Griffin Knight

info@lionpointgroup.com

March 20, 2020

STRATEGY. TECHNOLOGY. OPERATIONS

Anaplan Performance Series Part I: Calculation Structure and Performance Taxonomy

The goal of this article series is to give advanced Anaplan model builders and architects insight into how Anaplan works, along with actionable methods for improving model performance. In this post we discuss calculation structure, beginning with the introduction of blocks. We also introduce a new taxonomy for model performance. In the second & third article, we will look at how best to reduce the number of calculations in the model. The fourth & fifth article will focus on how to achieve optimal calculation efficiency.

Performance Overview

A large amount of performance decisions must be made on a case by case basis. This can make it difficult to diagnose the proper solution to a poorly performing model. For that reason, this article will first try to explain how the Anaplan engine calculates information, followed by examples and applications.

Anaplan Structure

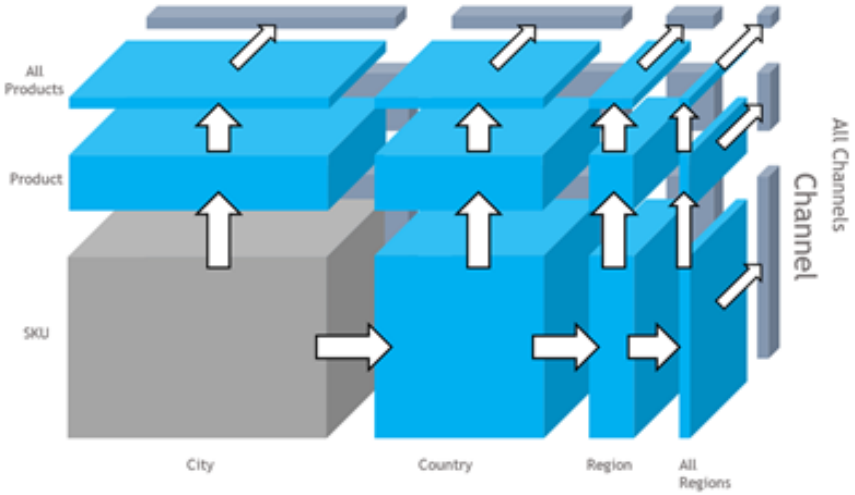
Line Items

The first thing to know when understanding Anaplan's architecture is that all calculations are done at the line item level. The result of this is a chain of dependencies from an output line item all the way to its source. A modules primary purpose is for the grouping of line items to enhance dashboard visualization, model cleanliness, and Imports/Exports. This means that when it comes to considerations of dimensionality in regards to performance, we are interested in line items as opposed to modules.

Blocks

In order to maximize calculation efficiency, Anaplan breaks all data into blocks. These blocks are made up of the dimensions of a line item, and each block cannot have any internal dependencies. This concept is what makes Anaplan far superior to Excel from a calculation standpoint. Where every cell in Excel needs to potentially refer to every other cell, Anaplan groups its calculations into blocks allowing faster computation by orders of magnitude. Let's look at an example.

Region	Product	Channel
City L3	SKU L2	Channel L1
Country L2	Product L1	All Channels
Region L1	All Products	
All Regions		



The above graphic is a visual representation of a line item. In this case it is dimensioned by 3 lists: City L3, SKU L2, and Channel L1. However, each of these lists has one or more parent hierarchies, which contribute to the block count. To count the number of blocks in a line item you find the product of the base dimensions plus the number of parents.

$$\prod_{i=p}^{d}(1+i)$$

d = Dimensions

p = # of Parents

Looking at our three dimensions, City L3 has 3 parents, SKU L2 has 2 parents, and Channel L1 has 1 parent. Applying this to the formula above we get:

$$d = 3, p_1 = 3, p_2 = 2, p_3 = 1$$
$$(1+3)(1+2)(1+1)$$
$$(4)(3)(2) = 24$$

If you were to count the number of blocks in the graphic above, it will equal 24. However, 23 of the 24 blocks are summaries, if summary methods were turned off, this line item would only have 1 block: (1 + 0) * (1 + 0) * (1 + 0).

Time & Versions

You may have noticed that Time and Versions act differently than other dimensions. For example you cannot SUM across them and a line item can reference itself through PREVIOUS() or NEXTVERSION(). But a line item without time or versions can never reference itself, what gives?

The answer is that every member of Time and Versions is treated as a separate block. One way to think about this is that each Time member (or Version) can be thought of as its own Line Item. Let's look at an example.

Block Example | Amount

		Q1 FY18	Q2 FY18	Q3 FY18	Q4 FY18	FY18	Q1 FY19	Q2 FY19	Q3 FY19	Q4 FY19	FY19
North America	Channel 1	0	0	0	0	0	0	0	0	0	0
	All Channels	0	0	0	0	0	0	0	0	0	0
All Regions	Channel 1	0	0	0	0	0	0	0	0	0	0
	All Channels	0	0	0	0	0	0	0	0	0	0

The above line item is dimensioned by Region L1, and Channel L1, each with 1 parent. If Time was not a dimension this line item would have 4 blocks (1+1) * (1+1). However, with 10 periods of time, this module now has 40 blocks 4 * 10. This flexibility can come with a cost though, as we will see later. We can now revise our formula to account for Time and Versions.

$$\left(\prod_{i=p}^{d}(1+i)\right)(T)(V)$$

Taxonomy

There are two components I think about when breaking down the performance of a model. One way to think about this is in the formula below.

Count x Efficiency = Performance

Note that it is much more nuanced than this, however, we believe it's an important first step to understanding how to make a model perform better. In general, a smaller model will tend to perform better than a large one, just as efficient calculations will perform better than non-efficient ones.

Additionally, calculations can be broken down into two pieces: inter-block and intra-block. The inter-block portion of a calculation is one that deals with other blocks (dimension intersections). The intra-block side deals with the calculations of cells within a block. Combining this with the calculation above we get a grid that we believe applies a solid taxonomy to breaking down model performance.

	Number of Calculations	Efficiency of Calculations
Inter-block	The number of blocks within a line item, or across line items.	The efficiency of block to block interactions
	Reducing the blocks you have.	Optimizing the blocks you have.
Intra-block	The number of cells within a block.	The efficiency of calculations across cells in a block
	Reducing the cells you have.	Optimizing the cells you have.

Here are some examples of performance considerations within each quadrant.

	Number of Calculations	Efficiency of Calculations
Inter-block	▪ Line Item Summaries ▪ Reduce Dependencies ▪ Time Ranges	▪ Non-Common Dimensions ▪ Calculation Sequence ▪ Selective Aggregation ▪ Dimension Order
Intra-block	▪ Extra Dimensionality ▪ Subsets	▪ Format Types ▪ Early Exits ▪ Formula Repetition

It is important to note that many of these items can arguably be placed in a number of these quadrants. However understanding the different “dimensions” of Anaplan performance through a structured taxonomy can assist in a model builders’ ability to diagnose and/or prevent the performance issues within a model.

In our next article, we will go into the upper-left side of the chart, discussing the inter-block considerations in regards to the number of calculations in a model.

Griffin Knight

info@lionpointgroup.com

March 20, 2020

STRATEGY. TECHNOLOGY. OPERATIONS

Anaplan Performance Series Part II: Inter-Block Number of Calculations

In Part I we discussed how Anaplan structures its data along with a new taxonomy for looking at performance considerations. In this article we will go into the ways in which we can reduce the number of blocks within a model.

	Number of Calculations	Efficiency of Calculations
Inter-block	The number of blocks within a line item, or across line items. Reducing the blocks you have. ▪ Line Item Summaries ▪ Reduce Dependencies ▪ Time Ranges	The efficiency of block to block interactions Optimizing the blocks you have. ▪ Non-Common Dimensions ▪ Calculation Sequence ▪ Selective Aggregation ▪ Dimension Order
Intra-block	The number of cells within a block. Reducing the cells you have. ▪ Extra Dimensionality ▪ Subsets	The efficiency of calculations across cells in a block Optimizing the cells you have. ▪ Format Types ▪ Early Exits ▪ Formula Repetition

Inter-block, Number of Calculations

Reducing the number of calculations in an Anaplan model is a simple and effective way to improve performance and reduce size. Reducing the number of calculations from an inter-block perspective means reducing the number of blocks within the model. The ways that this can be done is through line item summaries, reducing dependencies, and utilization of Time Ranges.

Line Item Summaries https://help.anaplan.com/anapedia/Content/Modeling/Build%20Models/Summary_Methods.html

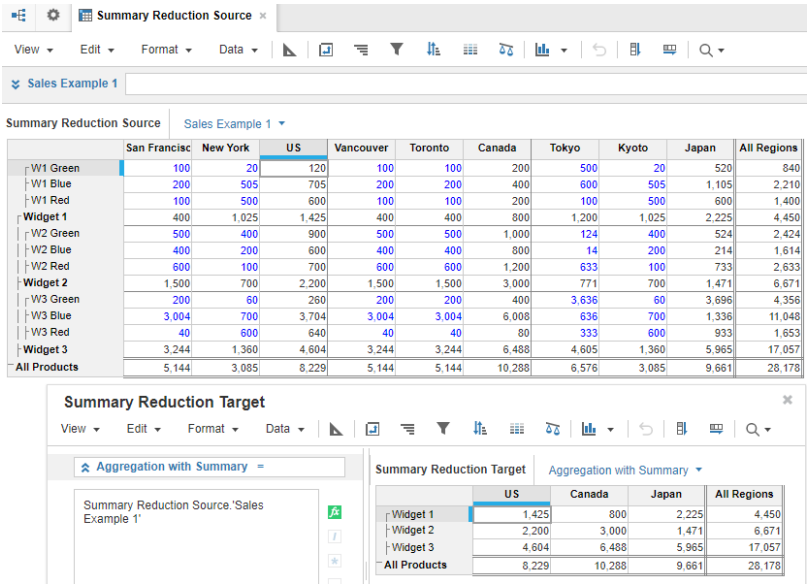
For number formatted cells, Anaplan turns Summaries on by default. Therefore, if the model builder is going quickly, or doesn't realize the potential impact, there can be a lot of useless blocks in the model. It is also important to note that the type of line item summary also can affect performance. For example, a summary method of Sum is much easier to calculate than a summary of Formula.

Let's go back to our example before when we introduced the concept of blocks. One line item dimensioned by 3 dimensions each with parents, has 24 blocks with summary methods turned on. By simply turning the summary method to "None" it drops to 1 block.

Region	Product	Channel
City L3	SKU L2	Channel L1
Country L2	Product L1	All channels
Region L1	All products	
All regions		

As a general rule of thumb, the only time line item summaries need to be turned on is if (1) the module is on a dashboard in which the value of the parent need to be shown or (2) it is needed for fringe calculation purposes.

