ABSTRACT

Most of the time, OLAP cubes are built from data that has additive measures, meaning that as you drill down, the sum of all the lower levels will add up to the value at the highest level of the hierarchy. This is not always the case. Sometimes applications need drilldown capabilities on data where the measures are non-additive. And, sometimes data is additive in one dimension, but not another. Take for example, a car leasing company that has 2,000 cars to lease. They want to build a cube with two dimensions: **Time** and **Geography**. Across the Geography dimension, the number of cars is additive. Let’s say that the levels in the Geography dimension are: Company, Region, State and City. At the Company level, the number of cars is 2,000. When we drill down to the Region level, the total at all the regions adds up to 2,000. When we drill down to the next level (State), the total number of cars in all the states adds up to 2,000. etc. This same measure (total number of cars) is NOT additive in the Time dimension. Let’s say that the levels of the Time dimension are: Year, Quarter and Month. If we take the number of cars that are leased each Month, they could add up to more than 2,000. And likewise, if we add up all the cars leased each Quarter, they could add up to more than 2,000. But, still this company wants to build a cube with this data. This paper looks at strategies and methods to building a cube with non-additive data. Then, a step by step approach is taken to actually build the cube.

INTRODUCTION

Scenario: a Car leasing company has 2000 cars to lease which are distributed among 36 cities (in 9 states within 3 regions). They want to build a cube that will tell them how many cars are leased and how many are available as they drill down through **time** and **geography**. There are four measures that they want to follow: Available_Cars, Leased_Cars, Amount_Billed, and Amount_Collected. Available_Cars and Leased_Cars are additive in the Geographic hierarchy, but not in the Time hierarchy. For example, you can add up all the available cars in all the states (geographic hierarchy) and that will give you 2,000. But, no matter what year you are examining (time hierarchy), there are only 2,000 cars available in any given year AND only 2,000 cars available for all the years. (You can’t add up that number across the years. Amount_Billed and Amount_Collected are additive across both the Time and the Geography dimension.

THE DATA

The data is stored in a SAS dataset named SASUSER.CUBE_DATA.
Figure 1. Source Data

There is one row per city / month. There are 36 cities with 24 months worth of data for each city for a total 864 rows.

DATA PREPARATION

The above data needs to be manipulated so that we can create two dimensions for the cube: Time and Geography. For the Time dimension, the levels are YEAR and MONTH and for the Geography dimension, the levels are REGION, STATE and CITY. The right four columns (AMOUNT_BILLED, AMOUNT_COLLECTED, AVAILABLE_CARS, and LEASED_CARS) are all measures. Since AVAILABLE_CARS and LEASED_CARS are NON-ADDITIVE across the TIME dimension, we need to summarize the data and create a SAS dataset for each level of the dimensions.

To do the necessary summarization, both PROC MEANS and the DATA step are used.

```sas
proc means data=in_cube.cube_data noprint sum chartype;
  var Amount_Billed Amount_Collected Available_Cars Leased_Cars;
  class Year Month Region State City;
  output out=in_cube.YM_11000(where=(_type_='11000') drop=_F:)
    sum(Amount_Billed)= sum(Amount_Collected)=
    sum(Available_Cars)= sum(Leased_Cars)=;
  output out=in_cube.YMR_11100(where=(_type_='11100') drop=_F:)
    sum(Amount_Billed)= sum(Amount_Collected)=
    sum(Available_Cars)= sum(Leased_Cars)=;
  output out=in_cube.YMRS_11110(where=(_type_='11110') drop=_F:)
    sum(Amount_Billed)= sum(Amount_Collected)=
    sum(Available_Cars)= sum(Leased_Cars)=;
  output out=in_cube.YMRSCL_11111(where=(_type_='11111') drop=_F:)
    sum(Amount_Billed)= sum(Amount_Collected)=
    max(Available_Cars)= max(Leased_Cars)=;
run;
```

Program 1. PROC MEANS

This one PROC MEANS step creates four different SAS datasets all at different levels of summarization. Notice the naming convention of the datasets. The first dataset to be created is called YM_11000 and is summarized at the YEAR and MONTH level. The key to the numeric pattern is the order of the variables on the CLASS statement.
LEVEL 1 Data

The first 13 rows of YM_11000 are shown below.

