Analysis and Reporting for Business Intelligence Solutions Built on Microsoft SQL Server

By Eric Johnson and Joshua Jones

CA ERwin

TABLE OF CONTENTS

INTRODUCTION

Whether you are using a relational data store as your data warehouse, or hosting your warehouse directly in Microsoft SQL Server Analysis Services (SSAS), you primary concern is delivering the data to the business; this is the final step in the Business Intelligence solution. When using SSAS, you can use it to perform a great deal of work on the data in order to prepare it for delivery. This includes calculating trends such as sums for sales data, averages for orders over time, and figuring out what regions of the country sell certain products better than others. This is where the real power in SSAS comes in; helping you to extract information from your data.

Designing a methodology for how your data is calculated, stored, and delivered at this level is just as important as the basics of database and data warehouse design. While this phase is a natural extension of the data warehousing concept, it should be developed as a separate piece of the system. This is because you'll need to create and maintain documentation on how this process works (and why), and you need to be able to come back to this process and scale it as the business grows.

SQL SERVER ANALYSIS SERVICES

SSAS has grown over the past few years to become one of the most robust data analysis platforms available for the enterprise. Building on its reputation for being very easy to start new projects in, the releases of SQL Server 2005 and SQL Server 2008 have shown incredible growth in functionality and scalability of the platform. To this end, many developers are choosing to base their BI solutions in SSAS because they can quickly develop and deploy their warehouse solutions.

A large portion of this functionality comes from the SSAS business semantic model, called by Microsoft the Unified Dimensional Model (UDM), which defines business entities, logic, calculations, and metrics. Behind the scenes of any SSAS implementation, there are either a significant number of ETL processes loading data directly into the SSAS cube from various other sources, or there is the relational data store that acts as the "back-end" for the cube. The UDM helps keep the data centered, and provides a single source for all of the data related to the BI solution. This helps developers create a sort of abstraction layer, hiding away those back end processes from the front end delivery mechanisms.

OVERALL DESIGN

When it comes to actually designing and building a data warehouse, there are two approaches generally accepted in the industry; the Kimball approach and the Inmon approach. Understanding these two different methods will help you shape your projects in the future.

In short Ralph Kimball, in his work entitled *The Data Warehouse Lifecycle Toolkit*, identifies and defines the problem of the "stovepipe". In many enterprises, it often occurs that independent systems, or data marts, identify and store data in their own unique ways, much like a collection of stovepipes. Getting data from these different systems and combining it into a decision support system can be extremely difficult. To help alleviate this, Kimball advocates the conformed dimension methodology. This states that all dimensions of interest, i.e. sales data, should have the same attributes and aggregates in every data mart across the enterprise. This way, a data warehouse can be built directly from the data marts throughout the business. The primary idea here is that the warehouse contains all of the dimensional databases for ease of analysis, and users can simply query the warehouse directly for all possible information needs.

