Paper 183-28

PROLAP – A Programmatic Approach to Online Analytical Processing

Jim Acker and Craig Austin, Frank Russell Company, Tacoma, WA

ABSTRACT

Traditional OLAP technologies impose rigid rules on how data must be organized. A variety of hybrid OLAP solutions have evolved in an attempt to make OLAP more flexible and adaptive to unique data problems. This paper introduces another hybrid called Programmatic Online Analytical Processing (PROLAP), which is focused on using a program to generate the data model at runtime.

INTRODUCTION

OLAP solutions assume your data is hierarchical and that you have a limited number of analysis variables. The dimensions in the hierarchies are assumed to "roll-up". For example, sales can roll-up from store to region. What do you do when your data does not so easily roll-up and/or you have large numbers of analysis variables? This paper offers a solution using SAS®.

OLAP CHALLENGES

The ability of an OLAP tool to provide drill-down functionality is supported by the concept of dimensions and hierarchies. Dimensions are data elements that are used to classify data, e.g. region or store. Hierarchies organize these classifications into different drill paths the user may want to follow when working with the data.

OLAP tools make certain assumptions about the data model. First, the assumption is that your data is already organized into a single table or a snowflake or star schema model. If it isn't, it is assumed that you can load such a schema from a denormalized data warehouse. The importance of this in the context of this paper is that you or your organization have committed significant resources to database schema design , loading and on-going maintenance. For many of us, that is not the case. We have valid OLAP requirements, but we don't have the resources to manage a schema centric approach and our source databases are more highly normalized, operational databases.

One of the important benefits of the approach we cover in this paper is that it shifts the focus from the schema to the SAS programmer, thus leveraging the skills you already have and greatly reducing the resources required to install and maintain an OLAP environment.

Here are some examples of other restrictions we encountered with traditional OLAP.

NON ADDITIVE MEASURES

One of the first problems we ran into was the need to use analysis variables that are non-additive. This means that the default aggregation functions of the OLAP tool, e.g. sum or average, are not sufficient to roll-up an analysis variable. For example, the price to earnings ratio (P/E) of an individual stock does not roll-up to the price to earning ratio for the entire portfolio.

Most OLAP solutions provide methods for dealing with this issue to a limited extent. The most common solution is a calculated measure. If you can't use common aggregation methods such as sum and mean, then you can usually break the variable in question into precedent components (such as price and earnings used to calculate the P/E ratio), then aggregate those components and calculate the new variable based on the aggregated results.

Let's take a look at this P/E Ratio example for added clarity. For simplicity sake, let's pretend that I own a portfolio of a meager 2 stocks, ABC Company and XYZ Corporation. At the End of 2001, ABC had a market cap (price) of $1M and earnings of $100K. P/E (ABC) = $1M ÷ $100K = 10. On the other hand, XYZ had a market cap of $5M and earnings of only $50K. P/E(XYZ) = $5M ÷ $50K = 100.

If I own only these two stocks, then the P/E ratio of my portfolio is the average of the P/E ratios calculated above, right? Not exactly. First, we need to consider how much of each stock that I own. So, lets say that I own 400 shares of ABC and 200 shares of XYZ. Then, the P/E ratio of my portfolio is simply the weighted average of the two P/E ratios using the number of shares as the weight, right? Not quite. The number of shares is only part of the picture. What really matters is how much those shares are valued at. (Number of shares times price per share equals market value.) So let's say that ABC was at $10 per share:
Market Value (ABC) = $10 per share * 400 Shares = $4000. XYZ was trading at $6 per share: Market Value(XYZ) = $6 per share * 200 Shares = $1200.

So, finally we can use the market values to weight the average of P/E, right? Nope, but we're getting closer! To calculate a P/E at the portfolio level, we need to use what is called a Harmonic-Mean methodology. Simply put, that means that we use the inverse of the weighted mean of the Earnings Price (E/P) ratio instead of the P/E ratio.

