Chapter 1

Introducing Microsoft Analysis Services 2005

Albert Einstein once said that information alone is not knowledge. This adage has never been more relevant than in today's information age where organizations are looking for ways to quickly make sense of mountains of data generated every day. Only by carefully screening and analyzing that data can an organization unlock its power and fully understand its customers, its markets, and its business. I am not aware of an official slogan for Analysis Services but the one that may best serve this purpose could be "Knowledge is power".

As its name suggests, the promise of Microsoft SQL Server Analysis Services 2005 is to promote better data analytics by giving information workers powerful ways to analyze consistent, timely, and reliable data. Empowered with Analysis Services, you are well positioned to solve the perennial problem with data – that there is too much of it and finding the right information is often difficult, if not impossible.

This introductory chapter gives you a panoramic view of Microsoft SQL Server Analysis Services 2005 (SSAS) and the Microsoft Business Intelligence platform. Throughout the rest of this book, I will use the terms Analysis Services and SSAS interchangeably to refer to Microsoft SQL Server Analysis Services 2005. In this chapter, we will discuss:

- What is SSAS?
- The services SSAS provides
- The role of SSAS in the Microsoft Business Intelligence Platform
- How SSAS unifies the relational and dimensional reporting models
- SSAS architecture
- SSAS in action hands-on lab

1.1 What is Analysis Services 2005?

The processes of collecting and analyzing information assets to derive knowledge from data are typically referred to as *Business Intelligence,* or *BI,* for short. Simply put, SSAS can be viewed as a sophisticated software platform for delivering business intelligence by providing rich and efficient ways to "get out" what was "put in". To be more specific, we can describe SSAS as a server-based platform that provides two core services -- On-Line Analytical Processing (OLAP) and data mining. Let's cover these two terms in more detail.

Definition Microsoft Analysis Services is a server-based platform for on-line analytical processing (OLAP) and data mining.

1.1.1 Introducing OLAP

There are two types of database-driven systems that serve orthogonal requirements. *On-Line Transactional Processing* (OLTP) systems are designed for fast transactional input to support business systems. On the other side, OLAP systems are optimized for fast data output to support data analytics and reporting.

Definition On-Line Analytical Processing (OLAP) applications are optimized for fast data querying and reporting.

Let's consider a popular BI scenario that can be implemented using the OLAP technology and SSAS.

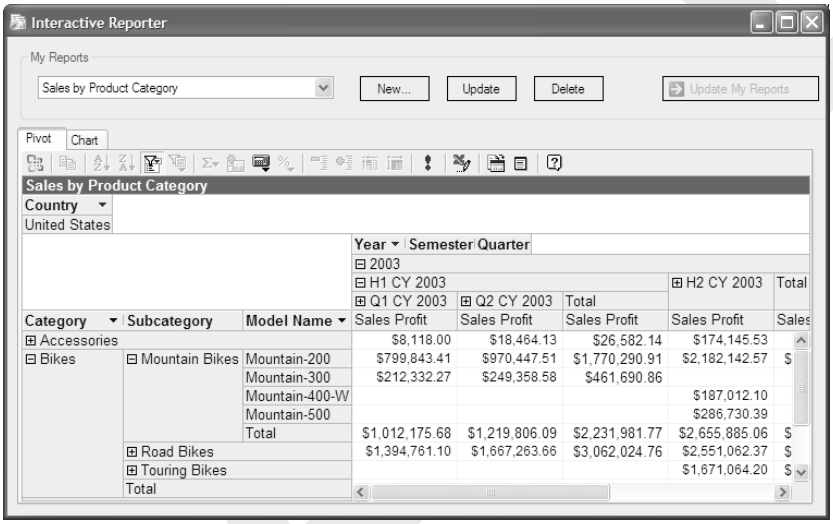


Figure 1.1 Use "smart" OLAP browsers connected to SSAS to build interactive reporting applications.