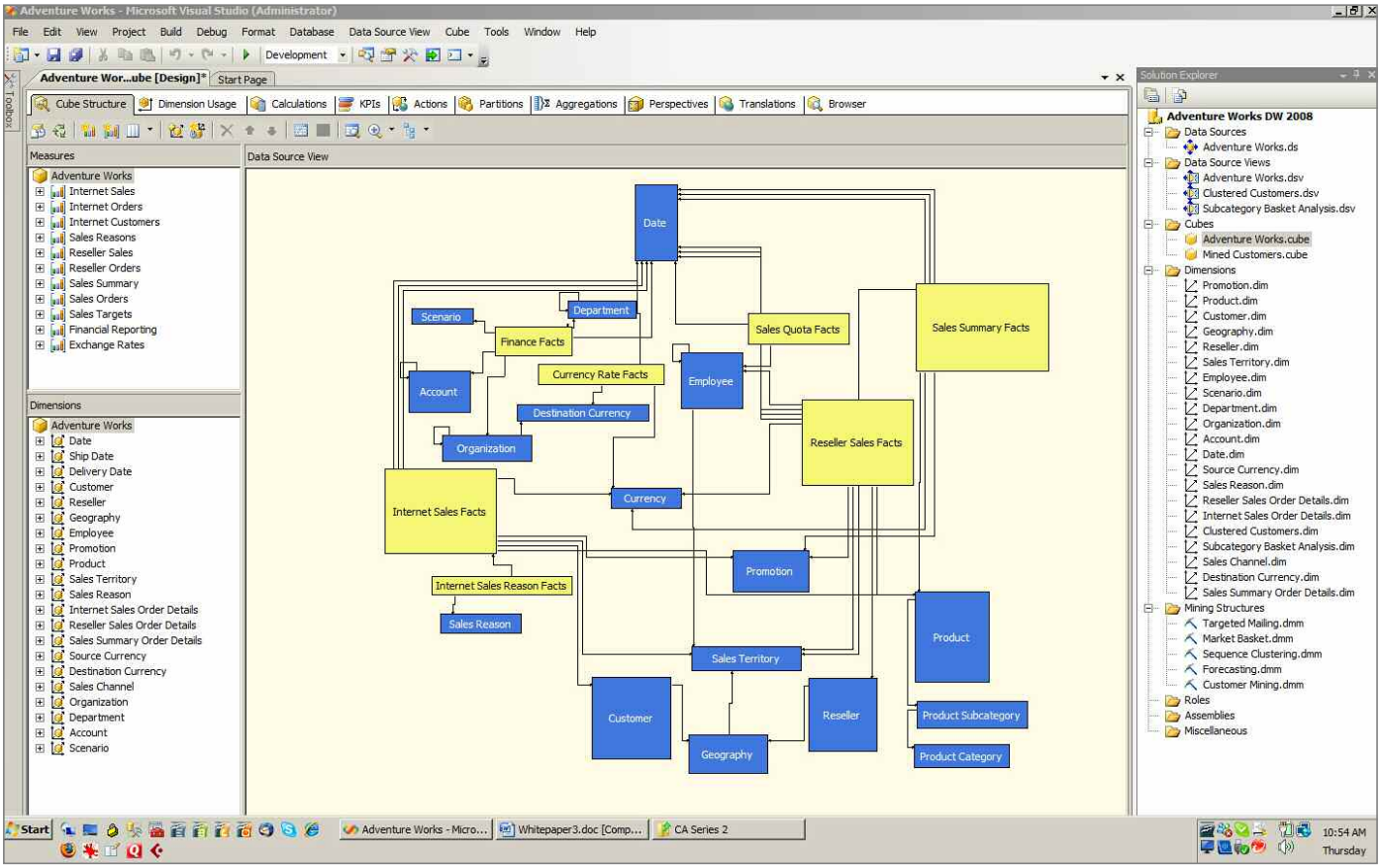


Figure 1. *A sample cube viewed from within Microsoft Visual Studio.*

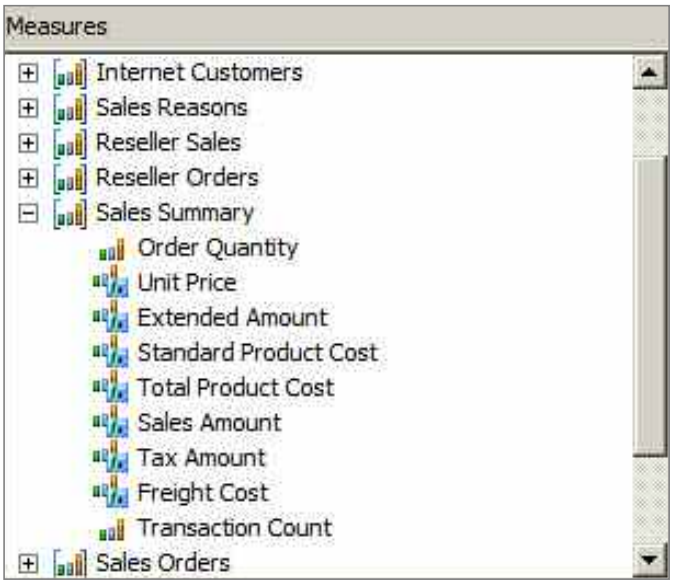


Figure 2. *The "Sales Summary" measure group in Visual Studio.*

Bill Inmon, in his work *Corporate Information Factory*, advocates are more normalized, non-dimensional format. This means that data marts are likely to contain completely disparate information, and the users query them directly, instead of a data warehouse.

