Extracting Planning Analytics/TM1 Dimension Hierarchy using TMVGate

Technical Note: TA2019005

Contents

1.0 Overview

Many users appreciate the flexibility of TM1's dimension structure.

TM1 can handle a variety of hierarchy types such as multiple hierarchies, unbalanced trees and child elements with multiple parents. All these capabilities seem natural in TM1. However, when it comes to other BI or visualization tools that interface or require a direct linkage to TM1, these become a nightmare.

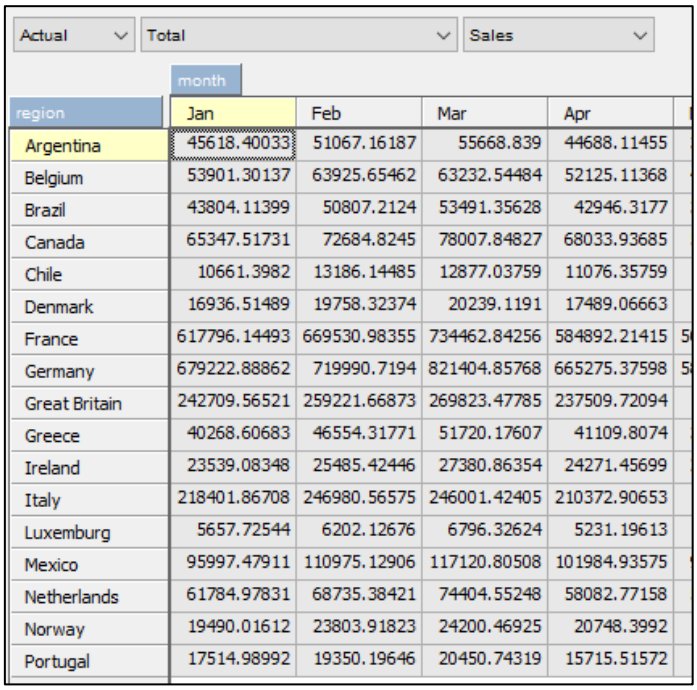
Most BI or visualization tools use a tabular mode which requires the hierarchy structure to be a balanced tree with a single hierarchy structure. Generally, these tools will not be able to consume the hierarchy structure in TM1 directly.

This technical document aims to provide a general guide on how to produce a dimension hierarchy extract using TMVGate that can be imported into BI Tools. For illustration purposes, Microsoft Power BI will be used.

2.0 Extracting Cube Data

Using TMVGate, cube data is extracted based on a specified cube view, which can be either private or public. In general, most of the dimension elements specified in the cube view will be at the same level in order for them to make sense in the BI reports. The elements will either be defined in a Subset or explicitly selected in the cube view definition.

Let's take the TM1 default SData server, specifically, the cube "SalesCube" as an exmaple. The following screen shows a view with the "Region" dimension at the row dimension with N level elements (i.e. Level 0). Likewise, the view shows the "Month" dimension at the column dimension with N level elements.