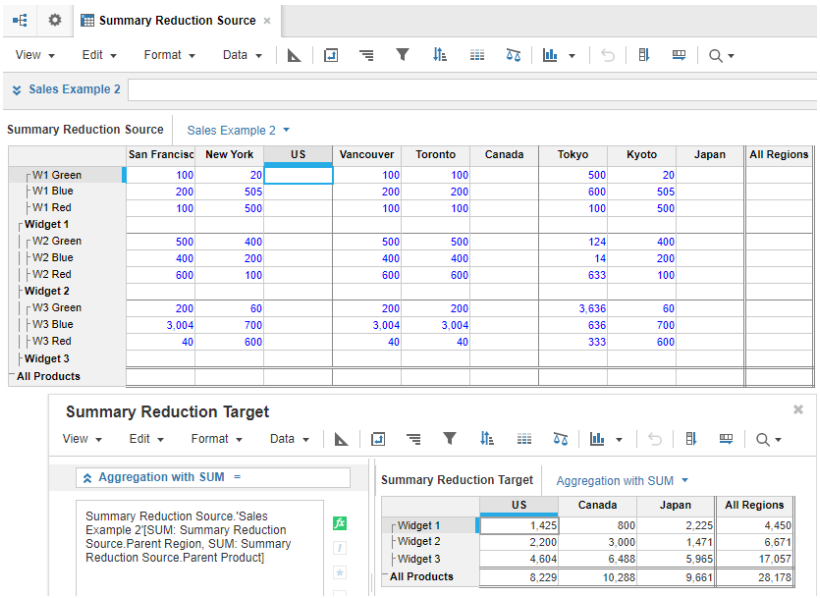
But how are we supposed to aggregate the child amounts without using the summary? The answer is that you can use an aggregation calculation instead.
https://help.anaplan.com/anapedia/Content/Calculation_Functions/CF_Aggregation_Functions.html)



	San Francisc	New York	US	Vancouver	Toronto	Canada	Tokyo	Kyoto	Japan	All Regions
W1 Green	100	20	120	100	100	200	500	20	520	840
W1 Blue	200	505	705	200	200	400	600	505	1,105	2,210
W1 Red	100	500	600	100	100	200	100	500	600	1,400
Widget 1	400	1,025	1,425	400	400	800	1,200	1,025	2,225	4,450
W2 Green	500	400	900	500	500	1,000	124	400	524	2,424
W2 Blue	400	200	600	400	400	800	14	200	214	1,614
W2 Red	600	100	700	600	600	1,200	633	100	733	2,633
Widget 2	1,500	700	2,200	1,500	1,500	3,000	771	700	1,471	6,671
W3 Green	200	60	260	200	200	400	3,636	60	3,696	4,356
W3 Blue	3,004	700	3,704	3,004	3,004	6,008	636	700	1,336	11,048
W3 Red	40	600	640	40	40	80	333	600	933	1,653
Widget 3	3,244	1,360	4,604	3,244	3,244	6,488	4,605	1,360	5,965	17,057
All Products	5,144	3,085	8,229	5,144	5,144	10,288	6,576	3,085	9,661	28,178

	US	Canada	Japan	All Regions
Widget 1	1,425	800	2,225	4,450
Widget 2	2,200	3,000	1,471	6,671
Widget 3	4,604	6,488	5,965	17,057
All Products	8,229	10,288	9,661	28,178

Above, we have an example of aggregation using summary methods. While this method is quick and easy, it leads to unnecessary blocks if the module is not used on a dashboard. Below we see an example of how we achieve the same result but by using [SUM:], with summaries turned off.



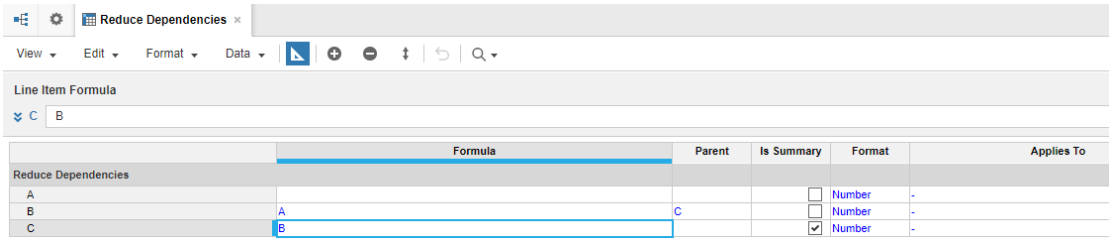
Using an Aggregation function leads to the same result, but without the excess blocks created from the summary method. Note that the Line items we are summing off of ('Summary Reduction Source'.'Parent Region' and 'Summary Reduction Source'.'Parent Product') are just PARENT(ITEM()) of the City and Widget dimensions, respectively.

David Smith goes into this concept in the first part of his article on reduction of calculations: https://community.anaplan.com/t5/Best-Practices/Reduce-Calculations-for-Better-Performance/ta-p/33667

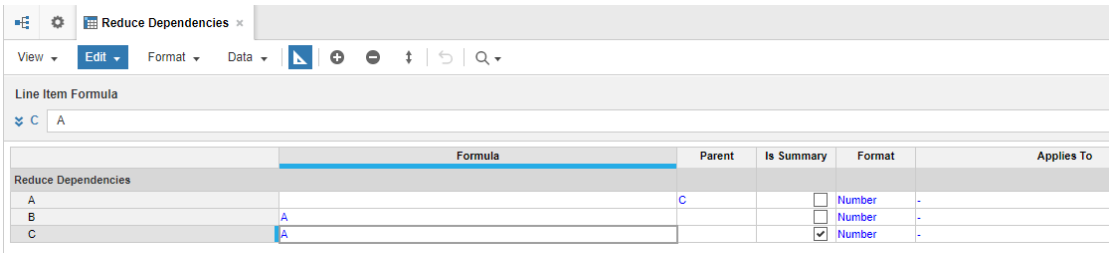
Reduce Dependencies

While Summary Methods was concerned with reducing the number of blocks within a line item, reducing dependencies can be an effective way to reduce the number of line items (and in turn blocks) in your model.

When a line item references another, it forms a dependency on the line item it is referencing. Below is an example of three-line items that are daisy chained together.



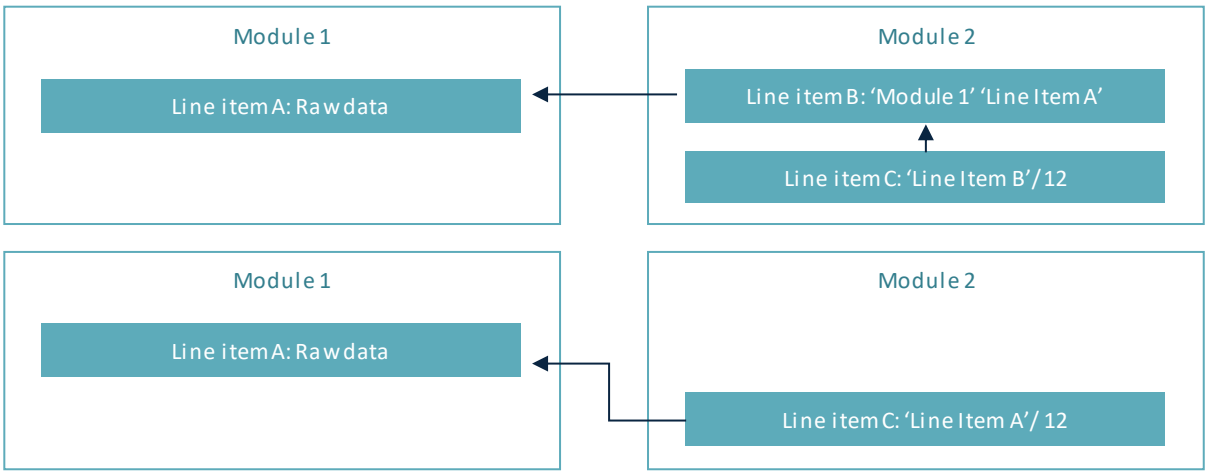
As we can see, line item C is dependent on B, which is dependent on A. While this is a basic example, we can see that since there is no change occurring with B, C can instead reference directly to A.



Note that in practice these line items would probably exist in different modules, making them more difficult to spot.

You may have noticed that we haven't actually reduced the number of blocks yet (I will go into the impact of just reducing dependencies in the calculation sequence section in Part IV). The ability to dispose of unused blocks comes after we have removed the dependencies. In our example above, why do we need both B & C? They both are referencing A so can't we just use A? If that is the case, we can delete B, C, or both. Deleting these line Items will reduce the number of blocks.

Let's look at another example that we commonly see in models. Let's say we have two modules, each with the same dimensions. Module 1 is the raw data module, with Module 2 being used for calculations. Many times, we see people recreate the line items from Module 1 in Module 2 with no calculations applied. The calculation line items then reference the "copied" line item in Module 2. While this may make it easier to think through calculations, in reality it is a wasted line item because all of the calculations can instead refer directly to the source line item in Module 1. Below is a graphic explaining this.



As you can see, line item B was essentially useless, and deleting it allows line item C to directly reference line item A.

Another example of this is Filters. Many modelers put a new filtering line item in each module that they want to filter. However, if the dimensions are the same, you can utilize the same filter across as many modules as you want. This has the benefit of not only reducing the number of blocks, but also allowing all filters to be controlled by one line item, making it much easier to apply changes to filters across the model. This same concept can apply to Conditional Formatting line items in different modules.

The entire concept of reducing dependencies has to do with the common maxim: "Calculate once, reference many times". This means that if a line item is not uniquely changing the referenced line, we don't need it. This will lead to the leanest model possible with only essential line items. We will caveat this with the fact that sometimes performance rules need to be broken for optimal dashboarding, however the impact is usually low.

Time Ranges & Versions (https://help.anaplan.com/anapedia/Content/Modeling/Dimensions/Time%20Ranges.htm)

As we learned before, each individual time period and version is treated as its own block. Therefore, we should be extra diligent about only including as many time periods and versions as necessary.

When it comes to Time, the entirety of the model time scale often is not necessary. This is especially true in models with very long time periods set. By utilizing time ranges you can set your time range to only include periods that are applicable to your calculations, hence reducing blocks.

Other ways to improve performance through block reduction is by not using the time dimension at all. One example of this is a situation we ran into with a client where we needed to calculate IRR on daily cash flows over the last 20 years. With 100 companies within 8 funds across 7,300 days (365 * 20), with different projections and scenarios applied, we had an IRR calculation on a line item with tens of thousands of blocks. Considering the IRR formula is already computationally expensive (We will go into this in another article), running scenarios on this was causing a 5+ second delay for the end user. The solution was to replace real time day with a fake time day, which reduced the delay to less than a second. By utilizing fake date, we divided our block count for that line item by 7,300.

It's important to note that we are only referring to Time Ranges in this section and not traditional Subsets, even though they are very similar in practice. This is again due to the special nature of Time, in that the reduction of time periods reduces blocks, while a normal subset does not.

For versions, this means going through each version and ensuring that it is necessary to the model. Adding new versions should be a much bigger decision than adding a new member to a regular list!

You should now know some of the methods that can be used to reduce the number of blocks in your model. In our next article we will go into how to reduce the size of the remaining blocks.