Fast and intuitive reporting

Suppose that your company's CEO is asking you for a report showing last-year sales. You run an SQL query that produces the magic number, e.g. the company made one million dollars in sales. While this figure may be great (or not so great depending on the company situation), it is unlikely to provide enough information to your CEO and marketing department.

There are a myriad of questions your business users may ask you. How do the sales of this year compare to last year? What are the top selling products? How do products sale by region, resale channels, etc? You may opt to address some of these questions by authoring standard reports but, at some point, this process may become counterproductive, especially when large datasets need to be processed.

One elegant and efficient solution is to implement an OLAP application that allows business users to produce reports interactively, as shown in Figure 1.1. It is implemented as a .NET Windows Form client application that leverages a "smart" OLAP browser connected to SSAS

2005. In this case, the browser is Microsoft Office Web Components (OWC) which is part of the Microsoft Office suite. Empowered with this application, end users could view data from different angles (called *dimensions* in the OLAP terminology) by creating dynamic views interactively. Moreover, report queries will be satisfied almost instantaneously because OLAP servers are optimized for fast retriaval.

Definition The OLAP terminology uses the term *dimension* to refer to an analytical perspective that can be used to browse the data. Common examples of dimensions include Product, Customer, and Time.

In Figure 1.1, the user has decided to see sales data broken by *Product by Category* dimension on rows and *Time* dimension on columns. The user has expanded ("drill down") the Product by Category dimension to see data broken further down by Product Category, Subcategory, and Model Name. The Time dimension is also expanded to show sales figures by Calendar Year, Semester, and Quarter levels. Data is further filtered by Country to show sales figures for United States only.

I dubbed this type of reporting "interactive" because with a few mouse clicks the user can change the report to view data from different angles. For example, assuming the SSAS model supports this, the user can opt to view the sales data by Customers on rows and Fiscal Year on columns. Optionally, this application allows the user to save the report view to a database and retrieve it on as-needed basis. I will show you how you can build this type of applications in chapter 19.

As you could see, interactive reporting is much more powerful and flexible than standard "canned" reporting. The tradeoff is that you need to spend some extra effort to design and implement your OLAP database in such a way that it conforms to the OLAP dimensional model, as we will discuss in section 1.2. Don't despair, though. SSAS definitely goes a long way in making this endeavor easier.

SSAS as OLAP server

Can you produce interactive reports from a relational database instead of going through all the trouble to implement an OLAP solution? Most likely the answer will be yes. For example, you can connect OWC directly to a relational database and achieve similar reporting results. There is really no magic that OLAP performs behind the scenes as a database engine. The same aggregation results could be achieved by just sending SQL statements. So, why should you use OLAP and SSAS for that matter?

To answer this question, consider the **F**ast **A**nalysis of **S**hared **M**ultidimensional **I**nformation (FASMI). This test was proposed by The OLAP Report, an independent organization that studies the OLAP technology and market (see the Resources section). It outlines five easy to remember criteria that each OLAP server should adhere to.

The *Fast* rule means that the OLAP server has to be optimized for fast (almost instantaneous) data retrieval. The OLAP Report suggests that most queries should be answered in five seconds. Of course, performance depends on a variety of factors, including hardware and software configuration, level of optimization, etc., but OLAP servers have to be *FAST*.

The *Analysis* rule means that the OLAP calculation engines should be superior in supporting advanced business calculations compared with RDBMS and these calculations shouldn't mandate the use of a professional programming language.

Each OLAP server should also provide *Shared* access to data. This criterion has two aspects. First, it mandates that every OLAP server should protect sensitive data preferably at the most

granular (cell) level. In addition, an OLAP server should support *writeback*, i.e. allowing the users not only to read, but also to change data.

The *Multidimensional* characteristic is the most important requirement. Every OLAP system should provide a multidimensional view of data that extends beyond the two-dimensional analysis of RDBMS. OLAP users should be able to see data from different angles called *dimensions*, as we've just seen in the example shown in Figure 1.1.