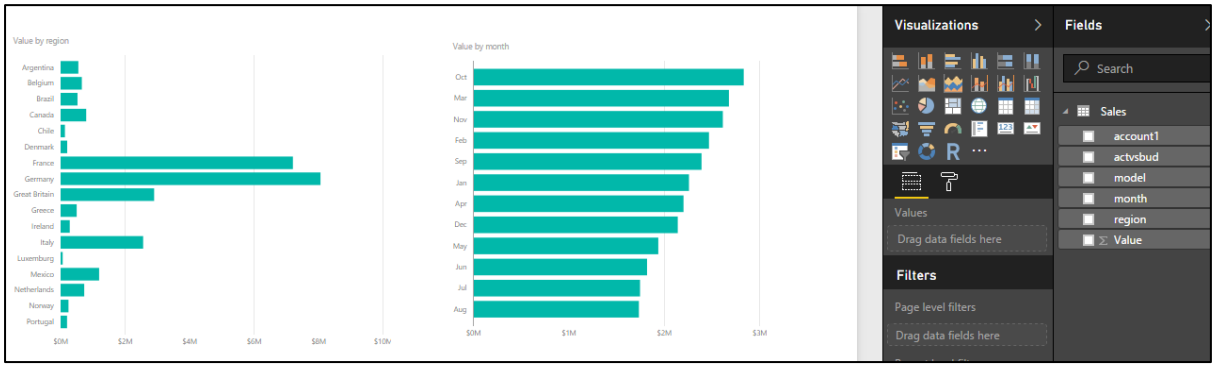
Actual | Total | Sales

month

region	Jan	Feb	Mar	Apr
Argentina	45618.40033	51067.16187	55668.839	44688.11455
Belgium	53901.30137	63925.65462	63232.54484	52125.11368
Brazil	43804.11399	50807.2124	53491.35628	42946.3177
Canada	65347.51731	72684.8245	78007.84827	68033.93685
Chile	10661.3982	13186.14485	12877.03759	11076.35759
Denmark	16936.51489	19758.32374	20239.1191	17489.06663
France	617796.14493	669530.98355	734462.84256	584892.21415
Germany	679222.88862	719990.7194	821404.85768	665275.37598
Great Britain	242709.56521	259221.66873	269823.47785	237509.72094
Greece	40268.60683	46554.31771	51720.17607	41109.8074
Ireland	23539.08348	25485.42446	27380.86354	24271.45699
Italy	218401.86708	246980.56575	246001.42405	210372.90653
Luxemburg	5657.72544	6202.12676	6796.32624	5231.19613
Mexico	95997.47911	110975.12906	117120.80508	101984.93575
Netherlands	61784.97831	68735.38421	74404.55248	58082.77158
Norway	19490.01612	23803.91823	24200.46925	20748.3992
Portugal	17514.98992	19350.19646	20450.74319	15715.51572

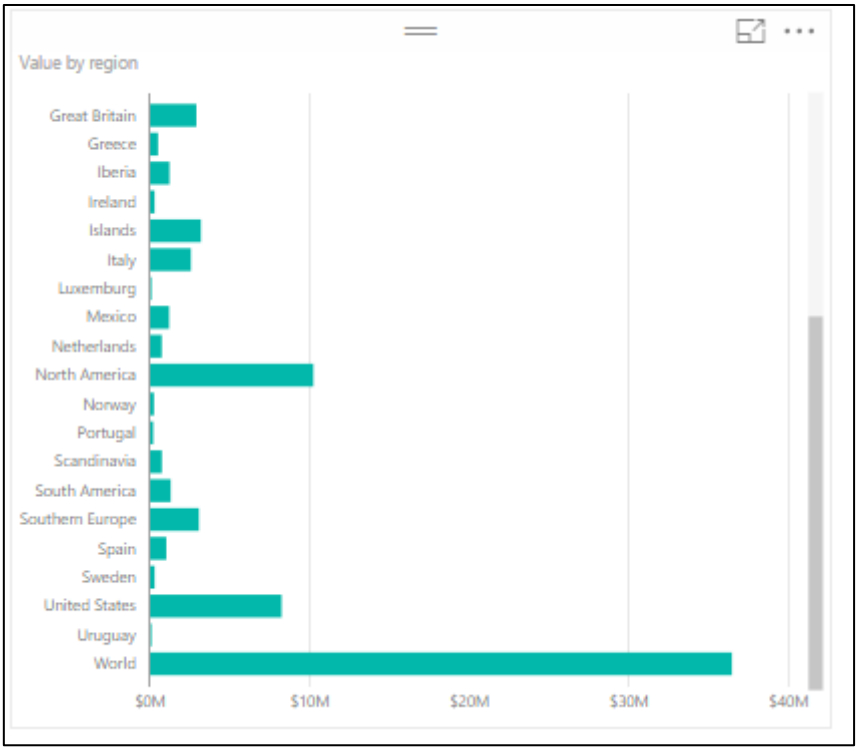
In Power BI, this cube data is translated into simple bar charts as shown below:

By Country and By Month



Power BI does not have recognize the concepts of levels. If you start pulling different-levelled elements into Power BI, the visualizations may become misleading.

In this example, we are pulling all the sub-regions, regions, World into the Power BI.



Of course, there will be cases where the user would like to pull different level elements into Power BI based on their requirements. There is nothing wrong in doing so except that user must understand the implications of mixing different level of elements in their report.