Griffin Knight

info@lionpointgroup.com

March 20, 2020

STRATEGY. TECHNOLOGY. OPERATIONS

Anaplan Performance Series Part III: Inter-Block Number of Calculations

In Part II we discussed the best way to reduce the number of blocks within a model. In this article we will go into the ways in which we can reduce the number of cells, and in-turn calculations, within a block.

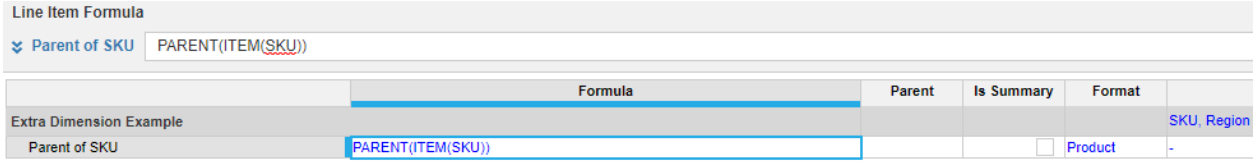
	Number of Calculations	Efficiency of Calculations
Inter-block	The number of blocks within a line item, or across line items. Reducing the blocks you have. ▪ Line Item Summaries ▪ Reduce Dependencies ▪ Time Ranges	The efficiency of block to block interactions Optimizing the blocks you have. ▪ Non-Common Dimensions ▪ Calculation Sequence ▪ Selective Aggregation ▪ Dimension Order
Intra-block	The number of cells within a block. Reducing the cells you have. ▪ Extra Dimensionality ▪ Subsets	The efficiency of calculations across cells in a block Optimizing the cells you have. ▪ Format Types ▪ Early Exits ▪ Formula Repetition

Intra-block, Number of Calculations

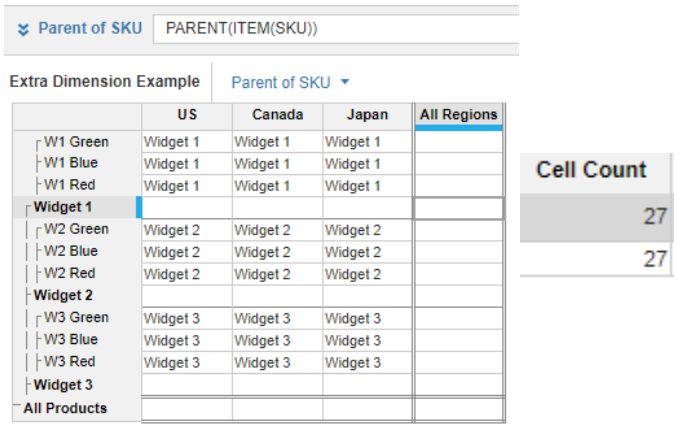
Now that we have reduced the number of blocks in our model, it is now time to reduce the number of cells within those blocks. Two ways to do this is through Extra dimensionality and Subsets.

Extra Dimensionality

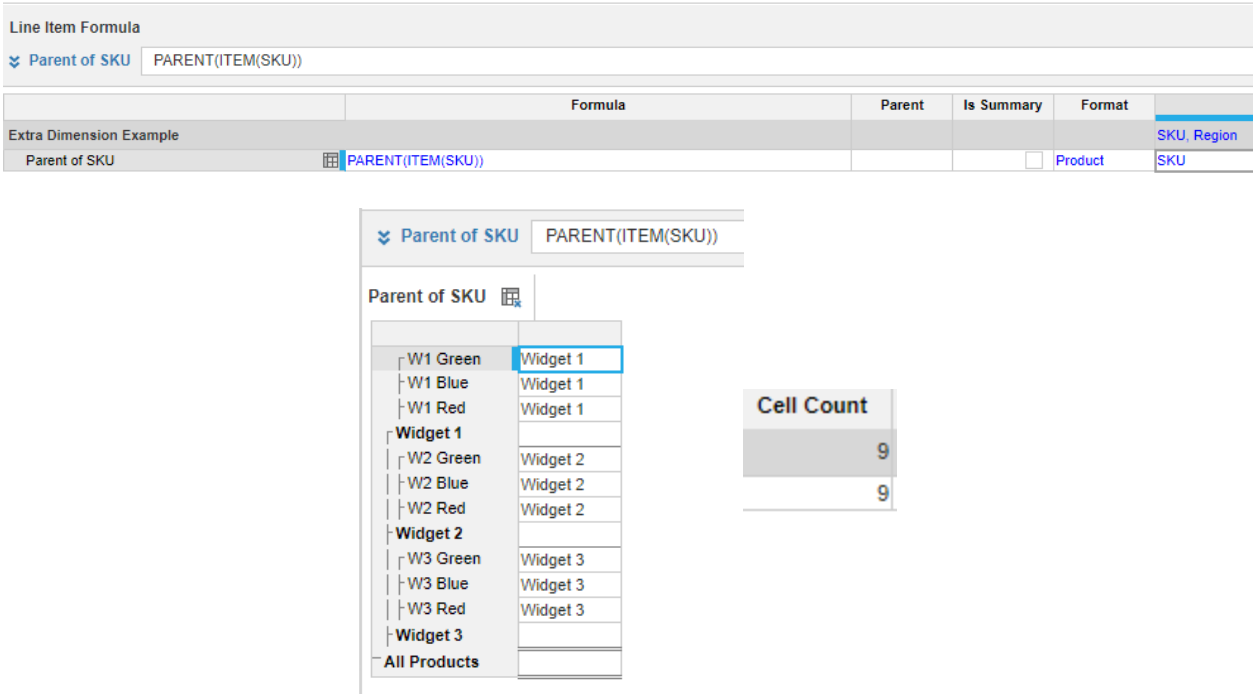
Extra Dimensionality is concerned with ensuring that a line item only has the necessary dimensions required for a calculation. A common example is the PARENT(ITEM()) formula. A line item that just uses PARENT(ITEM()) On Parent = (https://help.anaplan.com/anapedia/Content/Calculation_Functions/All/PARENT.html) On Item = (https://help.anaplan.com/anapedia/Content/Calculation_Functions/All/ITEM.html) never needs more than 1 dimension applied.



info@lionpointgroup.com
http://lionpointgroup.com/
https://www.linkedin.com/company/lionpoint-group/



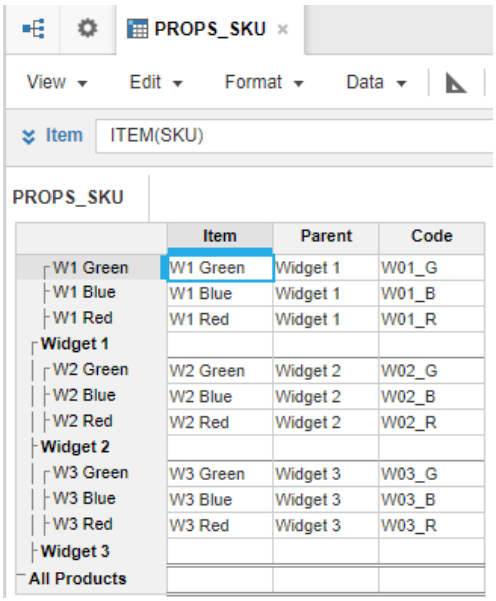
As we can see, there are only nine list members in the SKU list, meaning we only need to calculate nine cells. However, since the Region dimension is also applied, we are calculating 27 cells!



By removing the Region dimension (in turn making the line item a subsidiary view) we ensured that this formula is calculating on necessary cells (9 vs. 27).

Now you may be wondering why this is in the Intra-block section and not in the Inter-block section, didn't we reduce the number of blocks by removing a dimension with a parent? SKU has 2 parents and Region has 1 so we reduced the blocks in this line item from (1+2) * (1+1) = 6 to (1+2) = 3. While this is true, in our example there was no summary method, so in reality we did not reduce the block size, (1+0) * (1+0) vs. (1+0). What we did do was reduce the number of cells of our block by removing a dimension that was not needed.

A best practice way to ensure that you don't have extra dimensionality (or reducing the number of subsidiary views for model cleanliness purposes) is to have a "properties" module for each of your dimensions as the need arises. Common line items to include would be: ITEM(), PARENT(), CODE(), etc. This not only ensures that you don't have extra dimensionality, but it also makes sure you only are performing these calculations once in the model. Below is an example of a Properties module (our preferred syntax is usually: "PROPS_List Name").

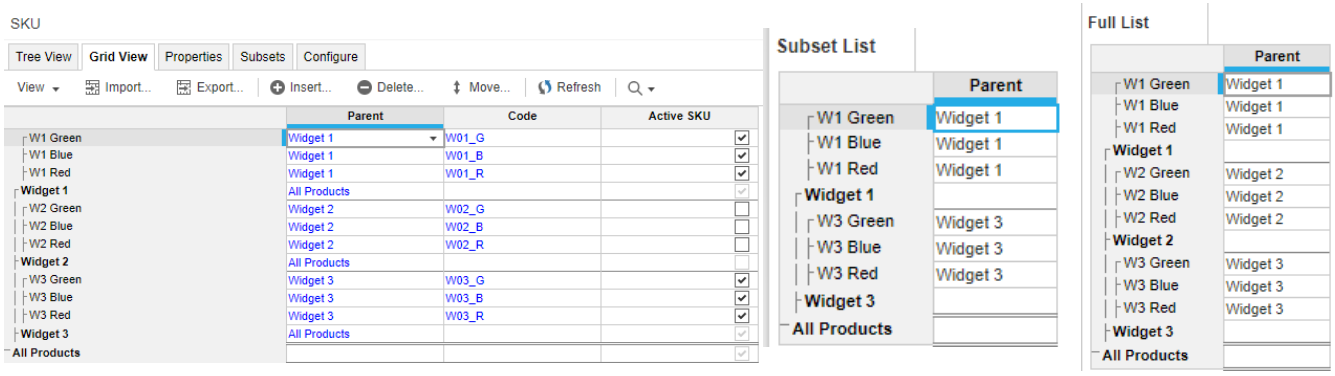


	Item	Parent	Code
W1 Green	W1 Green	Widget 1	W01_G
W1 Blue	W1 Blue	Widget 1	W01_B
W1 Red	W1 Red	Widget 1	W01_R
Widget 1			
W2 Green	W2 Green	Widget 2	W02_G
W2 Blue	W2 Blue	Widget 2	W02_B
W2 Red	W2 Red	Widget 2	W02_R
Widget 2			
W3 Green	W3 Green	Widget 3	W03_G
W3 Blue	W3 Blue	Widget 3	W03_B
W3 Red	W3 Red	Widget 3	W03_R
Widget 3			
All Products			

Being aware of your dimensionality as you are building is one of the key differentiators between an intermediate and an advanced model builder. The way that modules pre-set the dimensions of line items can make this easy to miss. However, every time a line item is created, you should ask yourself if the calculation requires all of the dimensions currently applied.