Finally, the *Information* criterion measures the ability of the OLAP system to store and aggregate vast volumes of data without performance degradation. So, to sum up, the strongest motivation factors to favor the OLAP technology instead of RDBMS for reporting is its superior performance, user-friendly reporting model, and rich calculations. I hope that in this chapter and throughout this book, I will convince you to consider OLAP for implementing efficient and rich reporting solutions.

1.1.2 Introducing Data Mining

The second main service that SSAS provides is data mining. Data mining is a science in itself. Generally speaking, data mining is concerned with the process used to predict the unknown based on known statistical facts. Instead of asking us to look at a crystal ball, the SSAS team has implemented sophisticated mathematical models that can analyze large volumes of data, discover patterns and trends, and produce prediction results.

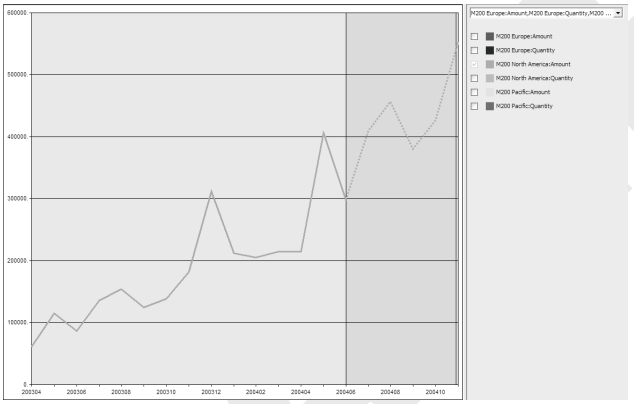


Figure 1.2 Use SSAS data mining to discover trends in data, such as forecasting future sales.

Typical examples where data mining can be used efficiently include sales forecasting and basket analysis. For example, by examining the historical sale figures, data mining can answer the following questions:

- What are the forecasted sales numbers for the next few months?
- What products may this customer buy together with the chosen product?
- What type of customers (gender, age groups, income, etc.) is likely to buy this product?

A common practical example of using data mining is shown in Figure 1.2. Imagine that your company is selling products and the marketing department is asking you to estimate the sales in North America for the next five months. Instead of dusting off your college math book, you prudently decide to use SSAS. Once you build the data mining model, with a few mouse clicks you can produce a report as the one shown in Figure 1.2. We will discuss data mining in more detail in chapters 7 and 8.

1.1.3 Overview of SSAS

Shortly after acquiring the Panorama OLAP technology in 1997, Microsoft introduced the first release of SSAS. It shipped as an add-on to the SQL Server 7.0 and was named Microsoft OLAP Services. The second version coincided with the SQL Server 2000 release. The product name was changed to Analysis Services 2000 to reflect the fact that now SSAS provided not only OLAP, but also data mining capabilities. Today, SSAS 2000 is a leading OLAP platform according to The OLAP Report (see the Resources section). After five years of gestation effort, Microsoft released SQL Server Analysis Services 2005 in November 2005.

SSAS editions and licensing

As its predecessors, SSAS 2005 ships as an add-on to SQL Server 2005. However, note that, although it is bundled with SQL Server, SSAS is not necessary dependent on the SQL Server relational engine.

Note There are some features in SSAS 2005, such as using multiple data sources in a single data source view and proactive cache notifications, that work only when SQL Server is used as a data source.

For step-by step instructions on how to install SSAS 2005, refer to Appendix A at the end of this book. Let's now briefly discuss how SSAS is packaged and its licensing requirements. To address different user needs, SQL Server 2005 is available in five editions – Express, Workgroup, Standard, Enterprise, and Developer editions. However, SSAS is available in the last three editions only (see Table 1.1). The Developer edition has the same feature set as the Enterprise edition but it is licensed for one machine only.

Table 1.1 SSAS supports three editions to address different user needs.