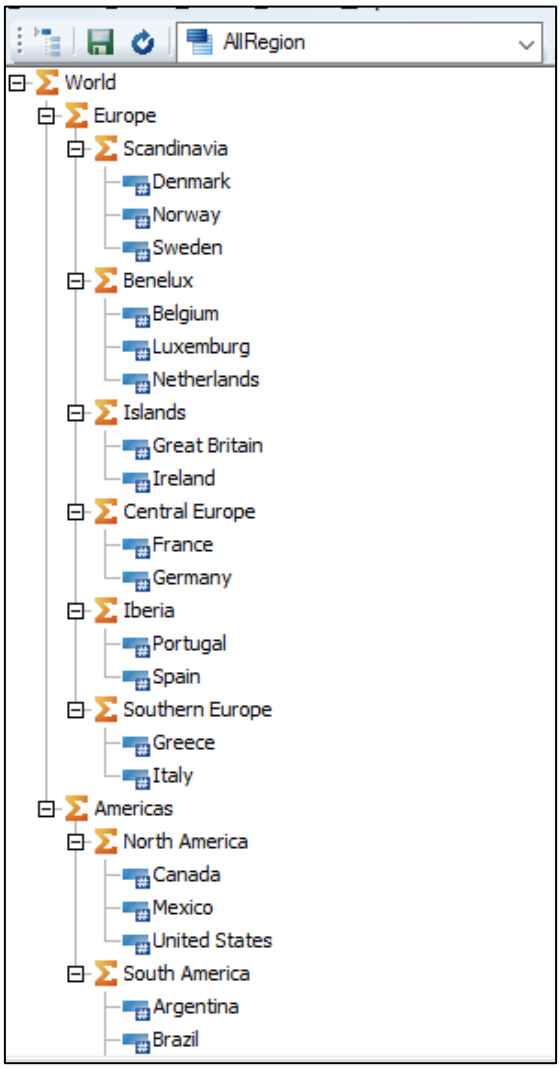
The next topic we will discuss how we can enrich our data analysis with hierarchy information.

3.0 Method 1 – Flattening: Enriching the Cube data with hierarchy structure

Using the same cube view data which has the countries (N level) from the "Region" dimension, we want to enrich the Power BI reports to show the sub-regions and region hierarchy.

The first thing to do is to create a subset in the "Region" dimension which contains the elements in the cube view, and the required parent elements that you would like to be made available in Power BI.

In this example, we created a subset "AllRegion" that contains all the N level elements specified in the cube view, as well as the respective parents. These elements eventually roll up to the "World". This is important as the subset **MUST** contain all the required elements and parents in order for TMVGate to construct the tabular table for Power BI to consume the data.



Using the "Create Get Hierarchy URL" function with the "Flatten" output type, the TMVgate URL statement will look something like this:

```
http://.....&pDimension=Region&pSubset=AllRegion&pPrivate=false&pFormat=JSON&pAlias&pFillGap=LR
```

Notice that we are using a fill gap option of LR (Left to right). Please review the user guide on the various fill options.

When we extract this hierarchy information into Power BI, the data table will look something like this:

Region_L3	Region_L2	Region_L1	Region_L0	Region
World	Europe	Scandinavia	Denmark	Denmark
World	Europe	Scandinavia	Norway	Norway
World	Europe	Scandinavia	Sweden	Sweden
World	Europe	Benelux	Belgium	Belgium
World	Europe	Benelux	Luxemburg	Luxemburg
World	Europe	Benelux	Netherlands	Netherlands
World	Europe	Islands	Great Britain	Great Britain
World	Europe	Islands	Ireland	Ireland
World	Europe	Central Europe	France	France
World	Europe	Central Europe	Germany	Germany
World	Europe	Iberia	Portugal	Portugal
World	Europe	Iberia	Spain	Spain
World	Europe	Southern Europe	Greece	Greece
World	Europe	Southern Europe	Italy	Italy
World	Americas	North America	Canada	Canada
World	Americas	North America	Mexico	Mexico
World	Americas	North America	United States	United States
World	Americas	South America	Argentina	Argentina
World	Americas	South America	Brazil	Brazil
World	Americas	South America	Chile	Chile
World	Americas	South America	Uruguay	Uruguay

Using TMVGate, we flattened the TM1 hierarchy structure into a tabular format which can be consumed by Power BI (or other tools).

With the above example, “Region” column is always the element principal name. “Region_L0” will be the matching lowest level element specified in the subset in the corresponding Alias if alias option is specified in the TMVGate extract URL.

The following example illustrates a hierarchy extract with alias specified. You can notice that “Region_L0” and “Region” are showing different values.

