Why is my database growing? It’s killing my calc times!

There are times when planning and forecasting databases grow for apparently no reason at all. The static data (YTD actuals) that is loaded hasn’t changed and the users say they aren’t doing anything different.

If you load budgets or forecasts to Essbase, you probably do what I’m about to tell you. If you are a systems administrator and have never seen how finance does a budget or forecast, this might be an education.

The culprit? More data!

Budgets and forecasts are not always completed at the bottom of the hierarchy and rolled up. I don’t mean technically, as you might be thinking, *Yes they do, they load to level 0 members and it gets consolidated up the outline*. When it comes to budgets and forecasts, they are largely done in a top down approach. What this means is that finance is given a goal, or number, they have to hit, and they have to PUSH it down to lower business groups. The way a financial analyst creates a top-down budget, many times, is to allocate a value based on a metric, like headcount or sales.

Assume a budget for desktop support services is required. Let’s say management has mandated that the expense doesn’t grow from last year. Since this cost is to support the people in the business, the expense is divided by the expected headcount and allocated evenly. If a business unit has 20% of the people, that unit will get 20% of the expense. Since the expense to be allocated isn’t going to change, but the headcount will, the following will be the result:

Because the analyst doesn’t want to worry about missing any changes to the headcount forecast, he or she will create a data retrieve with headcount for every cost center, whether it has headcount or not. A lock and send sheet now takes the percentage of headcount each cost center has and multiplies that factor by the total expense. As headcount gets re-forecasted, this expense has to be reallocated. With this methodology, all the user has to do is retrieve the sheet with all the headcount forecast. The math does the allocation and the result is sent back to the database.

Easy, right?

This makes a ton of sense for an accurate forecast or budget with minimal effort. Not so fast, as this has two major flaws.

First, the volume of data loaded may be drastically higher than it needs to be. Assume the worksheet has 500 cost centers (500 rows). If half of these have no headcount, there are an additional 250 blocks created that hold zeros (assuming the cost center/organization hierarchy is sparse). This method, although very efficient for updating the numbers for the analyst when headcount changes, is causing the database to grow substantially. In this isolated example, there is twice as much data than is required.

Secondly, since the data has to be loaded at level 0, the analyst thinks loading at every cost center is a requirement. The materiality of the data at this level is often irrelevant. Let’s say that the analyst is really forecasting at the region, but loading data at the cost center because it is required to be loaded at level 0. Assume there are 10 regions in which these 500 cost centers exist. A forecast at the 250 cost centers that have headcount is not required. The forecast only needs to be loaded for 10 cost centers, one for each region. If this method were used, we would only create 10 blocks, rather than the 250, or 500 originally. When the system has hundreds of users, and thousands of accounts, you can see how the size of the database would grow substantially. This also provides no additional value and huge performance problems. In the example above, the number of blocks can be reduced from 500 to 10. It is far quicker to calculate 10 blocks than 500.

Even if the data needs to be at the cost center, many times the allocation is so small, the result of the allocation is pennies, or dollars. You would be hard-pressed to find a budget where a few dollars is material. In situations like this, the users have to ask themselves if the detail is worth the performance impact.

Users, Help Yourselves

Educate your users and co-workers on the impacts of performing these types of allocations. If loading data at every cost center is required, change your formula. Rather than calculating the expense as

=headcount / total headcount * Total Expense

add an IF statement so when the retrieve has no headcount, the calculation produces a #MI,

rather than a 0. This would be more efficient

=IF(headcount=0,"#MI", headcount / total headcount * Total Expense)

If this is not necessary, change the way the data is loaded. Rather than picking all the cost centers, retrieve the headcount from the regions and build the send template to load to one cost center for each region.

The Real World

I worked for a large financial institution with a 100 Billion dollar budget. More than 70% of all the data was less than 10 dollars, and 30% was equal to zero! The budget was never looked at below region, which was 4 levels deep in an organization hierarchy that included more than 30,000 cost centers.

After consolidating the insignificant data and educating the users, the calc times decreased from 50 minutes to less than 5. All aspects of performance were better.

Easily Find Out How This is Impacting Your Application

There are a lot of ways to see if this phenomenon impacts your database. If the database is small, the export could be loaded to Excel. With some basic IF statements, the number of cells that were higher or lower than an identified threshold could be determined. Because I regularly work in a lot of different environments with large amounts of data, I wrote an application to traverse through an Essbase export to produce statistics on the data. The application is attached for download. **Make sure you have the .NET libraries installed or this will not execute**. Version 3.5 or higher is required, and can be found by searching download .net framework. There is a good chance it is already installed.

This is a simple application that I developed quickly to help me understand the degree to which a database is impacted by the example explained above. It will traverse through roughly 25,000 lines every second, and will provide the following metrics:

- the number and percentage of values above a threshold entered
- the number and percentage of values below a threshold entered
- the number and percentage of values that are 0
- the number and percentage of values that are #Missing, or Null
- The number of lines in the export and the number of seconds it took to process

To use this, export the database at *level 0* and choose *column format*. You will be prompted for the path and file name of the export, and the threshold to evaluate.

Download Essbase Export Analysis, and give it a try.