Subsets (https://help.anaplan.com/anapedia/Content/Modeling/Dimensions/List%20Subsets.html)

Perhaps the most straightforward way to reduce the number of cells in a block is to utilize subsets. The importance of subsets on performance grows with the size of the list in question, along with the number of list members that can be removed through a subset. In the example below, we have our full list of widgets, but let's say that Widget 2 was discontinued. This means that we don't need it for our forecasting purposes.



	Parent	Code	Active SKU
W1 Green	Widget 1	W01_G	☑
W1 Blue	Widget 1	W01_B	☑
W1 Red	Widget 1	W01_R	☑
Widget 1	All Products		☐
W2 Green	Widget 2	W02_G	☐
W2 Blue	Widget 2	W02_B	☐
W2 Red	Widget 2	W02_R	☐
Widget 2	All Products		☐
W3 Green	Widget 3	W03_G	☑
W3 Blue	Widget 3	W03_B	☑
W3 Red	Widget 3	W03_R	☑
Widget 3	All Products		☐
All Products			☐

	Parent
W1 Green	Widget 1
W1 Blue	Widget 1
W1 Red	Widget 1
Widget 1	
W3 Green	Widget 3
W3 Blue	Widget 3
W3 Red	Widget 3
Widget 3	
All Products	

	Parent
W1 Green	Widget 1
W1 Blue	Widget 1
W1 Red	Widget 1
Widget 1	
W2 Green	Widget 2
W2 Blue	Widget 2
W2 Red	Widget 2
Widget 2	
W3 Green	Widget 3
W3 Blue	Widget 3
W3 Red	Widget 3
Widget 3	
All Products	

Note that it may not be feasible to delete Widget 2 and its children since there is relevant historical data that needs to be preserved. However, for forecasting purposes it no longer needs to be included in the dimension, making it an ideal candidate for a Subset.

It is important to note that while Subsets can be extremely beneficial, if overused, they can become very burdensome to maintain.

We have now gone through how to reduce the number of blocks in a model (Link to Part I), and how to reduce the number of cells in a block. In the next two articles we will go into calculation efficiency from an inter and intra-block level.

Griffin Knight

info@lionpointgroup.com

March 20, 2020

STRATEGY. TECHNOLOGY. OPERATIONS

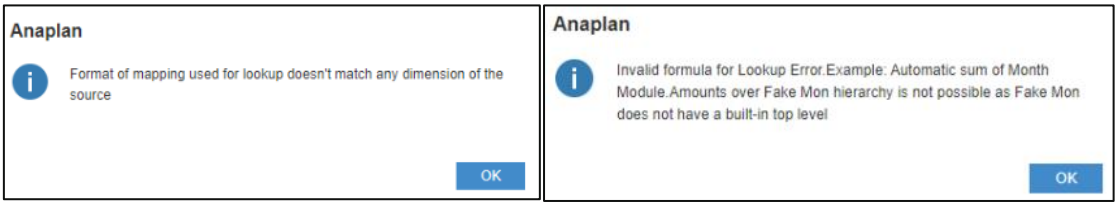
Anaplan Performance Series Part IV: Inter-Block Number of Calculations

In Parts II & III we went into how to improve model performance by reducing the number of calculations in the model. In Part IV and V we will explore methods that will improve the efficiency of existing calculations.

	Number of Calculations	Efficiency of Calculations
Inter-block	The number of blocks within a line item, or across line items. Reducing the blocks you have. ▪ Line Item Summaries ▪ Reduce Dependencies ▪ Time Ranges	The efficiency of block to block interactions Optimizing the blocks you have. ▪ Non-Common Dimensions ▪ Calculation Sequence ▪ Selective Aggregation ▪ Dimension Order
Intra-block	The number of cells within a block. Reducing the cells you have. ▪ Extra Dimensionality ▪ Subsets	The efficiency of calculations across cells in a block Optimizing the cells you have. ▪ Format Types ▪ Early Exits ▪ Formula Repetition

Inter-block, Efficiency of Calculations

We previously learned that Anaplan groups cells into blocks, enabling significantly higher levels of computation than calculating at a cell level. For this to work, blocks need to be able to communicate with each other to perform calculations. If the model builder does not provide the correct instructions, the blocks are unable to communicate, as you may have experienced:



Not only do we need to provide the blocks a way to communicate, the method we choose also impacts the speed at which the communication occurs. We want the blocks to talk to each other as fast as possible.

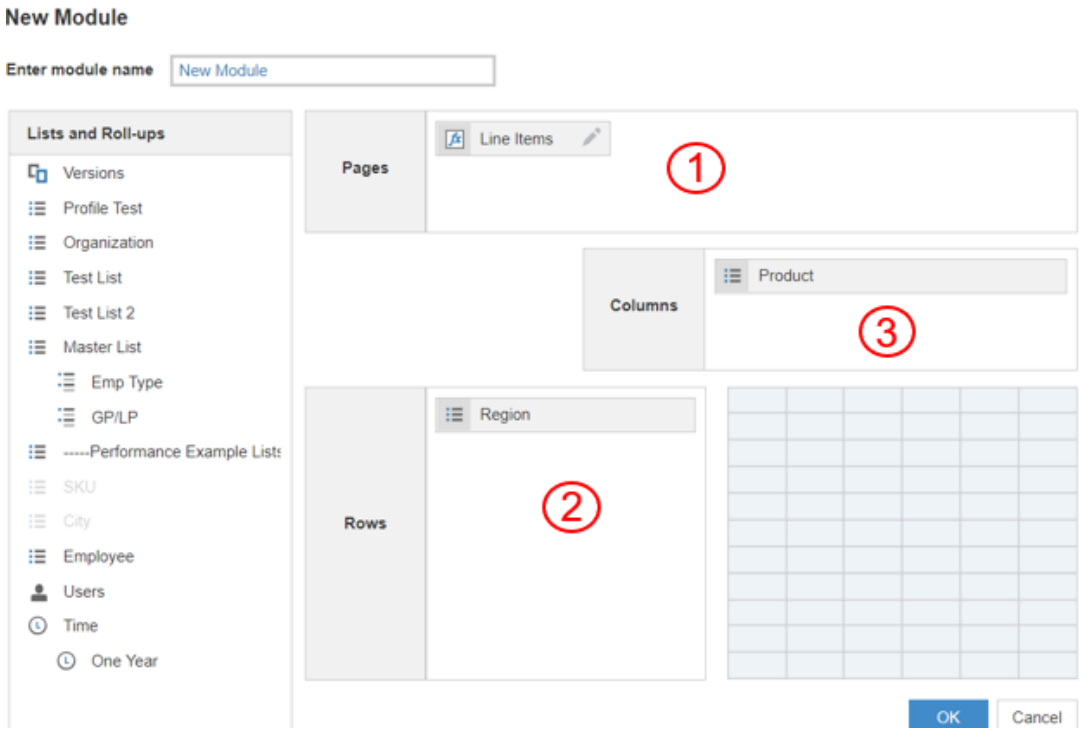
Four ways to do this is by understanding dimension order, calculations on non-common dimensions, calculation sequence, and selective aggregation.

Dimension Order (https://community.anaplan.com/t5/Best-Practices/Dimension-Order/ta-p/32934)

One of the simplest ways to improve the inter-block performance of your model is to look out for the order of your dimensions. Below is an example of two modules with the same dimensions, but in different order.

-----Dimension Order Examples-----		
Price Module		Product, Region
Units Sold Module		Region, Product

Dimensions get out of order during the creation of the module. Depending on where the dimensions are placed in the "New Module" screen, that is the priority with which they will be ordered. The priority of the dimensions goes: pages, rows, and columns.



Luckily, it's as easy to fix as it is to cause. In order to get your dimensions in the correct order, you simply click the ellipses in the "Applies To" of the module, and press "OK". This will set your dimensions to the "natural" order, which is prioritized by position in "Lists".

Now that we are aware of this issue and how to fix it, let's now look at why it causes issues in the first place.

When two blocks interact with each other, it needs to index each cell to be sure that the cells of one block match up with another. However, when the dimensions are in a different order, it makes this process slightly more time consuming. Below is an example of how Anaplan may index the two modules in our example.

As you may notice, Widget 1 in Canada is indexed with a 4 in the Units Sold module, but a 2 in the Price module. Luckily, Anaplan will notice this inconsistency and fix it before it goes about making the calculation. However, this fix comes at a cost to performance. Below is an example of how the system must reconcile the two line items before it calculates them.

Units Sold Module	Widget 1	Widget 2	Widget 3
US	1	2	3
Canada	4	5	6
Japan	7	8	9

Price Module	US	Canada	Japan
Widget 1	1	2	3
Widget 2	4	5	6
Widget 3	7	8	9

Price Module			
Product	Region	Index	Price
Widget 1	US	1	$ 3.00
Widget 1	Canada	2	$ 4.00
Widget 1	Japan	3	$ 3.00
Widget 2	US	4	$ 5.00
Widget 2	Canada	5	$ 6.00
Widget 2	Japan	6	$ 2.00
Widget 3	US	7	$ 6.00
Widget 3	Canada	8	$ 8.00
Widget 3	Japan	9	$ 5.00

Units Sold Module			
Product	Region	Index	Units Sold
Widget 1	US	1	200
Widget 2	US	2	100
Widget 3	US	3	140
Widget 1	Canada	4	150
Widget 2	Canada	5	60
Widget 3	Canada	6	700
Widget 1	Japan	7	250
Widget 2	Japan	8	190
Widget 3	Japan	9	100

Revenue Module		
Product	Region	Sales
Widget 1	US	$ 600
Widget 2	US	$ 500
Widget 3	US	$ 840
Widget 1	Canada	$ 600
Widget 2	Canada	$ 360
Widget 3	Canada	$ 5,600
Widget 1	Japan	$ 750
Widget 2	Japan	$ 380
Widget 3	Japan	$ 500

What would it look like if our dimensions were in the correct order?