Region_L3	Region_L2	Region_L1	Region_L0	Region
全球	欧洲	斯堪的纳维亚半岛	丹麦	Denmark
全球	欧洲	斯堪的纳维亚半岛	挪威	Norway
全球	欧洲	斯堪的纳维亚半岛	瑞典	Sweden
全球	欧洲	（比利时、荷兰、卢森堡三国经济联盟	比利时	Belgium
全球	欧洲	（比利时、荷兰、卢森堡三国经济联盟	卢森堡	Luxemburg
全球	欧洲	（比利时、荷兰、卢森堡三国经济联盟	荷兰	Netherlands
全球	欧洲	群岛	英国	Great Britain
全球	欧洲	群岛	爱尔兰	Ireland
全球	欧洲	中欧	法国	France
全球	欧洲	中欧	德国	Germany
全球	欧洲	伊比利亚半岛	葡萄牙	Portugal
全球	欧洲	伊比利亚半岛	西班牙	Spain
全球	欧洲	南欧	希腊	Greece
全球	欧洲	南欧	意大利	Italy
全球	美洲	北美	加拿大	Canada
全球	美洲	北美	墨西哥	Mexico
全球	美洲	北美	美国	United States
全球	美洲	南美	阿根廷	Argentina
全球	美洲	南美	巴西	Brazil
全球	美洲	南美	智利	Chile
全球	美洲	南美	乌拉圭	Uruguay

We are now ready to link up the hierarchy with the cube view data in Power BI.

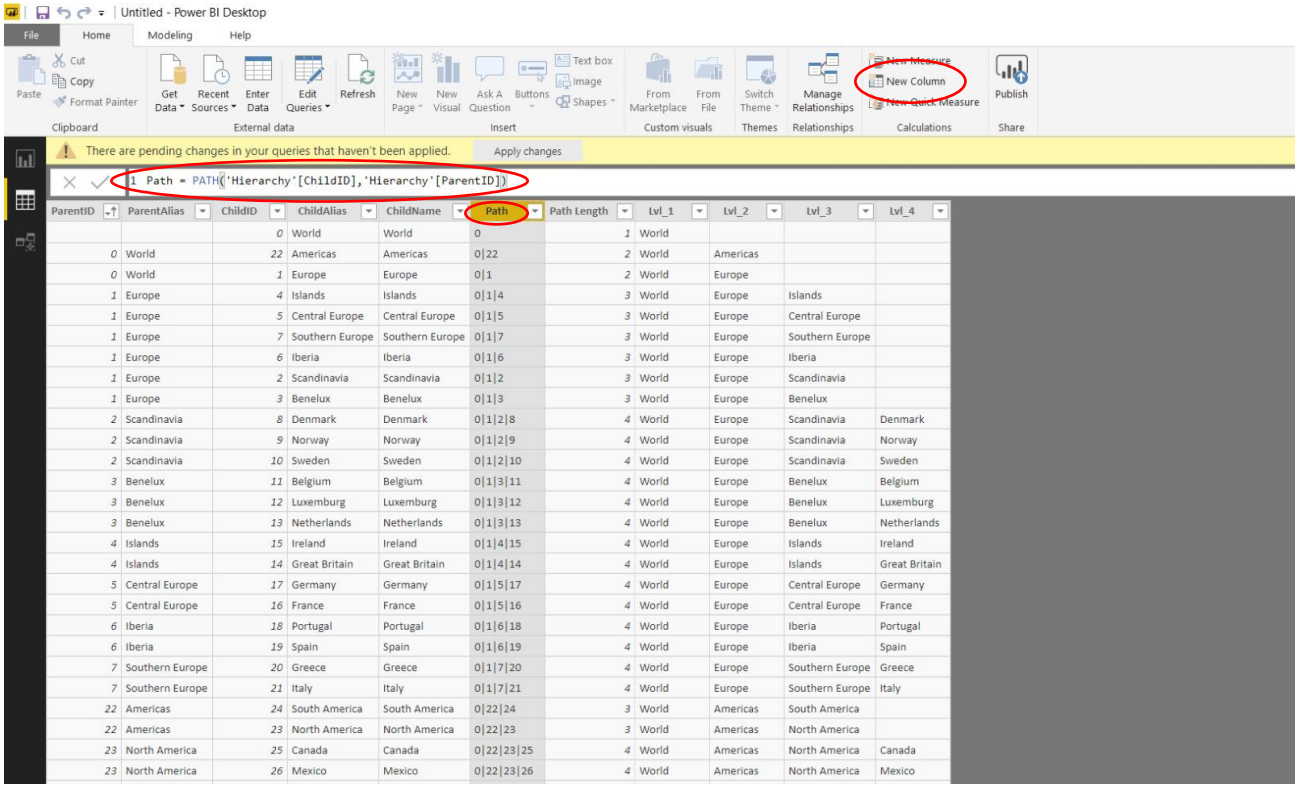
5.0 Method 2 – Parent-Child Extraction: Linking up the Cube data with hierarchy structure