That's easy enough. All that we need to do is calculate the E/P by inverting the P/E before we aggregate, then invert the weighted mean after we aggregate. That sounds very easy. However, the problem is that when we use calculated measures in an OLAP application, there does not appear to be any way to control which measures are calculated before the aggregation and which measures are calculated after the aggregation.

Conceptually, we could make this work in a good OLAP tool by creating two calculated columns, one being E/P_Calculated, and the other being P/E_Calculated. EP_Calculated is nothing more than 1/PE. P/E_Calculated is then 1/EP_Calculated. The reason that we need the P/E_Calculated is because its result is based on a simple calculation (inverse) applied to the weighted mean of E/P_Calculated. It is not a simple weighted aggregate of P/E. With the OLAP tool, I would need to ensure that P/E and P/E_Calculated are not aggregated from the stock level to the portfolio level. I may, or may not be able to accomplish that with a high end OLAP tool. But, things are starting to get complicated and we haven't even begun to address the more difficult tasks of excluding outliers, rolling a portfolio up to a fund of portfolios, and calculating any of about 100 other characteristics.

We work for a large financial institution where all of the data at the stock level, the portfolio level, and the fund level is already available and used on a regular basis. Rather than trying to replicate all the formulas already coded elsewhere, all that we really need to do is extract the right data from the right tables at the right time.

THE HYBRID APPROACH

As we looked at ways to tackle non-additive measures, it was only natural to try existing solutions before plowing our own trail. The solution to our problem looked a little like ROLAP (see glossary) because the source data was already stored in a relational database. The solution looked a little like MOLAP (see glossary) because most of the data is already aggregated at all levels of the hierarchy. Yet, the solution more closely matched HOLAP (see glossary) because it

was a combination of both ROLAP and MOLAP. With any of these approaches the problem may be solvable, but at what cost?

Using hybrid extensions to existing OLAP tools, we found that we could access our pre-aggregated data at any specific level of the hierarchy, but this was only feasible if the source data were structured specifically for the HOLAP queries. As we mentioned in the opening, having data already denormalized in a data warehouse or multi-dimensional data store is a significant restriction. In reality, our data sources are structured for the applications which individually create, maintain, and exhibit that data. We do not have a data warehouse with star schemas or snowflakes which are virtually required in order to make feasible the HOLAP approach advertised by many OLAP venders. We have a vast array of existing code that has been highly optimized for each source system to extract reasonable quantities of data at once using complicated join conditions that are necessary because of the highly normalized nature of our data sources. We couldn't even begin to simplify that code enough so that we could use it with a conventional HOLAP model.

The next likely candidate for an existing solution might be MOLAP alone. With a MOLAP approach, we could pre-extract the aggregated data (at all levels of the hierarchy) and store the results it in a separate multidimensional dataset. The OLAP server would then be able to find the respective level it needed in the multidimensional datasets. While this worked, the MOLAP approach again takes a schema centric approach and requires significant human and system resources to support. To make matters worse, all of the testing that we did with this approach proved to be much slower than anticipated, even with relatively small samples of data. So, just how much data are we talking about here?

At Russell we track over 15,000 securities, over 6,000 funds, and fund portfolios for over 1,000 clients. That alone, doesn't sound too astonishing. But the numbers get really big when we start adding in some of the other dimensions, like Industry (> 200), Sector (~ 12), Countries (> 250), Regions (> 200), Time (assume 10 years of quarterly data for 40 quarters total) and then consider the exponential effect of all those dimensions.

In one example, we would like to drill from the fund level, to region to country to sector to industry to stock. Each fund (6000 of them) holds stock in an average of only 3 regions each consisting of an average of only 12 countries. Each country has stocks in most (say 10 of the 12) sectors. Each sector contains an average of 10 industries. And each industry contains an average of only 10 stocks. That's 6000 funds * 3 Regions * 12 countries * 10 sectors * 10 industries * 10 stocks * 40 quarters (for ten years of analysis) = over 8.6 Billion rows. Multiply that times 100 characteristics that we would like to be able to analyze at 8 bytes each and we're looking at over 6.9 Terabytes just to store the numeric measures in multidimensional cube that would be needed for the MOLAP approach. That excludes the identifier for each of the dimensions, so realistically we would be looking at even more space than that. This extremely large multidimensional cube was one of our prohibiting factors to using MOLAP.