Units Sold Module	Widget 1	Widget 2	Widget 3
US	1	2	3
Canada	4	5	6
Japan	7	8	9

Price Module	Widget 1	Widget 2	Widget 3
US	1	2	3
Canada	4	5	6
Japan	7	8	9

Price Module			
Product	Region	Index	Price
Widget 1	US	1	$ 3.00
Widget 2	US	2	$ 5.00
Widget 3	US	3	$ 6.00
Widget 1	Canada	4	$ 4.00
Widget 2	Canada	5	$ 6.00
Widget 3	Canada	6	$ 8.00
Widget 1	Japan	7	$ 3.00
Widget 2	Japan	8	$ 2.00
Widget 3	Japan	9	$ 5.00

Units Sold Module			
Product	Region	Index	Units Sold
Widget 1	US	1	200
Widget 2	US	2	100
Widget 3	US	3	140
Widget 1	Canada	4	150
Widget 2	Canada	5	60
Widget 3	Canada	6	700
Widget 1	Japan	7	250
Widget 2	Japan	8	190
Widget 3	Japan	9	100

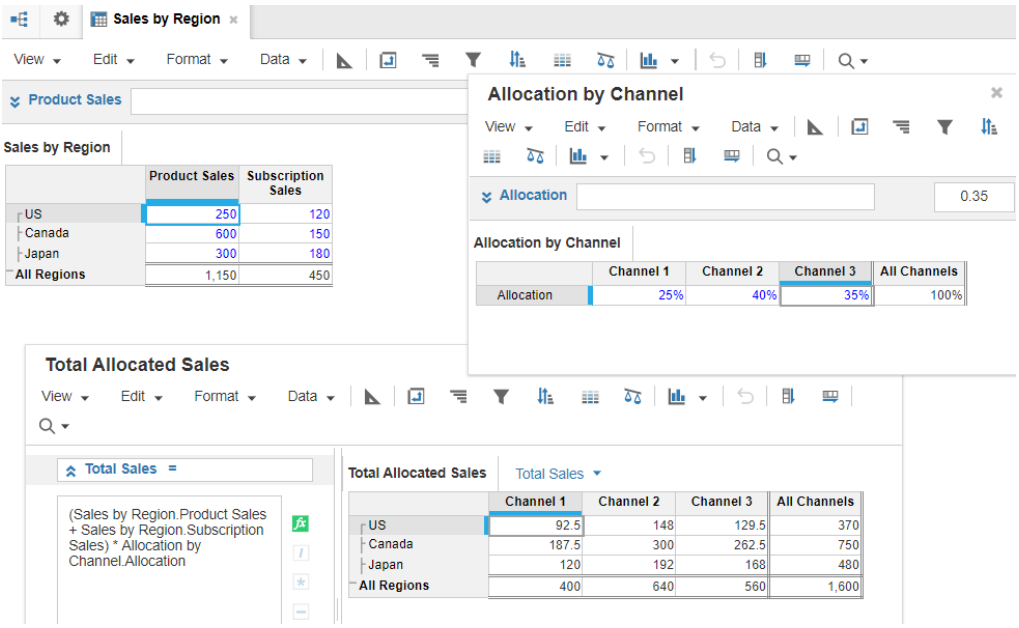
Revenue Module		
Product	Region	Sales
Widget 1	US	$ 600
Widget 2	US	$ 500
Widget 3	US	$ 840
Widget 1	Canada	$ 600
Widget 2	Canada	$ 360
Widget 3	Canada	$ 5,600
Widget 1	Japan	$ 750
Widget 2	Japan	$ 380
Widget 3	Japan	$ 500

Without having to first reconcile the index of the modules, the engine can get straight to calculating. As you may be able to tell, the indexing process gets longer the more cells we are dealing with.

Ensuring the order of your dimensions is one of the quickest and easiest ways to ensure that your blocks are communicating as efficiently as possible.

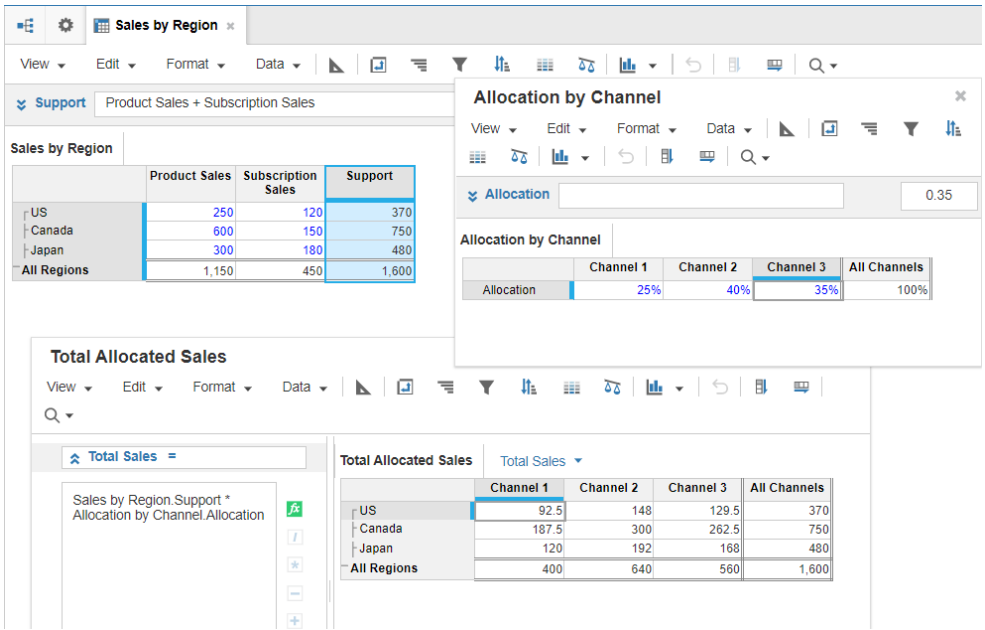
Calculations on Non-Common Dimensions

One way that inter-block performance can be evaluated is to look out for calculations on non-common dimensions. This is best explained with an example. Let's say we have a module called ‘Sales by Region’, which is dimensioned by Region. In this module we have two line items: ‘Product Sales’ and ‘Subscription Sales’. Now let's say we have a second module: ‘Total Allocated Sales’. This module is dimensioned by Region as well, but it is also dimensioned by Channel. The Total Sales module needs to take the total sales of the Region and multiply it by the channel allocation %. Spend some time looking at the example below.



What is wrong with the above scenario? The issue is that you are doing a calculation on a ***non-common*** dimension. The 'Total Sales' line item is adding together the product sales and subscription sales line items which are in a different block. The sales line items are in a block that is made up of just the Region dimension, the Total Sales is a block consisting of Region ***and*** Channel.

This is the root of the issue; you are calculating a Region only calculation with Channel applied as well. The better way to do this is to perform the sum of the sales line items on a ***common*** dimension. To achieve this, you could create a support line item that sums the Region sales together before introducing the Channel dimension. Below is an example showing this.



Using a support line item to maintain calculations over common dimensions allows us to ensure optimal inter-block communication & performance.

Calculation Sequence - Line Items

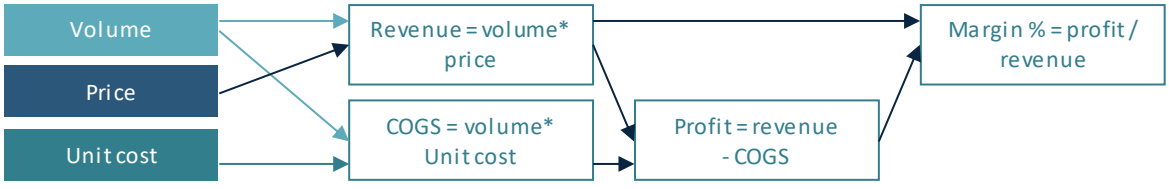
Explaining calculation sequence is best done through a metaphor. Imagine a toll bridge, in which each car is a required calculation and the toll payment is the execution of the calculation. What can we do to process as many calculations as fast as possible? In Articles II & III our answer would have been to reduce the numbers of cars on the road. Optimizing calculation sequence would be equivalent to building more lanes.

	Formula	Example
Volume	No Formula	100
Price	No Formula	$ 5.00
Unit Cost	No Formula	$ 3.00
Revenue	Volume * Price	$ 500
COGS	Volume * Unit Cost	$ 300
Profit	Revenue - COGS	$ 200
Margin %	Profit / Revenue	40%

Anaplan cores have 144 threads that can be used in parallel to perform calculations, these can quite literally be thought of as toll bridge lanes our cars can use. The trick however is making sure that our formulas make use of all the lanes. If we have 5 cars that are chained together, they will all have to use the same lane, even if other lanes are empty.

If our goal is to calculate Margin % (Profit / Revenue), we first need to calculate Revenue (Volume * Price), followed by COGS (Volume * Unit Cost) to get Profit (Revenue - COGS). Below is a representation of how these calculations would need to be ordered so we can get to our end goal of Margin %.

Three waves



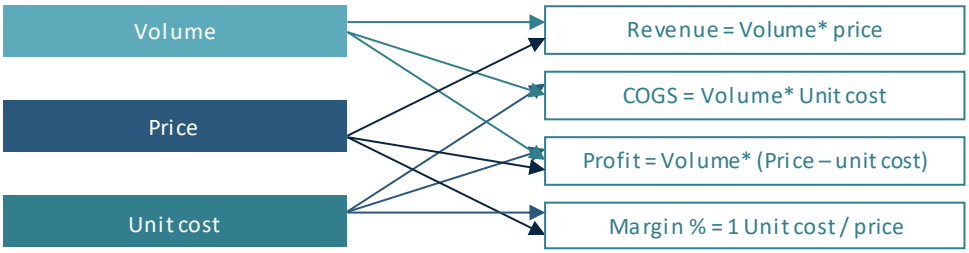
As you can see, in order to get to Margin %, we need to first calculate Profit, which requires us to first calculate Revenue and COGS. Each of these steps is called a wave, which can be thought of as the number of cars that are chained together. Here we have a series of cars that must use the same toll lane despite 143 other lanes being open!

How can we maximize our utilization of Anaplan's 144 threads? Let's imagine we wrote our formulas a little differently.

	Formula	Example
Volume	No Formula	100
Price	No Formula	$ 5.00
Cost	No Formula	$ 3.00
Revenue	Volume * Price	$ 500
COGS	Volume * Unit Cost	$ 300
Profit	Volume * (Price - Unit Cost)	$ 200
Margin %	(1 - Unit Cost / Price)	40%

As you can see, we get the exact same results but now each of our 4 calculation line items only refers to the data line items, and in turn is not dependent on any other calculation.

One wave

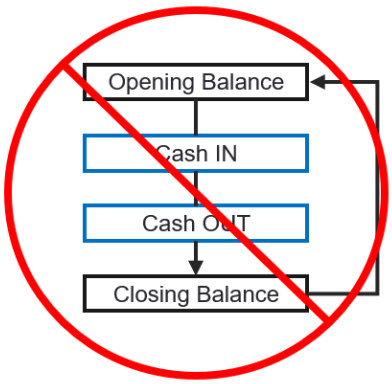


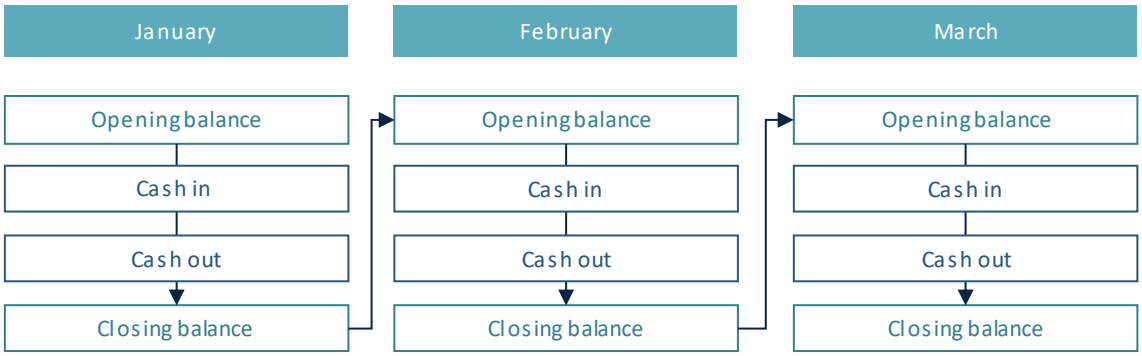
With these new formulas, we can now perform all of the calculations in one wave as opposed to three, and all of our cars can use their own lane in the toll bridge.