If you are working with a more complex hierarchical structure, you can opt to use Method 2 instead. For this example, we will use the same "AllRegion" subset (that contains all the N level elements specified in the cube view, as well as the respective parents) that was used in the example above.

This time, while using the "Create Get Hierarchy URL" function, we will select the "Parent-Child" option instead. After importing the data into PowerBI, our output would like something like the screenshot below, with the "Parent ID", "Parent Alias", "ChildID", "ChildAlias" and "ChildName".

ParentID	ParentAlias	ChildID	ChildAlias	ChildName
		0	World	World
0	World	1	Europe	Europe
1	Europe	2	Scandinavia	Scandinavia
1	Europe	3	Benelux	Benelux
1	Europe	4	Islands	Islands
1	Europe	5	Central Europe	Central Europe
1	Europe	6	Iberia	Iberia
1	Europe	7	Southern Europe	Southern Europe
2	Scandinavia	8	Denmark	Denmark
2	Scandinavia	9	Norway	Norway
2	Scandinavia	10	Sweden	Sweden
3	Benelux	11	Belgium	Belgium
3	Benelux	12	Luxemburg	Luxemburg
3	Benelux	13	Netherlands	Netherlands
4	Islands	14	Great Britain	Great Britain
4	Islands	15	Ireland	Ireland
5	Central Europe	16	France	France
5	Central Europe	17	Germany	Germany
6	Iberia	18	Portugal	Portugal
6	Iberia	19	Spain	Spain
7	Southern Europe	20	Greece	Greece
7	Southern Europe	21	Italy	Italy
0	World	22	Americas	Americas
22	Americas	23	North America	North America
22	Americas	24	South America	South America

You will have to add a few columns to manipulate the data before it is useable by your end users. First, click on “New Columns” at the top right of the screen.



ParentID	ParentAlias	ChildID	ChildAlias	ChildName	Path	Path Length	Lvl_1	Lvl_2	Lvl_3	Lvl_4
		0	World	World	0	1	World			
0	World	22	Americas	Americas	0|22	2	World	Americas		
0	World	1	Europe	Europe	0|1	2	World	Europe		
1	Europe	4	Islands	Islands	0|1|4	3	World	Europe	Islands	
1	Europe	5	Central Europe	Central Europe	0|1|5	3	World	Europe	Central Europe	
1	Europe	7	Southern Europe	Southern Europe	0|1|7	3	World	Europe	Southern Europe	
1	Europe	6	Iberia	Iberia	0|1|6	3	World	Europe	Iberia	
1	Europe	2	Scandinavia	Scandinavia	0|1|2	3	World	Europe	Scandinavia	
1	Europe	3	Benelux	Benelux	0|1|3	3	World	Europe	Benelux	
2	Scandinavia	8	Denmark	Denmark	0|1|2|8	4	World	Europe	Scandinavia	Denmark
2	Scandinavia	9	Norway	Norway	0|1|2|9	4	World	Europe	Scandinavia	Norway
2	Scandinavia	10	Sweden	Sweden	0|1|2|10	4	World	Europe	Scandinavia	Sweden
3	Benelux	11	Belgium	Belgium	0|1|3|11	4	World	Europe	Benelux	Belgium
3	Benelux	12	Luxemburg	Luxemburg	0|1|3|12	4	World	Europe	Benelux	Luxemburg
3	Benelux	13	Netherlands	Netherlands	0|1|3|13	4	World	Europe	Benelux	Netherlands
4	Islands	15	Ireland	Ireland	0|1|4|15	4	World	Europe	Islands	Ireland
4	Islands	14	Great Britain	Great Britain	0|1|4|14	4	World	Europe	Islands	Great Britain
5	Central Europe	17	Germany	Germany	0|1|5|17	4	World	Europe	Central Europe	Germany
5	Central Europe	16	France	France	0|1|5|16	4	World	Europe	Central Europe	France
6	Iberia	18	Portugal	Portugal	0|1|6|18	4	World	Europe	Iberia	Portugal
6	Iberia	19	Spain	Spain	0|1|6|19	4	World	Europe	Iberia	Spain
7	Southern Europe	20	Greece	Greece	0|1|7|20	4	World	Europe	Southern Europe	Greece
7	Southern Europe	21	Italy	Italy	0|1|7|21	4	World	Europe	Southern Europe	Italy
22	Americas	24	South America	South America	0|22|24	3	World	Americas	South America	
22	Americas	23	North America	North America	0|22|23	3	World	Americas	North America	
23	North America	25	Canada	Canada	0|22|23|25	4	World	Americas	North America	Canada
23	North America	26	Mexico	Mexico	0|22|23|26	4	World	Americas	North America	Mexico