<table>
<thead>
<tr>
<th>year</th>
<th>month</th>
<th>Region</th>
<th>State</th>
<th>city</th>
<th>TYPE_</th>
<th>Amount_Billed</th>
<th>Amount_Collected</th>
<th>Available_Cars</th>
<th>Leased_Cars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>132000</td>
<td>110209</td>
<td>1604</td>
<td>1320</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>135200</td>
<td>105404</td>
<td>1640</td>
<td>1352</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>141300</td>
<td>110503</td>
<td>1676</td>
<td>1413</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>145400</td>
<td>115274</td>
<td>1712</td>
<td>1454</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>151000</td>
<td>119554</td>
<td>1748</td>
<td>1510</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>153800</td>
<td>119923</td>
<td>1784</td>
<td>1526</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>156100</td>
<td>125085</td>
<td>1820</td>
<td>1561</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>159300</td>
<td>124121</td>
<td>1856</td>
<td>1593</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>159100</td>
<td>130160</td>
<td>1892</td>
<td>1619</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>170300</td>
<td>142677</td>
<td>1928</td>
<td>1703</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>173100</td>
<td>136621</td>
<td>1964</td>
<td>1731</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>176200</td>
<td>135863</td>
<td>2000</td>
<td>1762</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>11000</td>
<td></td>
<td></td>
<td></td>
<td>151000</td>
<td>121773</td>
<td>1604</td>
<td>1510</td>
</tr>
</tbody>
</table>

Figure 2: Level 1 Data

Notice the values for the twelfth observation. This row represents the total Amount_Billed and total Amount_Collected for ALL of 2008. The other columns (Available_Cars and Leased_Cars) which are NON-Additive, reveal the values at the END of the 2008. In other words, in 2008, we had 2000 cars to lease, and we leased 1,762 of them. So, this is the row we want to show when we are looking at data for the year of 2008.

In order to get these values when we drilldown, we need to run the following code:

```plaintext
/* Level 1 : 10000 */
data in_cube.Y_10000 in_cube.All_000000;
  drop AB AC AA LC YAB YAC YAA YLC;
  set in_cube.ym_11000(rename=(Amount_Billed=AB Amount_Collected=AC
                               Available_Cars=AA Leased_Cars=LC)) end=e;
  by year;
  if first.year then do;
    Amount_Billed =0; Amount_Collected=0;
    Available_Cars=0; Leased_Cars=0;
  end;
  Amount_Billed + AB; Amount_Collected+AC;
  Available_Cars + AA; Leased_Cars + LC;
  YAB + AB; YAC + AC; YAA + AA; YLC + LC;
  if last.year then do;
    TYPE_ = '10000';
    Month=.;
    Leased_Cars=LC;
    Available_Cars=AA;
    output in_cube.Y_10000;
  end;
  if e;
  Year=.; Month=.; _TYPE_='00000';
  Amount_Billed = YAB; Amount_Collected = YAC;
  Available_Cars = AA; Leased_Cars = LC;
  output in_cube.All_000000;
run;
```

Program 2.

3
Examine the two datasets created above.

**Figure 3: ALL_00000 Dataset**

The ALL_00000 data set contains maximum values for all four measures. In other words, the total Amount_Billed for ALL transactions was $3,841,500. The total Amount_Collected for ALL transactions was $3,073,797. The total number of Available_Cars (for both years) was 2000, and the total number of Leased_Cars (for both years) was 1,820.

**Figure 4: Y_10000 Dataset**

The Y_10000 data set is summed for each value of YEAR. Here it may be a little more obvious that Available_Cars and Leased_Cars are NON-Additive. There are only 2,000 cars to lease for BOTH years, not for EACH year.

For classification purposes, the above datasets are referred to as LEVEL 1 datasets.

**LEVEL 2 Data**

The following code creates the LEVEL 2 datasets. These are summarized for each YEAR and REGION.
Program 3.