We believe that calculation sequence is one of the most important performance considerations when it comes to inter-block calculation efficiency.

Now in our example we were only looking at line item blocks in isolation. This concept applies in the same way to Time & Versions, which we know from Part I work the same way as line items in their structure.

Circular reference





Opening Balance | PREVIOUS(Closing Balance)

Time Calculation

	Jan 19	Feb 19	Mar 19	Apr 19	May 19
Opening Balance	0	133	300	628	258
Cash IN	515	801	903	162	403
Cash OUT	382	634	575	532	224
Closing Balance	133	300	628	258	437

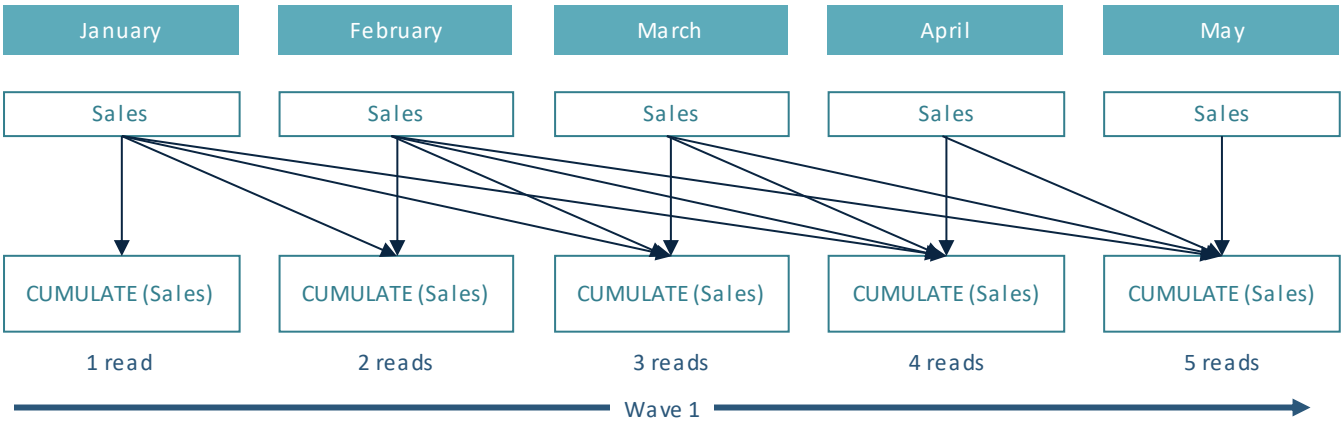
As you can see, using PREVIOUS() (https://help.anaplan.com/anapedia/Content/Calculation_Functions/All/PREVIOUS.html) on a time dimension means that we avoid circular references because we are referencing different blocks.

The reason this is important is because it works exactly like calculation sequence in line items: using PREVIOUS (or NEXT) has the potential to create a massive chain of blocks. Imagine you have a 5-year timescale on months using previous, that is the equivalent of 60 line items each daisy chained together. The last month cannot be calculated until the one before it has been calculated, all the way to the first month.

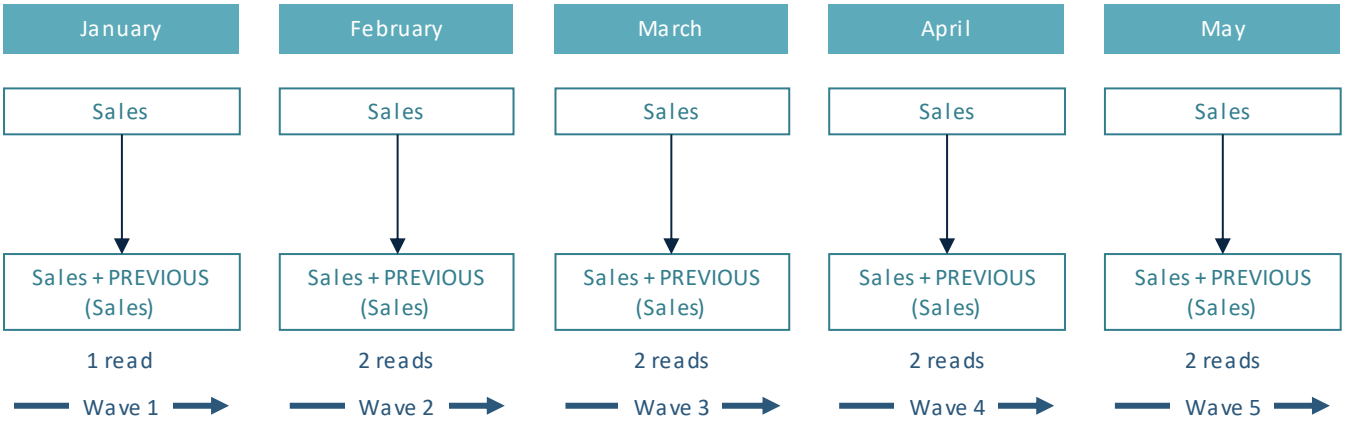
With that fact known, it's important to keep an eye on your time dimension if looking for performance improvements, especially if the timescale is large enough.

There is also another option: Cumulate (https://help.anaplan.com/anapedia/Content/Calculation_Functions/All/CUMULATE.html)

CUMULATE() is interesting in that it is the opposite of +PREVIOUS() from a technical level. Where +PREVIOUS() requires each line item to be calculated in order, CUMULATE() can calculate simultaneously. CUMULATE() however, is going to require significantly more calculations (or “reads”) than +PREVIOUS() would, despite all processing concurrently. In CUMULATE() each period needs to refer to every time period behind it - which can lead to a lot more calculations.



Compare this with +PREVIOUS()



So to continue with our beaten metaphor: CUMULATE() has a large number of cars but they can use their own lane, +PREVIOUS() has less cars but they must follow each other through a lane. Below is a chart showing the difference.

Timescale	Periods	CUMULATE reads	CUMULATE waves	PREVIOUS reads	PREVIOUS waves
1 Yr in Months	12	78	1	23	12
4 Yrs in Weeks	209	21,945	1 (2)	417	209
4 Yrs in Days	1,461	1,067,991	1 (10)	2,921	1,461

What this means in practice is that it depends. Sometimes CUMULATE() will be more efficient than +PREVIOUS(), but sometimes not. The rule of thumb should be that on a larger time scale (> a few hundred), you may need to be wary of CUMULATE(), since the number of reads required is so great.

I would also note that, while it doesn't necessarily pertain to performance, you will often be able to avoid circular references if you don’t use CUMULATE(), since it effectively kills the independence of your time blocks.

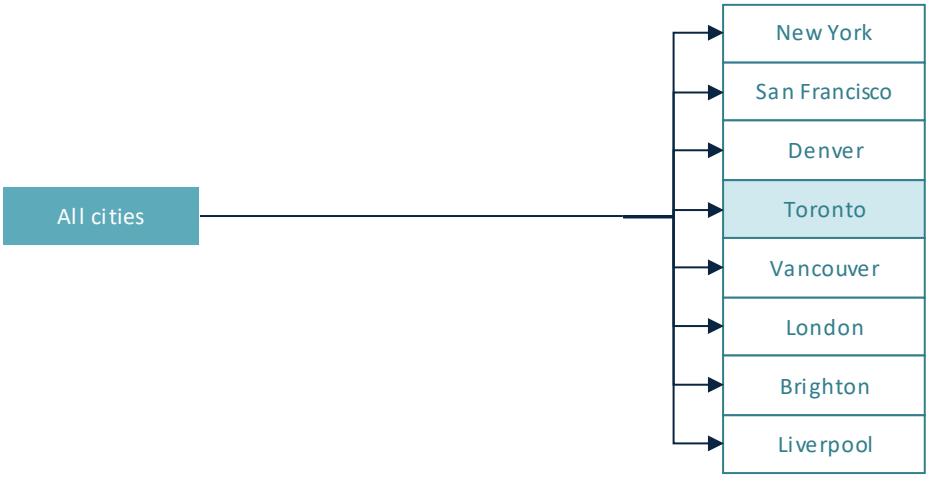
Selective Aggregation - Lists

Our last section for inter-block calculation efficiency is Selective Aggregation. Selective Aggregation is the idea that Anaplan will recalculate blocks of data in order to calculate a parent block.

Below is an example of a line item that is dimensioned by city.

Single level aggregation

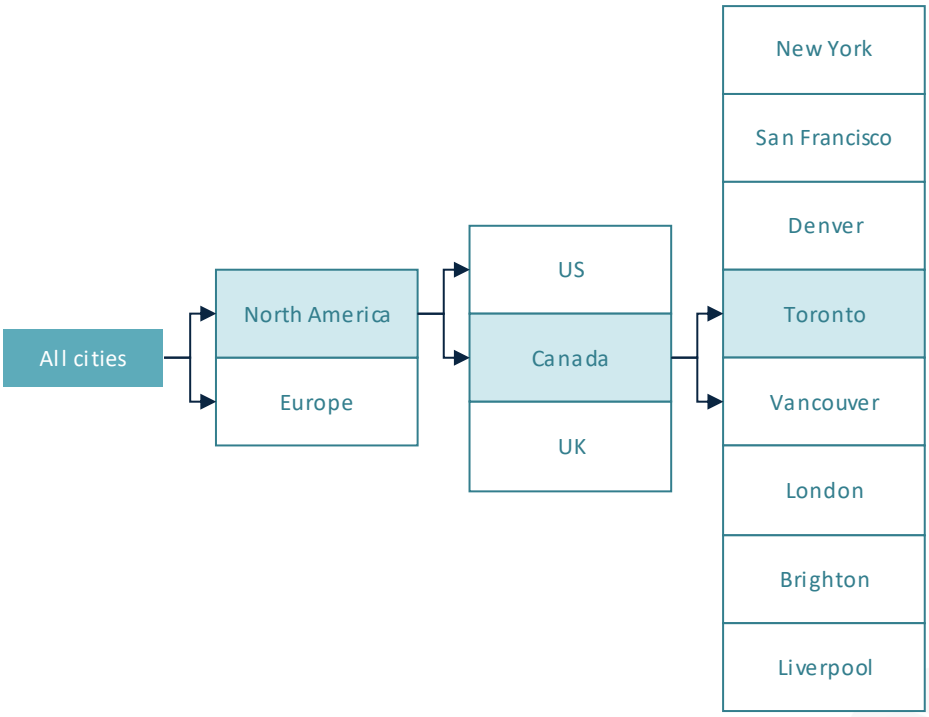
One level example	Sales
New York	214
San Francisco	643
Denver	534
Toronto	646
Vancouver	6,433
London	4,326
Brighton	245
Liverpool	2,354
All cities	**15,395**



If Toronto were to change, all other cities would need to be calculated in order to recalculate the "All Cities" top level. If there were no top level, this recalculation would not need to occur. However, the more levels we add to the hierarchy, the less calculations need to occur. Below is an example of the same list but with a multi-level hierarchy.

