

Adventures in Groovy – Part 39: First Common Parent

I can't tell you how many times I have been at a client and they wanted to replicate eliminations at the first common parent like HFM, or some other consolidations system. Maybe there is a good way to do this, but I could never find an efficient and effective way to accomplish it with Essbase calculations. Well, that is history. This is yet another example of how to solve an historically complex problem very simply.

What Is The First Common Parent

Eliminations functionality addresses the posting of inter-company eliminations in scenarios where a full legal consolidation model is not required, such as within a standard financial model. An example would be eliminating sales to another entity inside the organization so the total sales of the organization is not inflating the real sales of the organization. This is typically done at the first node that consolidates the two entities, or first common parent. In the example below, which will be used in the code below, we need to find the first common parent to do the eliminations for entity S253 and S592. The hierarchy below shows that the first parent of these two members is Mountain Division. This is the entity that will calculate and hold the eliminated sales.

The screenshot shows the Dimensions tool interface. The 'Company' dimension is selected. The hierarchy is: Tot_Company > US_Stores > West_Reg > Mountain_Div > AZ_Stores > S253 and S592. Red dashed arrows point from S253 and S592 to the table on the right.

Alias (English)	Data Storage
Total Company	Never Share
Total US Stores	Never Share
West Region	Never Share
Pacific Division	Never Share
Alaska Stores	Never Share
California Stores	Never Share
Hawaii Stores	Never Share
Oregon Stores	Never Share
Washington Stores	Never Share
Mountain Division	Never Share
Arizona Stores	Never Share
CAMELBACK AZ (253)	Never Share
PARADISE VALLEY AZ (114)	Never Share
ORO VALLEY AZ (114)	Never Share
FLAGSTAFF AZ (1084)	Never Share
PRESCOTT AZ (377)	Never Share
Colorado Stores	Never Share
Idaho Stores	Never Share
Montana Stores	Never Share
BILLINGS MT (592)	Never Share
MISSOULA MT (593)	Never Share
KALISPELL MT (1016)	Never Share
New Mexico Stores	Never Share
Nevada Stores	Never Share
Utah Stores	Never Share
Wyoming Stores	Never Share
South Region	Never Share
Midwest Region	Never Share
Northeast Region	Never Share

Surprisingly Easy With Groovy

The Groovy classes available to us have the ability to query metadata. This allows a calculation to return all kind of things, like the ancestors of members. Groovy takes over the rest by comparing the arrays returned.

For this example, the calculation prompts for two members. In an real-world example, these would likely be defined in the calculation, or maybe with UDAs or attributes. The prompts in this example, C1 and C2, are run time prompts connected to the hierarchy above. Once the members are defined, the next step is to query the ancestors of each of the two members.

First, a connection to the application that has the dimension and members is defined. Once that is done, a dimension object is created that is used to execute the queries. Since we need ancestors, we use IAncestors.

```
/*RTPS:{C1} {C2}*/  
Cube cube = operation.application.getCube("Fin")  
Dimension          companyDim          =  
operation.application.getDimension("Company")  
List               companyOne          =  
companyDim.getEvaluatedMembers("IAncestors(${rtps.C1.toString()  
)})", cube)  
List               companyTwo          =  
companyDim.getEvaluatedMembers("IAncestors(${rtps.C2.toString()  
)})", cube)
```

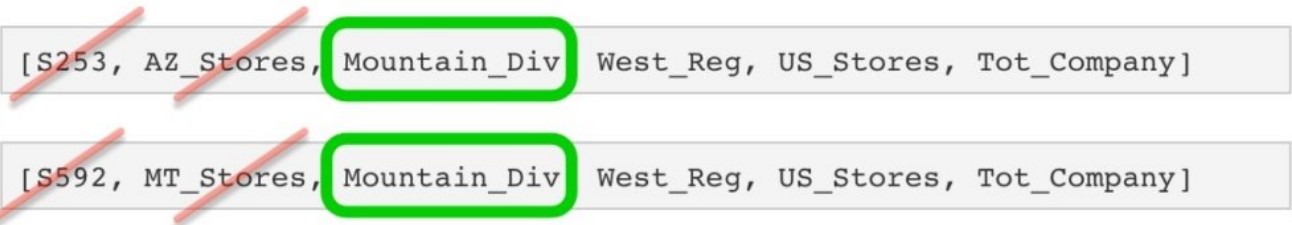
companyOne returns an array with the following values.

```
[S253, AZ_Stores, Mountain_Div, West_Reg, US_Stores,  
Tot_Company]
```

companyTwo returns an array with these values.

```
[S592, MT_Stores, Mountain_Div, West_Reg, US_Stores,  
Tot_Company]
```

The hard part, if you consider that hard, is over. Now that the two arrays are defined, a snazzy Groovy method are used. The intersect method will return the common elements of two lists.



```
[S253, AZ_Stores, Mountain_Div, West_Reg, US_Stores, Tot_Company]
```

```
[S592, MT_Stores, Mountain_Div, West_Reg, US_Stores, Tot_Company]
```

The order of the elements returned by the PBCS classes is ordered from the bottom of the hierarchy to the top. The first element would be the first common parent! This example doesn't illustrate it but this would work for staggered

hierarchies just the same.

```
List commonParents = companyOne.intersect(companyTwo)
println "First common parent for ${rtps.C1.toString()} and
${rtps.C2.toString()} is ${commonParents[0]}"
```

The println results in the following message to the job console.

```
First common parent for "S253" and "S592" is Mountain_Div
```

That is it boys and girls. In 6 lines of scripting (and it could be less as some variables are introduced to clearly articulate the process and methods), the first common parent is identified. The most difficult part would be the business logic to accomplish the actual business requirement.

All Done

Now that you know how to get the first common parent, this can be used to dynamically create the appropriate Essbase calculations to provide all the functionality that is needed. If you like this, share it with the community. Post comments if you have any questions. I love getting feedback.