The first DATA step gets the last row for each REGION for each YEAR. The PROC MEANS step gets the SUM for each REGION for each YEAR. The last DATA step merges the two datasets together so that each row has the SUMS for AMOUNT_BILLED and AMOUNT_COLLECTED and the LAST ROW for AVAILABLE_CARS and LEASED_CARS. The resulting dataset is shown below.

```
/* Level 2: 10100 */

proc sort data=in_cube.ymr_11100;
    by year region month;
run;

data test2m;
    set ymr_11100;
    by year region;
    if last.region;
run;

proc means data=ymr_11100 sum nway nolimit;
    class year region;
    var Available_Cars Leased_Cars Amount_Billed Amount_Collected;
    output out=test2y sum=;
run;

data in_cube.Y_R_10100(drop=_FREQ_);
    merge test2y(drop=Available_Cars Leased_Cars _type_)
        test2m(drop=Amount_Billed Amount_Collected _type_);
    by year region;
    month='.';
    _Type_='10100';
run;
```

The first DATA step gets the last row for each REGION for each YEAR. The PROC MEANS step gets the SUM for each REGION for each YEAR. The last DATA step merges the two datasets together so that each row has the SUMS for AMOUNT_BILLED and AMOUNT_COLLECTED and the LAST ROW for AVAILABLE_CARS and LEASED_CARS. The resulting dataset is shown below.

![Table](image)

Figure 5: Y_R_10100 Dataset.

The YMR_11100 created by the first PROC MEANS dataset is also a LEVEL 2 dataset.

LEVEL 3 Data

The following code creates the LEVEL 3 data. The data are summarized for each YEAR, REGION and STATE.
Program 4.

The pattern for this program is the same as for program 3. The first DATA step gets the last row for each STATE / REGION / YEAR. The PROC MEANS step gets the SUM for each STATE / REGION / YEAR. The last DATA step merges the two datasets together so that each row has the SUMS for AMOUNT_BILLED and AMOUNT_COLLECTED and the LAST ROW for AVAILABLE_CARS and LEASED_CARS.

The resulting dataset is shown in the PROC PRINT output below.

<table>
<thead>
<tr>
<th>year</th>
<th>month</th>
<th>Region</th>
<th>State</th>
<th>city</th>
<th>Amount_Billed</th>
<th>Amount_Collected</th>
<th>Available_Cars</th>
<th>Leased_Cars</th>
<th><em>Type</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td></td>
<td>Central</td>
<td>KS</td>
<td></td>
<td>255,200</td>
<td>203,069</td>
<td>250</td>
<td>233</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>Central</td>
<td>MI</td>
<td></td>
<td>103,100</td>
<td>83,386</td>
<td>114</td>
<td>107</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>Central</td>
<td>TX</td>
<td></td>
<td>122,290</td>
<td>135,223</td>
<td>209</td>
<td>157</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>East</td>
<td>FL</td>
<td></td>
<td>276,700</td>
<td>219,327</td>
<td>274</td>
<td>244</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>East</td>
<td>NC</td>
<td></td>
<td>235,400</td>
<td>190,720</td>
<td>260</td>
<td>231</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>East</td>
<td>NY</td>
<td></td>
<td>223,400</td>
<td>177,957</td>
<td>267</td>
<td>240</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>West</td>
<td>CA</td>
<td></td>
<td>225,600</td>
<td>183,992</td>
<td>221</td>
<td>203</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>West</td>
<td>OR</td>
<td></td>
<td>214,300</td>
<td>171,909</td>
<td>214</td>
<td>158</td>
<td>10110</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td>West</td>
<td>WA</td>
<td></td>
<td>149,300</td>
<td>121,194</td>
<td>155</td>
<td>149</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>Central</td>
<td>KS</td>
<td></td>
<td>260,200</td>
<td>209,702</td>
<td>250</td>
<td>228</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>Central</td>
<td>MI</td>
<td></td>
<td>102,400</td>
<td>79,140</td>
<td>114</td>
<td>107</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>Central</td>
<td>TX</td>
<td></td>
<td>209,800</td>
<td>169,987</td>
<td>209</td>
<td>196</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>East</td>
<td>FL</td>
<td></td>
<td>263,100</td>
<td>225,362</td>
<td>274</td>
<td>254</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>East</td>
<td>NC</td>
<td></td>
<td>262,400</td>
<td>217,724</td>
<td>288</td>
<td>225</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>East</td>
<td>NY</td>
<td></td>
<td>272,200</td>
<td>217,397</td>
<td>267</td>
<td>259</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>West</td>
<td>CA</td>
<td></td>
<td>228,200</td>
<td>187,436</td>
<td>221</td>
<td>209</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>West</td>
<td>OR</td>
<td></td>
<td>216,200</td>
<td>166,639</td>
<td>214</td>
<td>197</td>
<td>10110</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>West</td>
<td>WA</td>
<td></td>
<td>152,600</td>
<td>117,846</td>
<td>155</td>
<td>151</td>
<td>10110</td>
</tr>
</tbody>
</table>

