placed into a macro that is executed once for every data set created for a unique BinaryType. In the example used in this paper, there are 16 such data sets.

The macro and the data to create the 'cube' are available from the author via email: thefoor@hotmail.com. That macro is capable of creating a cube with 1 to 10 dimensions.

Below are examples of the Table of Contents and one report created by the author's macro.

Household Sales and Visits Data of Ajax SuperMarkets

Table of Contents

Click Report to See That Level of the Cube

Report	Level	Year	Month	Store	Dimension
0000000000	0	Collapsed	Collapsed	Collapsed	Collapsed
0000000001	1	Collapsed	Collapsed	Collapsed	Expanded
0000000010	2	Collapsed	Collapsed	Expanded	Collapsed
0000000011	3	Collapsed	Collapsed	Expanded	Expanded
0000000100	4	Collapsed	Expanded	Collapsed	Collapsed
0000000101	5	Collapsed	Expanded	Collapsed	Expanded
0000000110	6	Collapsed	Expanded	Expanded	Collapsed
0000000111	7	Collapsed	Expanded	Expanded	Expanded
0000001000	8	Expanded	Collapsed	Collapsed	Collapsed
0000001001	9	Expanded	Collapsed	Collapsed	Expanded
0000001010	10	Expanded	Collapsed	Expanded	Collapsed
0000001011	11	Expanded	Collapsed	Expanded	Expanded
0000001100	12	Expanded	Expanded	Collapsed	Collapsed
0000001101	13	Expanded	Expanded	Collapsed	Expanded
0000001110	14	Expanded	Expanded	Expanded	Collapsed
0000001111	15	Expanded	Expanded	Expanded	Expanded

Household Sales and Visits Data of Ajax SuperMarkets
Report 0000001010

"Go to Table of Contents"

Year	Month	Store	Household
"Collapse"	"Expand"	"Collapse"	"Expand"

Level	Year	Month	Store	Household	Sales	Units	Average	Average	POS/Store	POS/Visit
10	1999	Collapsed	7926	Collapsed	\$1,706	1,140	97.19	2.14	50	1
10	1999	Collapsed	7931	Collapsed	29,402	605	113.08	2.33	45	1
10	1999	Collapsed	7964	Collapsed	95,899	1,484	168.64	2.61	134	2
10	2000	Collapsed	7926	Collapsed	50,840	1,114	93.97	2.06	50	1
10	2000	Collapsed	7931	Collapsed	39,625	826	103.15	2.35	48	1
10	2000	Collapsed	7964	Collapsed	100,187	1,489	160.83	2.39	97	2

Year	Month	Store	Household
"Collapse"	"Expand"	"Collapse"	"Expand"

"Go to Table of Contents"

More information on ODS and on creating HTMLs with the SAS Data Step can be found in the Online Documentation and in these three papers available on the SAS website.

Heffner, W. F. (1998), "ODS: The Data Step Knows," in the *Proceedings for the Twenty-Third Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.

Olinger, C. R. (1999), "Twisty Little Passages, All Alike – ODS Templates Exposed" in the *Proceedings for the Twenty-Forth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.

Olinger, C. R. (2000), "ODS for Dummies" in the *Proceedings for the Twenty-Fifth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc.

A nice primer on HTML commands, now available in many half-price bookstores, is

Evans, Tim, Sams Teach Yourself HTML 4 in 10 minutes, Sams Corporation, Indianapolis, Ind., 1998, 230 pages. [Http://www.sampublishing.com/](http://www.sampublishing.com/).

TRADEMARKS

SAS® and all SAS products are trademarks are registered trademarks of SAS Institute Inc.