That led us right back to HOLAP, which we could not use without building a large datamart specifically for this application. Unfortunately, the solution to one set of problems is based on getting data from an enormous static multidimensional dataset and the solution to the other problem is based on retrieving the data dynamically from a highly denormalized set of relational tables. What we really needed was the ability to pull the data from a highly normalized set of database tables, constrain the amount we retrieve at runtime, and aggregate the data at runtime.

DIFFERENT ANALYSIS VARIABLES

We also ran into cases where our users wanted to see different analysis variables at different levels of a hierarchy. For example, they expected stock fundamentals like average day's trading volume at the stock level but that characteristic had no value (and no defined formula for calculating it) at the portfolio level. The hierarchy seems to be there, but the underlying analysis variables are different. Virtually every OLAP tool that we looked at assumed that each characteristic was defined and had values available at every level of the hierarchy.

PROLAP TO THE RESCUE

Since we have to program the routines to assemble the aggregations in any case, why not run a program that can fetch a highly constrained amount of data, combine existing aggregated data into a multidimensional dataset, perform any additional calculations needed while we're at it, and roll up all our hierarchies at runtime? In order to do all this, we need a tool that lets us put a program in place of a data source. Instead of passing a set of parameters to a data source to be processed and returned, we would pass the parameters to a program that knows how to get the data.

Since OLAP tools do the hierarchy aggregations and we now want a **program** to perform or assemble them instead, we also need a tool that we instruct when to do aggregations and when not to do aggregations.

THE ROLE OF FUTRIX®

Futrix is an OLAP tool built with SAS. Because it is built with SAS, it already has a rich set of compute and data services we can exploit. Futrix is also designed in such a way that its core functionality can be easily modified by over-riding SCL methods exposed to the developer.

Working with Futrix, we were able to implement a data source that is a SAS program. Where Futrix passes the state of the user's session to a data source, e.g. what dimensions, analysis variables and filters they have selected, we are now passing those same parameters to our own SAS program. Now we can do almost anything.

NON ADDITIVE MEASURES

We can now perform any calculation that SAS is capable of performing, on any dataset SAS has access to, to determine the correct value we should display at any level in our hierarchy. The calculations can be performed in data steps or procedures and the order of the calculations can be concisely controlled.

After the values are calculated at each level of the hierarchy, they are stored in a simulated SAS summary dataset. The summary dataset is simulated in the sense that it was not necessarily created by PROC MEANS or PROC SUMMARY, yet it contains the _TYPE_ variable that is characteristic of a summary dataset. The _TYPE_ variable can then be used to identify the hierarchical level of each row in the summary dataset. In other words, the _TYPE_ variable identifies which dimensions are shown at the aggregate level and which dimensions are shown at the detail level for each row in the dataset. The simulated summary dataset is then used as our multidimensional cube.

Futrix can be made aware that the data source in use is a SAS summary dataset, and is capable of using the _TYPE_ variable as a filter to select the appropriate rows based on the user's current selections.

DIFFERENT ANALYSIS VARIABLES

Using SAS summary tables, we can now show whatever variables we want at any level of the hierarchy. Futrix has a unique capability of allowing us to specify required dimensions for display of any given analysis variable. In other words, we can specify that the Average Days Trading Volume analysis variable is not to be shown unless the Security Identifier or Security Name dimension variable is also shown.

With the simulated summary dataset, we can store missing values for Average Days Trading Volume at all levels of the hierarchy where security is at a summary level instead of a detail level.

CONCLUSION

PROLAP is a hybrid OLAP approach that addresses data retrieval and presentation through the replacement of the data source interface with a program interface. The OLAP tool passes the same system state parameters to the program that it passes to the data source interface. The program takes care of the data model entirely and returns a multidimensional dataset in the form of a simulated SAS Summary dataset which the OLAP tool can consume, just as if the dataset was returned from the data source interface.