In general, it's not necessary to "pick a side"; awareness of the two different approaches will help you build every project according to user needs and the different environmental variables in each project.

Once you have an idea of how your data mart/data warehouse will be built at the high level, you'll need to dig into the actual design of your project. For our purposes, we will use a sample project provided with SQL Server 2008, the AdventureWorksDW data warehouse. We will specifically be looking at the sample Visual Studio project provided for the data warehouse. This allows you to follow along at home, while providing a common area for exploration of the project outside of this paper.

For those unfamiliar with the ubiquitous Microsoft SQL Server sample company, Adventure Works is a fictional bicycle sales company. The relational database holds sales and employee information, and the accompanying Analysis Services cube display that data either directly from the relational database or via the data warehouse. All of these samples can be downloaded from **www.codeplex.com**.

CUBES

Earlier in this series, we provided a high level overview of cubes. However, it's important to understand the fundamentals of cubes and their various structures. Once you understand what a cube is made of, understanding how to use it becomes much more intuitive.

The cube is the foundation of a multidimensional database. Cubes contain typically 2 or more dimensions as well as fact data. Dimensions are what we use to describe our factual data. Common dimensions are time, geography, and product details. We apply these dimensions to our fact data, such as sales, to qualify what that data means. For example, we apply a time dimension to sales data to see how many sales a particular employee made for each month of a year. We may change the time dimension to look at that same sales data quarterly, or maybe even to view year over year comparisons. These dimensions relate to the basic concept of dimensions in any data warehouse (not only Analysis Services). However, they are not identical. In SSAS, dimensions have *hierarchies*. Each dimension has attributes that relate to one another in a parent-child style relationship; in other words, a hierarchy. A classic example of a hierarchy is a Geography dimension that has Country-State-City-Zip Code attributes. These attributes form a hierarchical relationship to one another.

If you are following along, Figure 1 shows the entirety of the Adventure Works cube opened in Visual Studio. Here you can see the facts (yellow) and dimensions (blue) in their relational

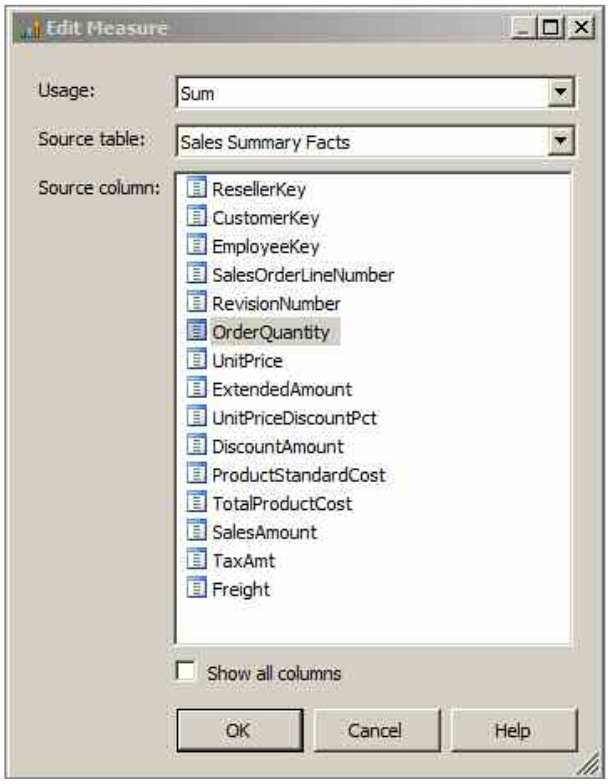


Figure 3. *The "Edit Measure" dialog box in Visual Studio.*

format in the center pane. On the right you can see the Solution Explorer, and on the left, you can see two more explorer panes: Dimensions and Measures.

In addition to dimensions and facts, Analysis Services cubes contain an object known as *measures*. Measures are basically a special dimension of the cube which are quantitative entities used for analysis. Measures are usually part of *measure groups*, which are used to help certain navigational and design tools to have better readability inside the cube.