---

3,041,500  3,079,797  4,000  3,582
By using the SUM statement in PROC PRINT, the 2 additive columns ‘add up’ to match the totals for the entire dataset. But, when we add the NON-Additive columns (Available_Cars and Leased_Cars) do not match the totals for the entire dataset.

LEVEL 4 Data

The following code creates the LEVEL 4 data. The data are summarized for each YEAR, REGION, STATE and MONTH.

```sas
/* Level 4: 10110 */

proc sort data=in_cube.ymrsc_11111;
  by year region state city month;
run;

data test4m;
  set in_cube.ymrsc_11111;
  by year region state city;
  if 1st.city;
run;

proc means data=in_cube.ymrsc_11111 sum nway noprint;
  class year region state city;
  var Available_Cars Leased_Cars Amount_Billed Amount_Collected;
  output out=test4y sum=;
run;

data in_cube.Y_RSC_10111(drop=_Fre: );
  merge test4y(drop=Available_Cars Leased_Cars _type_)
    test4m(drop=Amount_Billed Amount_Collected _type_);
  by year region state;
  month=.;
  _Type_='10111';
run;
```

Program 5: Generating LEVEL 4 Data.

We now have the following datasets:

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Level</th>
<th>Rows</th>
</tr>
</thead>
</table>
• **ALL_00000** - Level 0 - 1 row.
• **Y_10000** - Level 1Y - 2 rows – 1 per Year.
• **YM_11000** - Level 1M - 24 rows – 1 per Year per Month.
• **Y_R_10100** - Level 2Y - 6 rows – 1 per Year per Region.
• **YMR_11100** - Level 2M - 72 rows – 1 per Year / Month / Region.
• **Y_RS_10110** - Level 3Y - 18 rows – 1 per Year / Region / State.
• **YMRS_11110** - Level 3M - 216 rows – 1 per Year / Month / Region / State.
• **Y_RSC_10111** - Level 4Y - 72 rows – 1 per Year / Region / State / City.
• **YMRSC_11111** - Level 4M - 864 rows – 1 per Year / Month / Region/State/City.

The next step is to ‘Register’ the data so that we can build a cube. The registration of the datasets is done in SAS Management Console and will be illustrated in the presentation.

**BUILDING THE CUBE**

The first PROC OLAP step deletes the CUBE if it exists. The beginning of the second step is shown here.

```sas
libname in_cube 'c:\Olap_cube\Cars\Data';

proc olap delete_physical cube=Car_Lease;
   METASVR host="localhost" port=8561
   protocol=bridge
   userid="sasdemo" pw="sasbtc"
   repository="Foundation"
   olap_schema="SASMain - OLAP Schema";
run;

proc olap cube=Car_Lease
   path="c:\Olap_Cube\Cars"
   description="Car Lease Cube";
   METASVR host="localhost" port=8561 protocol=bridge
   userid="sasdemo" pw="sasbtc"
   repository="Foundation"
   olap_schema="SASMain - OLAP Schema";
```

Program 6. PROC OLAP step
The first **DIMENSION** statement is shown. The TIME DIMENSION contains the TIME HIERARCHY which has the LEVELS Year and Month.

```
DIMENSION Time hierarchies=(Time)
      CAPTION = 'Time Dimension'
      SORT_ORDER = ASCENDING ;

HIERARCHY Time
    levels=(Year Month)
    CAPTION='Time Hierarchy' ;
      LEVEL Year
        CAPTION='Year'
        SORT_ORDER=ASCENDING
        ;
      LEVEL Month
        CAPTION='Month'
        SORT_ORDER=ASCENDING
        ;
```

Program 7. The DIMENSION Statement and the HIERARCHY Statement for TIME.