We will be using DAX statements to manipulate the data. The following list contains the new columns required along with their DAX formulas:

1) Path = PATH('Hierarchy'[ChildID],'Hierarchy'[ParentID])
2) Path Length = PATHLENGTH('Hierarchy'[Path])
3) Lvl_1 = LOOKUPVALUE('Hierarchy'[ChildAlias],'Hierarchy'[ChildID],PATHITEM('Hierarchy'[Path],1,INTEGER))
4) Add as many Levels as required (e.g. Lvl_2 = LOOKUPVALUE(….), Lvl_3 = LOOKUPVALUE(…))

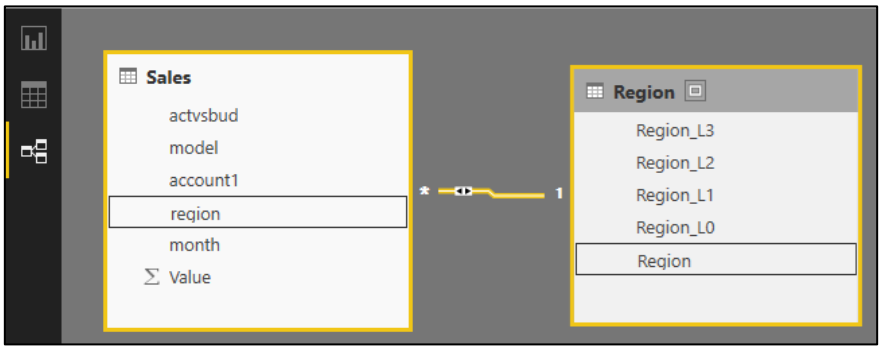
You can also refer to our live demo on our Youtube channel for reference: [Youtube link]

6.0 Linking up the Cube data with hierarchy structure

To link up the cube data with hierarchy information, we create a link between the cube data table with the hierarchy table.

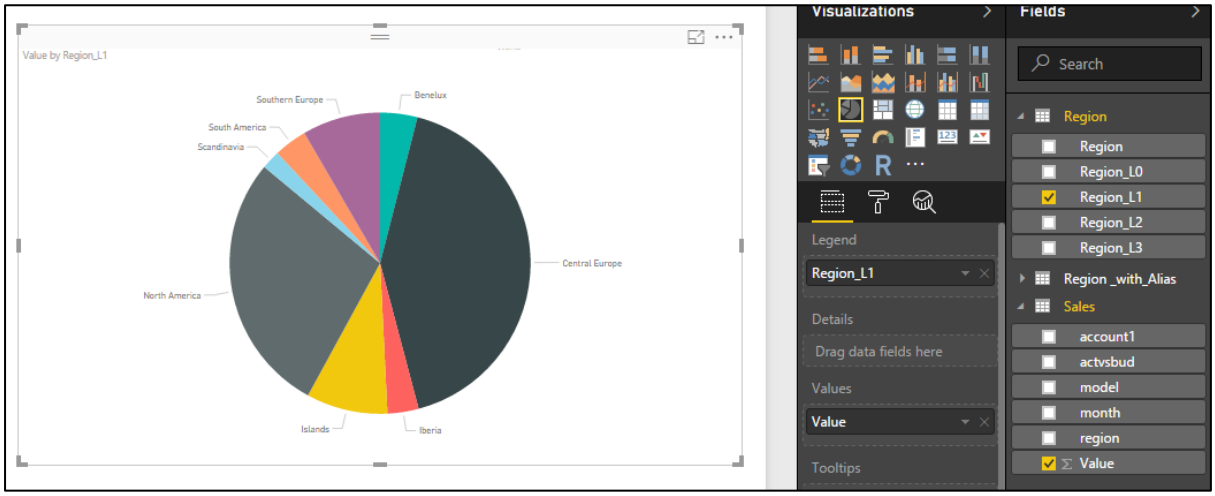
We assume that the cube data is extracted without the respective aliases for “Region” dimension.