Think of measures as collections of data columns, which may or may not have a custom expression (calculation) applied. For example, when browsing

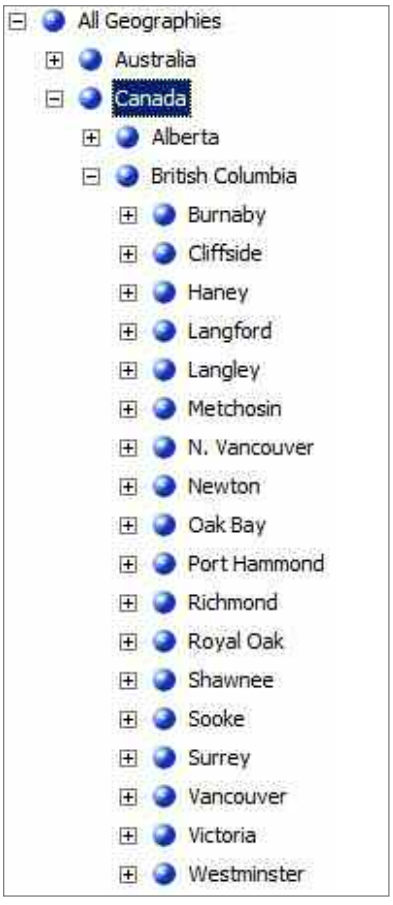


Figure 4. *The Geography dimension of the cube.*

the AdventureWorks project in Visual Studio, there is a "Sales Summary" measure group (shown in Figure 2) in the Adventure Works cube.

You can see each of the measures in the measure group. If you double-click the first measure in the list, "Order Quantity", you'll get the "Edit Measure" dialog box, shown in Figure 3.

In this case, we can see that the OrderQuantity column from the Sales Summary Facts table is being summed (in the Usage drop down). This means that this measure is simply a sum of all OrderQuantity values, or, logically, the sum of all orders.

The final two structures you need to be aware of in a cube are members and cells. Each hierarchy of a dimension contains one or more occurrences of that value in the underlying dimension table. For example, Figure 4 shows the Geography dimension, and the members "Australia" and "Canada", with the members if each lower level hierarchy.

Cells are the entities from which you retrieve data. Similar to cells in a spreadsheet (though more complex in construction), they represent the intersection of the axes of data. Unlike a spreadsheet whose values are often the intersection of two axes, X and Y, a cell in a cube is the intersection of three axes, X, Y, and Z. In this case, X, Y, and Z, are the intersections of dimensions. Furthermore, each hierarchical level in a dimension intersects with other hierarchical levels in other dimensions. This is what gives the cube its power; the presence of multiple levels of intersection between dimensions at levels in their hierarchies.

For any given project, the cube is likely to contain many dimensions (the example project contains 21 dimensions, and is a simple cube, relatively). When designing your data warehouse and cube, make sure to allow yourself the freedom to include all of the relevant dimensions to all of your facts. Don't be daunted by the number of dimensions, just take care to evaluate each potential dimension and ensure it is relevant to your users' needs.

PERFORMING CALCULATIONS

Without a doubt the most complicated portion of analyzing data is performing the numerous calculations needed to create meaningful data for the business. This includes enforcing a certain degree of business logic, along with tailoring the data to meet both historic and predictive needs. Whether or not you are using a tool like SSAS, you're going to need this essential piece of programming. Some key calculations that you may perform are:

- Averaging sales performance over time
- Finding standard deviations in statistics
- Summarizing product sales by region
- Predicting trends in financial performance

Generally, this math is going to be performed in your warehouse after your data is loaded; using whichever language is native to the warehouse. For SSAS, this would be Multidimensional Expressions (MDX); if you are basing your warehouse in the SQL Server relational engine, you'll probably use a combination of T-SQL and a CLR language (C#, VB.NET). In either case, this is the bulk of the heavy development that must happen in the warehouse.

One major component to developing the calculations in your warehouse is being able to consistently re-apply those calculations as new data is introduced. For example, you may load your warehouse from the raw data sources weekly, and perform all of the post-load calculations immediately following. However, you may occasionally find situations where you'll get older data loaded after newer data has already been used for calculations. Make sure to build your logic to be able to handle old data, and make sure it will update things like averages in derived columns without introducing invalid data.