The next DIMENSION statement is for the GEOGRAPHY dimension.

```
DIMENSION Geography hierarchies=(Geography)
      CAPTION='Geography Dimension'
      TYPE=GEOGRAPHY SORT_ORDER=ASCENDING ;

HIERARCHY Geography /* ALL_MEMBER='All Geography' */
    levels=(Region State City)
    CAPTION='Geography Hierarchy' DEFAULT ;
      LEVEL Region
        CAPTION='Region'
        SORT_ORDER=ASCENDING
        ;
      LEVEL State
        CAPTION='State '
        SORT_ORDER=ASCENDING
        ;
      LEVEL City
        CAPTION='City'
        SORT_ORDER=ASCENDING
        ;
```
The four MEASURE statements come next.

```
**Program 9. The MEASURE Statements.**

<table>
<thead>
<tr>
<th>MEASURE</th>
<th>Amount_Billed_Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT=SUM</td>
<td></td>
</tr>
<tr>
<td>Aggr_COLUMN=Amount_Billed</td>
<td></td>
</tr>
<tr>
<td>CAPTION='Sum of Amount Billed'</td>
<td></td>
</tr>
<tr>
<td>FORMAT=Dollar12.2</td>
<td>/<em>DEFAULT</em>/</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MEASURE</td>
<td>Amount_Collected_Sum</td>
</tr>
<tr>
<td>STAT=SUM</td>
<td></td>
</tr>
<tr>
<td>Aggr_COLUMN=Amount_Collected</td>
<td></td>
</tr>
<tr>
<td>CAPTION='Sum of Amount Collected'</td>
<td></td>
</tr>
<tr>
<td>FORMAT=Dollar12.2</td>
<td>/<em>DEFAULT</em>/</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MEASURE</td>
<td>Available_Cars</td>
</tr>
<tr>
<td>STAT=SUM</td>
<td></td>
</tr>
<tr>
<td>Aggr_COLUMN=Available_Cars</td>
<td></td>
</tr>
<tr>
<td>CAPTION='Available Cars'</td>
<td></td>
</tr>
<tr>
<td>FORMAT=comma8.</td>
<td>/<em>DEFAULT</em>/</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MEASURE</td>
<td>Leased_Cars</td>
</tr>
<tr>
<td>STAT=SUM</td>
<td></td>
</tr>
<tr>
<td>Aggr_COLUMN=Leased_Cars</td>
<td></td>
</tr>
<tr>
<td>CAPTION='Leased Cars'</td>
<td></td>
</tr>
<tr>
<td>FORMAT=comma8.</td>
<td>/<em>DEFAULT</em>/</td>
</tr>
</tbody>
</table>
```

Each MEASURE statement names the numeric variable that is the measure. It names the statistic as well as the CAPTION which is basically a label.

Next in the program are the AGGREGATION statements. They specify the aggregation and name all the variables that make up the granularity of that aggregation. Following the ‘/’ the table= option is specified that names the dataset that will be used for a specific aggregation level. This is where the magic really occurs. In ‘traditional’ drilldown hierarchies, as you drill from one level to the next, you are going from one VARIABLE to another in the SAME dataset. With NON-Additive measures, as you drilldown, you go from one DATASET to another. The dataset for a specific aggregation level is named on the AGGREGATION statement.
The only thing after the AGGREGATION statements is the RUN statement. When this above PROC OLAP step is submitted, the CAR_LEASE ‘cube’ is built.

During the presentation of this paper, the cube will be surfaced in Enterprise Guide as well as Web Report Studio.
CONCLUSION
By knowing how to write PROC OLAP code, you can create ‘cubes’ that might not be creatable through OLAP Cube Studio.

RECOMMENDED READING
There are a number of books and training courses available from SAS Institute, Inc. Some courses of interest might be: Creating and Viewing OLAP Cubes; Overview of SAS Business Intelligence and Data Integration Applications; What’s New in SAS 9.2 Business Intelligence; Designing, Tuning and Maintaining OLAP Cubes; and SAS OLAP Environment Administration.

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

Name: Ben Cochran
Company: The Bedford Group
Address: 3224 Bedford Ave.
City, State ZIP: Raleigh, NC
Work Phone: (919) 741-0370
E-mail: bedfordgroup@nc.rr.com

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.
Other brand and product names are trademarks of their respective companies.