Multi level aggregation

Multi level example	Sales
New York	214
San Francisco	643
Denver	534
US	**1,391**
Toronto	646
Vancouver	6,433
Canada	**7,079**
North America	**8,470**
London	4,326
Brighton	245
Liverpool	2,354
UK	**6,925**
Europe	**6,925**
All regions	**15,395**



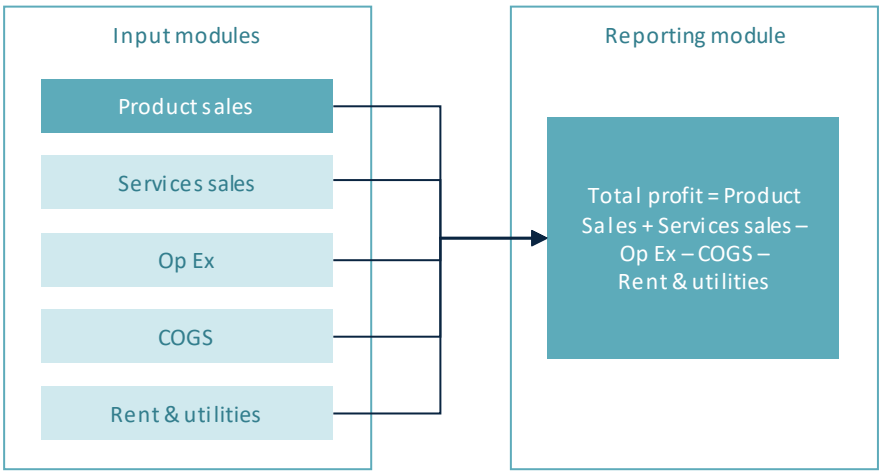
In this scenario, if Toronto were to change, it would require only Vancouver to be recalculated (and in turn the parents). This introduces the idea that a very large list can benefit from even an arbitrary level above it, to prevent large recalculation events.

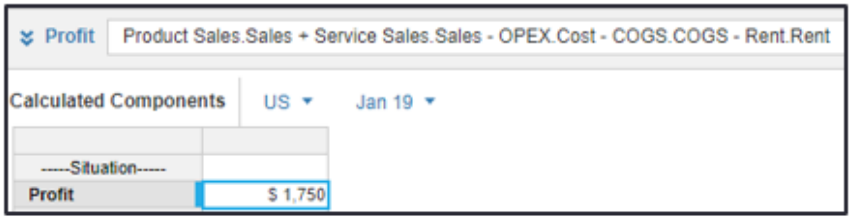
You may have noticed that these two examples look similar to the CUMULATE() and PREVIOUS() phenomenon we saw earlier. This is because they are - while multi-level aggregation can reduce the number of calculations, it introduces a chain of calculations that must occur before another can begin.

Selective Aggregation - Line Items (https://community.anaplan.com/t5/Best-Practices/Formula-Structure-for-Performance/tap/33177)

Just as levels in a list affect recalculation, so do line item hierarchies. We don't usually think of line items belonging to hierarchies (unless they are formally displayed in a Line Item Subset), but they effectively are when it comes to recalculation.

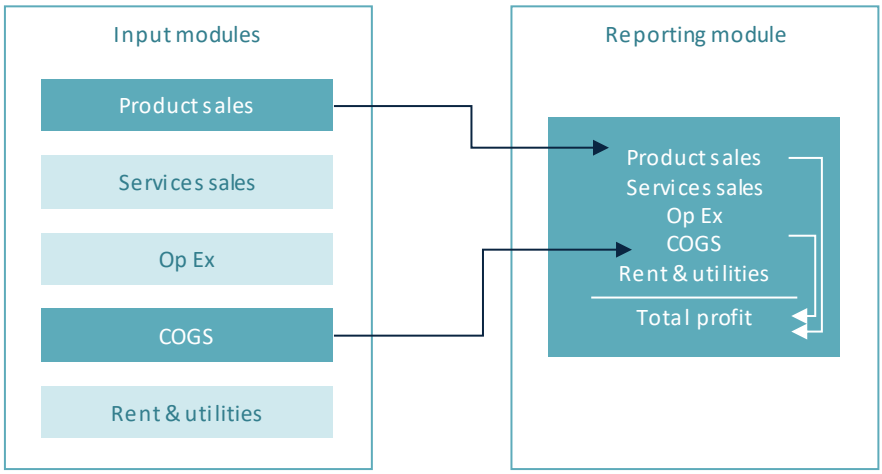
Below is an example, pulled from Anapedia, of a line item for profit that references 5 different line items, each in their own modules.

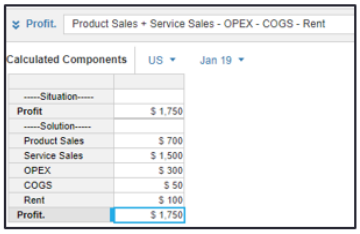


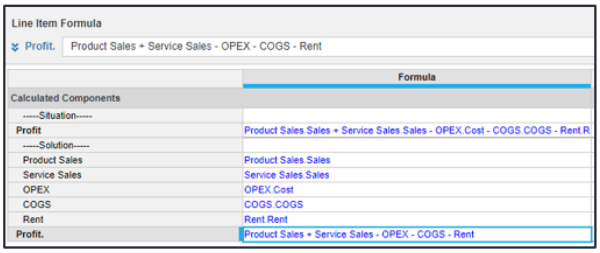


The issue with this, is that if one of those line items recalculates, all the others will have to as well. This can become an issue if those modules have complicated logic.

The solution is the same as what we did for our city list: introduce an arbitrary parent level.







By adding new line items that act as a parent, it "protects" us from having to recalculate at the downstream line items.

Now adding random line items (and parent levels) goes directly against some of the items discussed in part II & III. This is very much the balancing act that is Anaplan architecture. Sometimes more cells/blocks can improve performance if they can reduce unnecessary calculations.

We have now looked at how to improve the inter-block performance of our model, in the final article, we will look at intra-block performance.

Griffin Knight

info@lionpointgroup.com

March 20, 2020

STRATEGY. TECHNOLOGY. OPERATIONS

Anaplan Performance Series Part V: Inter-Block Number of Calculations

In Parts II & III we went into how to improve model performance by reducing the number of calculations in the model. In Part IV and V we will explore methods that will improve the efficiency of existing calculations.

	Number of Calculations	Efficiency of Calculations
Inter-block	The number of blocks within a line item, or across line items. Reducing the blocks you have. ▪ Line Item Summaries ▪ Reduce Dependencies ▪ Time Ranges	The efficiency of block to block interactions Optimizing the blocks you have. ▪ Non-Common Dimensions ▪ Calculation Sequence ▪ Selective Aggregation ▪ Dimension Order
Intra-block	The number of cells within a block. Reducing the cells you have. ▪ Extra Dimensionality ▪ Subsets	The efficiency of calculations across cells in a block Optimizing the cells you have. ▪ Format Types ▪ Early Exits ▪ Formula Repetition

Intra-block, Efficiency of Calculations

The final piece of the performance quadrant is intra-block calculation efficiency. This pertains to the efficiency and speed at which a block itself is calculated.

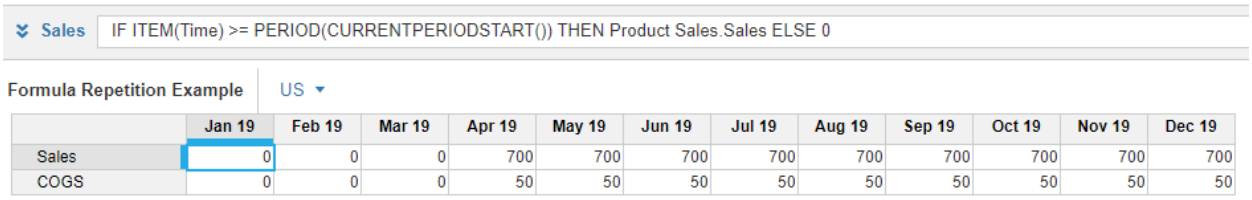
If inter-block efficiency was concerned with utilizing as many toll lanes as possible, intra-block efficiency deals with the speed of said cars.

There are three performance considerations for intra-block efficiency: Formula Repetition, Early Exits, and Format Types.

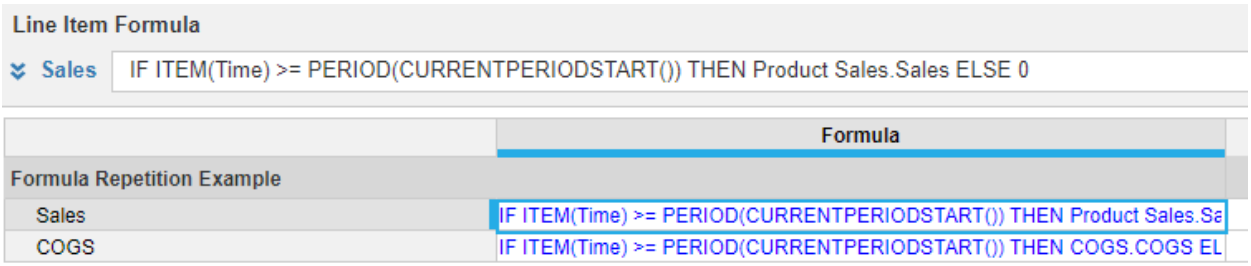
Formula Repetition

There is a common adage in the Anaplan world that says, “calculate once, reference many times”. Dealing with formula repetition means putting this into practice.

When a model has a high degree of formula repetition, there are blocks of information (line items) that both perform the same calculation, when it really only needs to be done once. Below is an example of two line items that both want to use logic for time periods in the future.



	Jan 19	Feb 19	Mar 19	Apr 19	May 19	Jun 19	Jul 19	Aug 19	Sep 19	Oct 19	Nov 19	Dec 19
Sales	0	0	0	700	700	700	700	700	700	700	700	700
COGS	0	0	0	50	50	50	50	50	50	50	50	50



For readability, here is both formulas written out in Sublime:

Sales

```
IF
    ITEM(Time) >= PERIOD(CURRENTPERIODSTART())
THEN
    Product Sales.Sales
ELSE 0
```

COGS

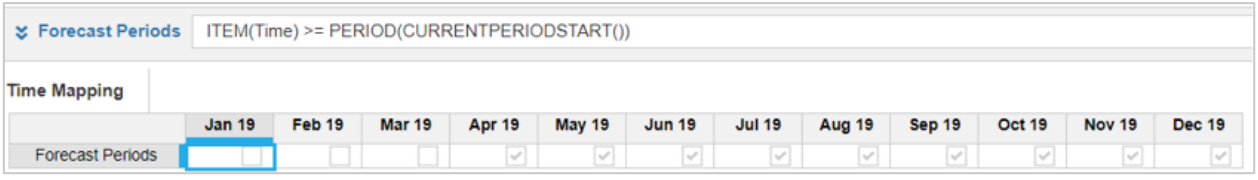
```
IF
    ITEM(Time) >= PERIOD(CURRENTPERIODSTART())
THEN
    COGS.COGS
ELSE 0
```

So what is the problem here? How else would we indicate future periods?