The link can be easily created in Power BI by using the “Manage Relationships” function. Simply link up the “region” field in the cube data table and the hierarchy data table.



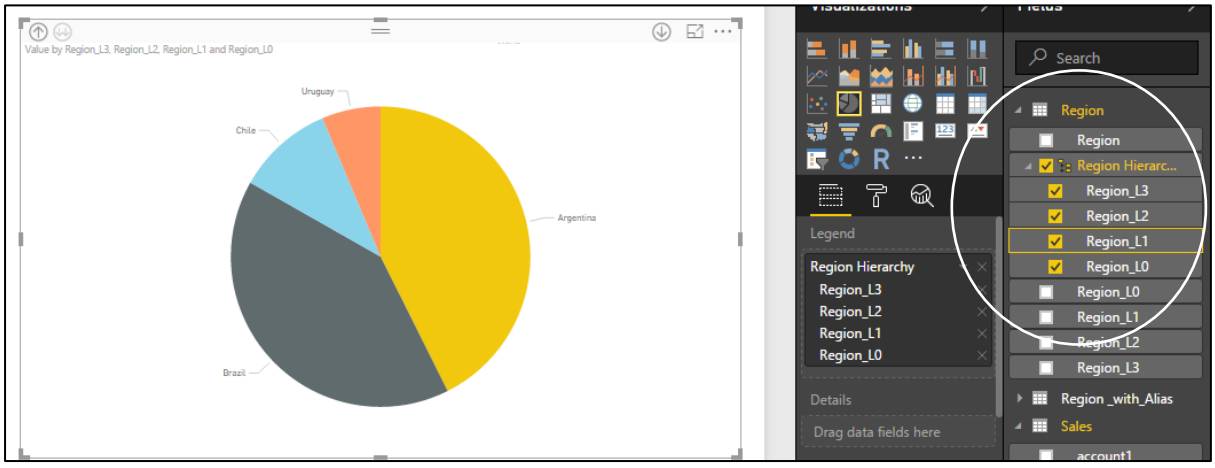
Once the relationship has been established, you can start selecting the various levels of the “Region” dimension in Power BI report.

In this example, we want to display the “Region_L1” data in our report. With all the required levels visible for selection, you can start selecting the hierarchy in Power BI.

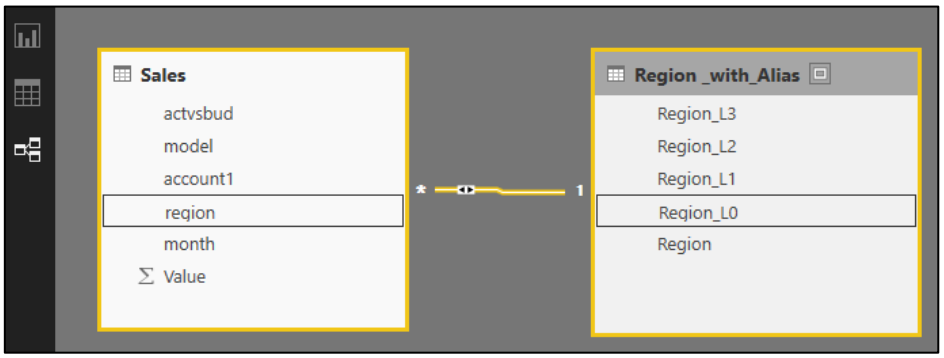


To further enhance the report, you can choose to create a hierarchy structure in Power BI similar to that in TM1. Please review Power BI guides on how to establish hierarchy.

The following is an example of the hierarchy defined in Power BI. Once a hierarchy has been defined, you can start using the Drill function in Power BI to navigate through the data.



For cube data extracted with aliases, you must have the equivalent alias extracted in the hierarchy through TMVGate. Instead of linking the element principal name, which is in the "Region" column, you will now choose the "Region_L0" column to establish the relationship.



By providing this flexibility in TMVGate, users can choose to extract cube data using the element principal name, and enrich it with the corresponding alias using the hierarchy extract.