This approach delegates all the complex problems of fitting data into restrictive OLAP models to the SAS programmer, where the full power of SAS can be used to better address them. The approach is very effective at dealing with the classic OLAP problems of non-additive measures and large multidimensional cubes, because it solves them outside of the OLAP model and returns a dataset that fits the model.

Futrix is an OLAP tool built on SAS. Futrix is unique in it's ability to implement the PROLAP approach.

REFERENCES

Ann Weinberger & Matthias Ender. "The Power of Hybrid OLAP in a Multidimensional World" *Proceedings of the Twenty-Fifth Annual SAS® Users Group International Conference*. March 2000. <http://www2.sas.com/proceedings/sugi25/25/dw/25p133.pdf> (December 16, 2002).

Jose Luis Ambite, Cyrus Shahabi, Rolfe R. Schmidt, and Andrew Philpot, "*Fast Approximate Evaluation of OLAP Queries for Integrated Statistical Data*" <http://www.isi.edu/~ambite/ambite-dgo2001.pdf> (December 16, 2002).

Luca Cabibbo and Riccardo Torlone, "*Design and Development of a Logical OLAP System*" <http://cabibbo.dia.uniroma3.it/pub/T4-R09.html> (December 16, 2002)

ACKNOWLEDGMENTS

We would like to thank Futrix and Qualex Consulting Services for their support in developing the PROLAP approach.

TRADEMARKS

CONTACT INFORMATION

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

Jim Acker, Craig Austin
Frank Russell Company
P.O. Box 1616
Tacoma, WA 98401-1616
Work Phone: 253-572-9500
Fax: 253-502-4299
Email: jimacker@russell.com / caustin@russell.com
Web: http://www.russell.com

GLOSSARY

OLAP – Online Analytical Processing is a category of software tools that allows high performance analysis of data. OLAP tools typically aggregate the data at many different levels allowing the user to drill from a highly summarized level of data and eventually down to the detail level while exposing many different levels of aggregation along the way.

Many OLAP tools can plug right in to an existing Data Warehouse with little or no additional programming required to expose the data contained in the Data Warehouse and perform all the necessary calculations to aggregate the data. However, a basic OLAP approach often assumes that the detail level of data is completely denormalized into a single table. With large amounts of data, the basic OLAP approach can often be unreasonably slow.

ROLAP – Relational OLAP is an OLAP extension where the detail level of data is stored in a relational database, usually in a star schema or snow flake model that is common to a data warehouse. This approach has the advantage of being more scaleable than a basic OLAP approach, but requires more maintenance and administration.

MOLAP – Multidimensional OLAP is an OLAP extension where the data is aggregated before runtime into all levels required for the hierarchy. The pre aggregated data is stored in a Multidimensional Database (MDDB). Since the data is already aggregated, this approach can provide great improvements to response time. However, the amount of space required to store the MDDB grows exponentially as more dimensions are added.

HOLAP – Hybrid OLAP is a combination of both ROLAP and MOLAP. The advantages of each are combined into a single implementation. The MDDB can be divided into several relational databases allowing different levels of the hierarchy to be stored in different places (stacking), or even allowing different parts of one level of aggregation to be stored in different places (racking).

POLAP - Progressive OLAP, though not used in the discussion of this paper, we thought would be worth mentioning. Before coining the term PROLAP, we decided run a quick search to find out if anyone was already using such a term. In fact POLAP was our first choice for an acronym. However, the POLAP acronym has already been used for Progressive OLAP, a method in which statistical models are used to estimate the aggregate values from a sample of data rather than aggregating from every record of detail.

LOLAP – Logical OLAP is the closest thing that we could find to PROLAP. Logical OLAP is a method in which a logical layer is inserted between the OLAP tool and the Data Warehouse. This layer is intended to provide independence between the OLAP tools and the data warehouse. If the logical layer is interpreted as a program or series of programs, then LOLAP and PROLAP may indeed be the same thing. On the other hand, if LOLAP is interpreted as a model under which source data relationships are defined and related to OLAP tools, then these approaches appear to have common traits, yet be different enough to deserve distinct terminology.