Not only will you need to build each calculation that you need, but in the end, when the cube is built and processed, there are some cases where the order of calculations and manipulations of the cube is necessary. With Analysis Services, all of this work is done via an MDX script that executes all of the pieces of logic built in to the cube. If you are already familiar with MDX, you can look at the commands being executed and understand what is happening. However, not all of us are MDX experts.

Fortunately, using Visual Studio to develop your Analysis Services project, you have the option of using a graphical interface to help develop the calculations you need to perform, and build the overall processing script for the cube. Figure 5 shows the main interface for calculations and script tasks.

On the left hand side, you can see the Script Organizer pane (top) and the Calculation Tools pane (bottom). The bottom pane has all of the pre-built tools, such as aggregations and templates available for you to use. The top pane is literally showing you, command by command, what the processing script is going to do. In this case, we have the 6th step highlighted. This is the "Internet Gross Profit" calculation step. A close up of the center pane for this step is show in Figure 6.

Here we can see the name of the script step, as well as the hierarchical position and expression actually being used. Below we see the properties and meta-data around this step. Contrast this with Figure 7, which is the script view of this same step (achieved using the toolbar at the top of the pane).

Notice that this is actually showing you the entire script; but because we had selected the "Internet Gross Profit" step in

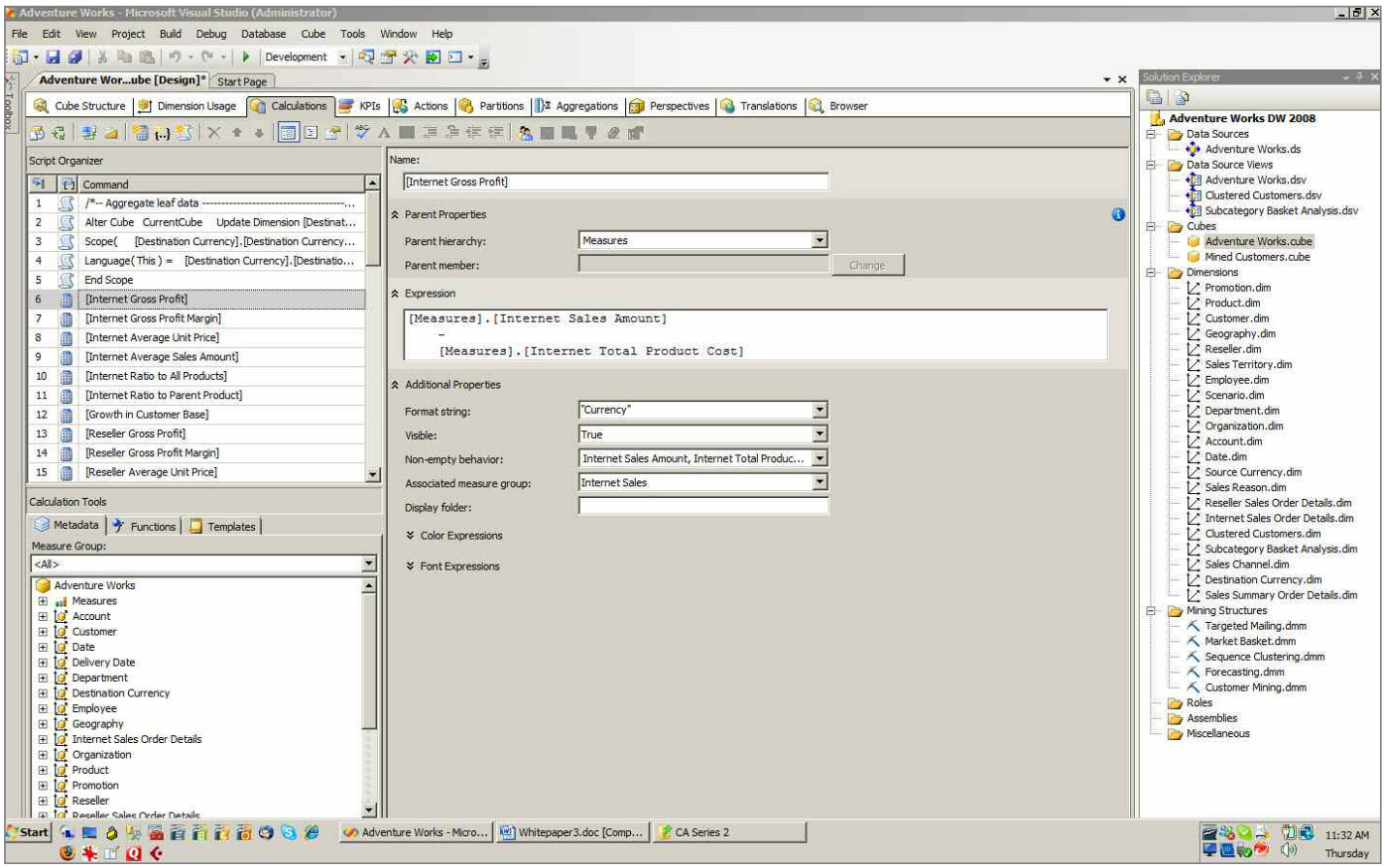


Figure 5. *Visual Studio interface for calculations and script tasks.*