When dealing with Boolean logic, the answer will always be true or false (a 1 or a 0). Everything that fits between the “IF” and the “THEN” of every statement, can be a Boolean.

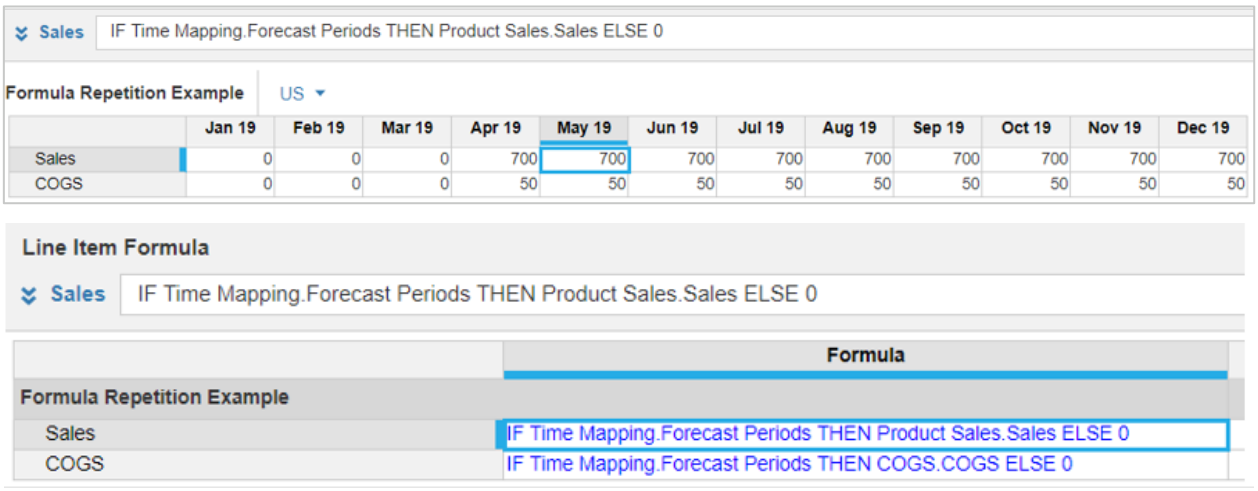
Therefore instead of having to calculate ‘ITEM(TIME) >= PERIOD(CURRENTPERIODSTART())’ for each of these line items, we can create a Boolean that performs this logic once, and reference it by our line items.

In addition, the forecast periods logic also does not need to be dimensioned by Country, as our module was. It only needs to be dimensioned by time. This is a create example of the use of a time mapping or filters modules - that can contain all of your logic that pertains directly to time. This is shown below.



	Jan 19	Feb 19	Mar 19	Apr 19	May 19	Jun 19	Jul 19	Aug 19	Sep 19	Oct 19	Nov 19	Dec 19
Forecast Periods												

Now to finish our example, we will now have each of our line items reference the forecast periods line item in the time mapping module.



Sales

```
IF
    Time Mapping.Forecast Periods
THEN
    Product.Sales
ELSE 0
```

COGS

```
IF
    Time Mapping.Forecast Periods
THEN
    COGS.COGS
ELSE 0
```

Reducing "IF" logic to a pre-calculated Boolean is highly advised. In addition to performance improvements by reduced calculation, there are also functional benefits. If for whatever reason the modeler wanted to change the forecast periods to be "> Current Period + 1", they would only now need to change the forecast periods line item in the time mapping module, instead of every line with replicating calculations.

Early Exits (https://community.anaplan.com/t5/Best-Practices/Formula-Structure-for-Performance/ta-p/33177)

Accounting for early exits while writing a formula allows your block to calculate only the necessary cells. Early exits are perhaps one of the most useful performance tricks to learn, primarily because they have no downside. A well written formula will ***always*** be better than a poorly written one.

Let's now look at an example that you may have seen before on Anapedia. We have a monthly time scale, from January to December. A single month is either a part of the summer promotion, the winter promotion, or neither. We now need to write a formula based on this logic. A module below depicts this.

Total Costs Example 1 | IF Summer Promotion THEN Summer Promotion Cost ELSE IF Winter Promotion THEN Winter Promotion Cost ELSE 0

Early Exists Module

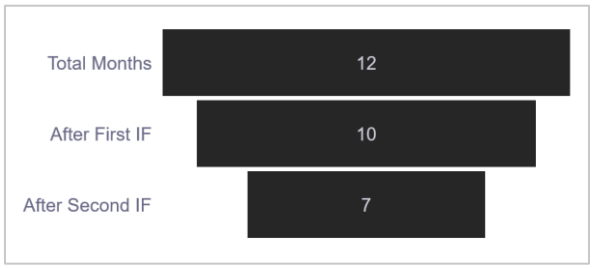
	Jan 19	Feb 19	Mar 19	Apr 19	May 19	Jun 19	Jul 19	Aug 19	Sep 19	Oct 19	Nov 19	Dec 19
Summer Promotion	☐	☐	☐	☐	☐	☑	☑	☐	☐	☐	☐	☐
Winter Promotion	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑
Summer Promotion Cost	-	-	-	-	-	$ 600	$ 200	-	-	-	-	-
Winter Promotion Cost	$ 100	$ 300	-	-	-	-	-	-	-	-	-	$ 300
Total Costs Example 1	$ 100	$ 300	-	-	-	$ 600	$ 200	-	-	-	-	$ 300

Below is the Total Costs formula written out in Sublime.

```
1  IF
2      Summer Promotion
3  THEN
4      Summer Promotion Cost
5  ELSE
6      IF
7          Winter Promotion
8      THEN
9          Winter Promotion Cost
10     ELSE 0
```

Let's look in more detail to why this is a poorly written formula. We start with 12 months that need to run through our formula. The first IF statement checks all 12 months to see if they are a part of the summer promotion, of which 2 of them are. June and July, being a part of the summer promotion have now ***exited from the rest of the formula***. We now have 10 remaining months that get testing for the second IF statement, in which 3 more exit. We now have 7 months that make it through the entire formula.

Below is a representation of the number of months that make it through each "level" of the IF statement.



The funnel chart shows that we had 58% (7/12) of our cells go through our entire calculation!

How can we fix this? The trick is to get as many cells (or blocks in this case since we are dealing with time) to leave the formula ***as early as possible,*** so they do not need to be calculated at each level.

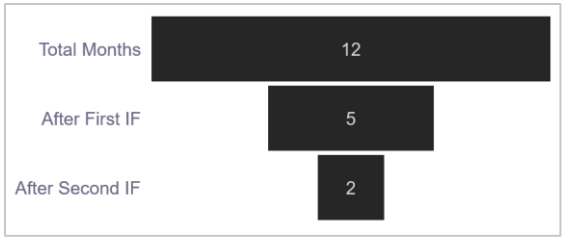
Let's look at a count of the number of months in each category to help us decide.

	Summer Promotion	Winter Promotion	No Promotion
# of Months	2	3	7

Based on this information we should start with No Promotion, followed by winter, and lastly summer. Written like this:

```
IF
    NOT Summer Promotion AND NOT Winter Promotion
THEN 0
ELSE
    IF
        Winter Promotion
    THEN
        Winter Promotion Cost
    ELSE
        Summer Promotion Cost
```

Let's look at our funnel chart now.



We see that with our updated formula, only 17% (2/12) of our cells made it through the entire formula, much better than what we had before.

The last improvement would be to take what we learned in the formula repetition section to turn our first IF statement into a Boolean.

```
IF
    No Promo
THEN 0
ELSE
    IF
        Winter Promotion
    THEN
        Winter Promotion Cost
    ELSE
        Summer Promotion Cost
```

Total Costs Example 3 | IF No Promo THEN 0 ELSE IF Winter Promotion THEN Winter Promotion Cost ELSE Summer Promotion Cost

Early Exists Module

	Jan 19	Feb 19	Mar 19	Apr 19	May 19	Jun 19	Jul 19	Aug 19	Sep 19	Oct 19	Nov 19	Dec 19
Summer Promotion	☐	☐	☐	☐	☐	☑	☑	☐	☐	☐	☐	☐
Winter Promotion	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑
No Promo	☐	☐	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐
Summer Promotion Cost	-	-	-	-	-	$ 600	$ 200	-	-	-	-	-
Winter Promotion Cost	$ 100	$ 300	-	-	-	-	-	-	-	-	-	$ 300
Total Costs Example 1	$ 100	$ 300	-	-	-	$ 600	$ 200	-	-	-	-	$ 300
Total Costs Example 2	$ 100	$ 300	-	-	-	$ 600	$ 200	-	-	-	-	$ 300
Total Costs Example 3	$ 100	$ 300	-	-	-	$ 600	$ 200	-	-	-	-	$ 300

To summarize what we learned in the example: we want to consider every cell in our block, and get it to exit the formula as fast as possible. However, you may not always know which situation occurs more frequently. This is OK most of the time, but when dealing with larger data sizes, it may be worthwhile to find out.

Early exits also tie into large nested IF THEN ELSE statements, which are best to be avoided. It is almost always better to split out your large statements into smaller ones, to take advantage of early exits.

Early Exits can also be utilized on non-number formulas as well. For example, FINDITEM() can be particularly performance intensive depending on the size and text size of the list/codes in question. If some of the text rows are blanks - there is no need to run the calculation. I would most likely expect this to happen on a transaction list where a cost center or GL account needs to be mapped.

Instead of the formula just being FINDITEM() we can add logic to exit the calculation for some cells depending on if we have more blanks or more non-blanks:

Early Exit for more non blanks

```
IF
    ISNOTBLANK(TEXT LINE ITEM)
THEN
    FINDITEM(LIST, TEXT LINE ITEM)
ELSE
    BLANK
```

Early Exit for more blanks

```
IF
    ISBLANK(TEXT LINE ITEM)
THEN
    BLANK
ELSE
    FINDITEM(LIST, TEXT LINE ITEM)
```

This might seem counter intuitive, aren't we making this block less performant by adding an IF statement? In this instance the answer is no due to saving the need to perform the FINDITEM().

Data Types

Lastly, the type of data of each block can have a significant impact on the performance and size of your model. The rule of thumb here is fairly straightforward: Booleans are good, text is bad.

It makes sense when you think about it. Booleans can only contain a true or a false, whereas a text field can be anything.

Conclusion

I hope that this article series has been helpful in describing how the Anaplan platform functions, along with relevant actionable methods for improving performance. It is also important to note that this is not an exhaustive list. My attempt here was to provide examples of how the framework presented in part I can be applied.

Regardless if you are building for performance or just diagnosing an issue - understanding the ***why*** is the best way to come up with the ideal solution.