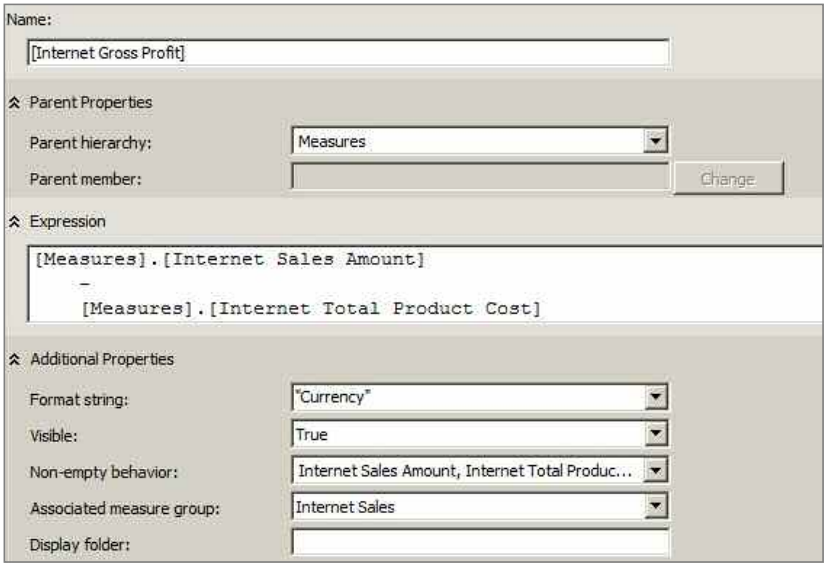


Figure 6. *Calculations for manipulating data in Visual Studio.*

the graphical view, and placed our cursor at the corresponding location in the processing script. This is showing us the actual MDX used to perform this calculation. Flipping back and forth between these views is a great way to learn MDX. As you use the graphical version, you can flip over to the script to see what you're building as you go. You can also use the graphical version to literally click and drag steps to rearrange the order of operations in the script.

By exploring the sample project, you'll find that there are a significant number of pre-built calculations. Most of the calculations that the average data warehouse uses can be found in the built-in collection; if you need to do something more

```
);

   Language( This ) =
   [Destination Currency].[Destination Currency].Properties( "Locale" );

End Scope;


/*-------------------------------------------------------------------------
 | Internet Sales Calculations |
-------------------------------------------------------------------------*/


Create Member CurrentCube.[Measures].[Internet Gross Profit]

 AS [Measures].[Internet Sales Amount]
    -
    [Measures].[Internet Total Product Cost] ,

Format_String = "Currency",
Associated_Measure_Group = 'Internet Sales',
Non_Empty_Behavior =
 {
   [Internet Sales Amount],
   [Internet Total Product Cost]
 };


Create Member CurrentCube.[Measures].[Internet Gross Profit Margin]
```

Figure 7. *The script that executes the calculation.*

complicated, you'll need to use MDX queries explicitly to create those calculations yourself. However, the nuances of MDX are outside the scope of this paper, but there are plenty of references available on the web and in your local bookstore.

BUSINESS INTELLIGENCE REPORTING

Once you've loaded and calculated all of your data, and have a scalable, resilient process that can update that data, you have to figure out how your business will access that data. There are tools available that can allow business users direct access to your cube and/or data warehouse, allowing them to create and save reusable custom queries. You may also need to "push" the data to your users in the form of a report. Generally, this method requires some sort of skilled personnel to take requests from the business users and turn it into readable reports that can be accessed via an internet browser (such as an intranet page hosted in Microsoft Office Sharepoint Services) or emailed directly to the user. There are a number of high profile reporting tools available in the marketplace today; Business Objects' Crystal Reports and Microsoft SQL Server Reporting Services just to name a few.

DEFINING REPORT NEEDS

As with any application development, designing and developing reports starts with interviewing users and getting detailed specifications about what the report should contain. You'll not only need to know what types of data are to be included, but what range of dates should be evaluated, as well as the level of detail. This tends to vary by user; managers and higher ranking employees tend to want to see aggregated data over large periods of time, such as monthly, quarterly, or year over year. First level managers tend to want to see more detailed data presented at the daily or weekly level.

You'll often find that you will design multiple reports that contain more or less the same data, but evaluated against a different time measure. Because of this, it is almost always worth your time to consider, when writing any report, whether or not that report could be re-used by another user.

DESIGNING REUSABLE REPORTS

Because the data warehouse is supposed to be the "single version of the truth" in your enterprise's data, nearly every department or type of employee is going to want to retrieve data from it. If you custom build reports for every single request, you'll quickly find yourself knee deep in reports that are 90% alike. Whenever you write a report, consider the type of data that is being retrieved and presented. Ask yourself, "Can I repurpose this report to fit another need?" Or, build reports that have both aggregated and detailed data available. Many report users will want to use a browser to access their report; this means you can build reports with drill-down capabilities. This means you can provide reports on topic areas such as "Sales by Region" or "Annual Sales", and build sub-reports into those reports that allow drill downs into more detailed data (by month, by state, etc.). Doing so will allow you to satisfy both the high level user looking for aggregated, trend oriented data as well as the low level, operational oriented user. In the end, this can save you a lot of development time.

Additionally, when you combine tools like SQL Server Reporting Services (SSRS) with Analysis Services, you allow yourself the luxury of taking advantage of features designed to support more efficient development. For example, SSRS reports natively support SSAS as a data source. And because of this integration, SSRS understands how to quickly navigate the cube in SSAS< meaning that you can build a drill down report from summary to detailed data in minutes, without having to write a completely new set of queries for the data. This can help minimize development time, and give you extreme flexibility in report design.

DELIVERY METHODS

Once you've gotten a report (or dozens of reports) written, you will have to deploy those reports. The current industry tendency is to provide access to reports via a web interface, hosted on an intranet or secure extranet site. These kinds of interactive reports are extremely versatile, and help designers provide useful reports in a single place. However, some users need static reports delivered to them, often via email. Fortunately, most modern reporting tools allow an email export of a static report. SSRS even goes so far as to offer subscriptions, which allow users to specify how often they receive a report, and whether or not they want a fresh version of the report rendered and set.

Conversely, administrators can specify that certain, highly utilized reports be run in the morning and cached throughout the day, so as to reduce overhead on the reporting system and provide a consistent report to all users. SSRS even offers a data-driven subscription, which allows designers to build and deliver reports based on conditional data in a database. This means you can have an endless number of criteria determine when and how a report gets generated. This way, you can customize the reporting system to meet your specific environment.

AD-HOC DATA ACCESS

Besides actual reports, many users, particularly business analysts, will actually need to access the data warehouse and/or cube in an interactive way. They need to be able to look at different views of the data in order to start identifying trends, and predicting future business performance. This type of ad-hoc access can be provided via in-house developed applications, or via third-party applications such as ProClarity or Business Objects.

Typically, these types of applications are graphical interfaces for query engines that submit MDX queries to the warehouse and present the results to the user in an east to read format. This usually includes drill down capabilities, so as to be able to present aggregated data first, and exploratory detailed data as the user requests it (by click through). The benefit of using these applications is not only ease of use for the user, but because it's also more secure. You can restrict access to more sensitive information based on the users login to the query tool, versus trying to control that access at the SSAS or even SQL Server level.

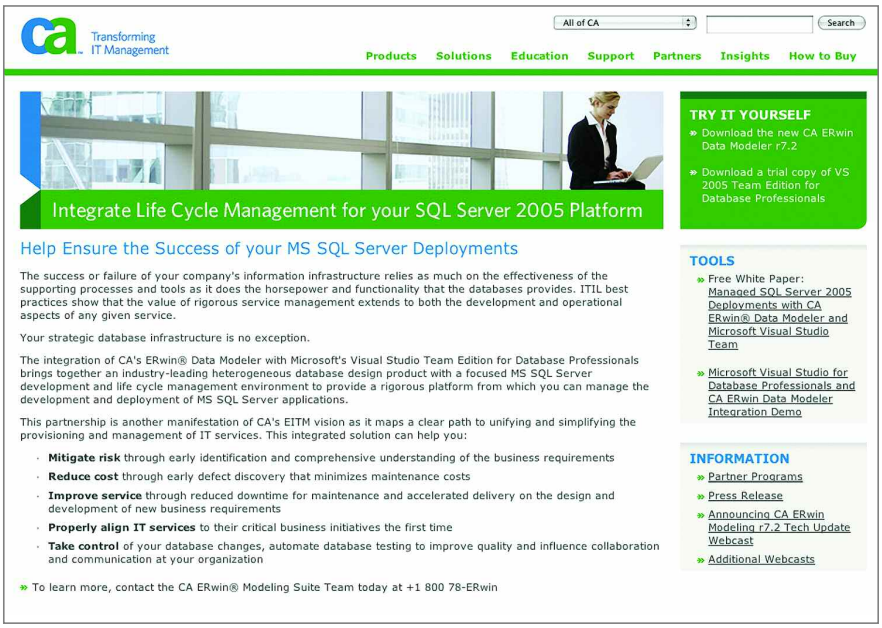
Microsoft provides some built-in functionality to accommodate this type of access; the Report Model. Report models are built by using Visual Studio (using the Report Model project type), the Report Manager (Reporting Services), or even Microsoft Office Sharepoint Server 2007 (MOSS). These models can be based on SQL Server databases, SSAS 2005 or later cubes, or Oracle databases running version 9.2.0.3 or later. This model is basically a meta-data version of your data. The front end tool, whether it's Reporting Services or some other product, will have a "friendly name" version of all of your databases, columns, dimensions, etc. This makes ad-hoc access much easier, as well as report design. A wizard is provided to build models, and allows you to customize which facts and dimensions are present in a given model. This means you can create models as small or large as fits the end user requirements. Often, model could be built to fit departmental needs for most users, with one or two large, all encompassing models for higher level managers and analysts.

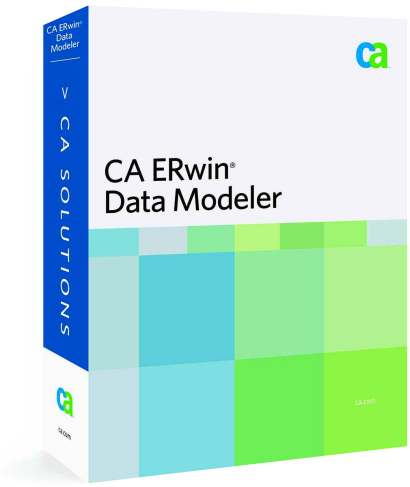
CONCLUSION

Modeling and building a data warehouse is a formidable project to undertake. When in the design and development phases, it tends to use a lot of resources as well as take considerable time. The payoff for the business comes once the historical data can be put to good use through analytics and delivery. However, considerable attention must be paid to this final phase of a BI solution.

There is much more to SQL Server Analysis services, such as custom data mining and advanced cube design. To get started, download the sample applications and projects from the web, and start learning to use this powerful tool.

Josh Jones and Eric Johnson are the Operating Systems and Database Systems Consultants, respectively, for Consortio Services, as well as co-hosts for the popular technology podcast "CS Techcast" (www.cstechcast.com). They offer consulting and training services in many areas of Information Technology. Check our www.consortioservices.com for more information. Eric and Josh have written a combined three books including the forthcoming "A Developers Guide to Data Modeling for SQL Server".





Visit: **ca.com/erwin/msdn**.
CA ERwin Data Modeler provides a way to create a logical structure of your data and map that structure to a physical data store.

ca.com/erwin/msdn