Edition	Choose when
Standard	You need to install SSAS on a single server. The Standard edition doesn't support advanced analytics and scalability features, such as partitioned cubes, proactive caching, and parallel processing.
Enterprise	You need all SSAS features and your OLAP solution must be highly scalable.
Developer	You design and develop SSAS databases. The Developer edition supports all SSAS features but it is not licensed for production use.

For more information about how SSAS editions and other SQL Server 2005 products compare to each other, read the document *SQL Server 2005 Features Comparison* (see Resources section). SSAS 2005 licensing model is simple. Basically, you need a SQL Server license on the machine where SSAS is installed and there are no special "middleware" exceptions. For example, suppose your operational requirements call for installing SSAS on a separate server than your SQL Server RDBMS box. In this case, you will need two SQL Server licenses – one for the SSAS server and another one for the SQL Server instance.

Why use SSAS?

Traditionally, Analysis Services and OLAP in general have been used in conjunction with data warehousing. Of course, this scenario is still applicable. Many organizations would use Analysis Services to build an OLAP layer on top of a relational data warehouse in order to take advantage

of the superior query performance of SSAS. However, thanks to new enhancements in SSAS, I believe you will find new scenarios for using it, including:

- *Rich data analytics* – For many organizations, SSAS can become the logical next step for advanced data analysis and interactive reporting.
- *Data mining* – An organization could find many uses for the predictive power of data mining.
- *Corporate performance management* – With the introduction of KPIs, SSAS can be used to capture vital company performance metrics. More on this in chapter 12 and 19.
- *Centralized repository for business metrics* – SSAS supports advanced calculations and is best suited for storing business metrics and calculations.
- *Ad hoc reporting* – Besides interactive reports, end-users can create ad hoc reports from SSAS. We will see how this could be done in chapter 18.

At the same time, as there is no such a thing as a free lunch, SSAS may be overkill for small organizations because it requires an additional design and maintenance effort. As a rule of thumb, if standard reporting meets your data analytics and performance needs, it may not be time to "graduate" to OLAP yet.

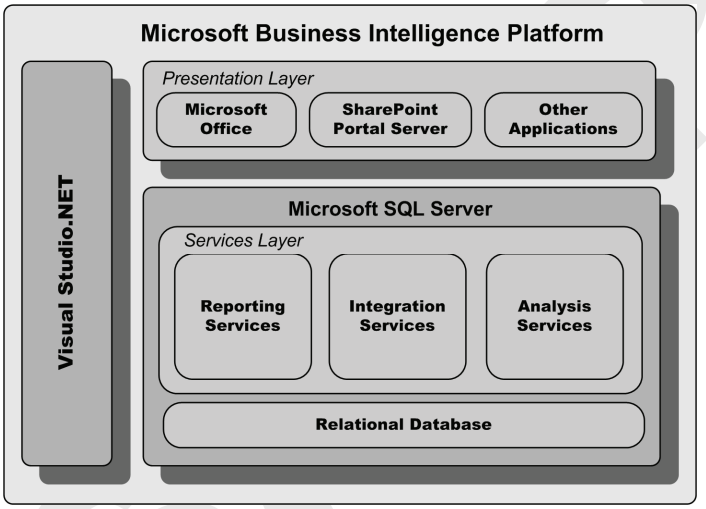


Figure 1.3 The Microsoft Business Intelligence Platform provides valuable services and tools that address various data analytics and management needs.

1.1.4 SSAS and Microsoft Business Intelligence Platform

SSAS is not the only Business Intelligence product that Microsoft provides. It is an integral part of the Microsoft Business Intelligence Platform that was initiated in early 2004 with the powerful promise to "bring BI to the masses". The Microsoft Business Intelligence Platform is a multi-product offering that addresses the most pressing data analytics and management needs that many organizations encounter every day.

To understand how SSAS fits into this initiative, it may be helpful to depict the Microsoft Business Intelligence Platform in the context of a typical three-tier architectural view that most of the readers are probably familiar with, as shown in Figure 1.3. Let's explain briefly the building blocks of the Microsoft Business Intelligence Platform.

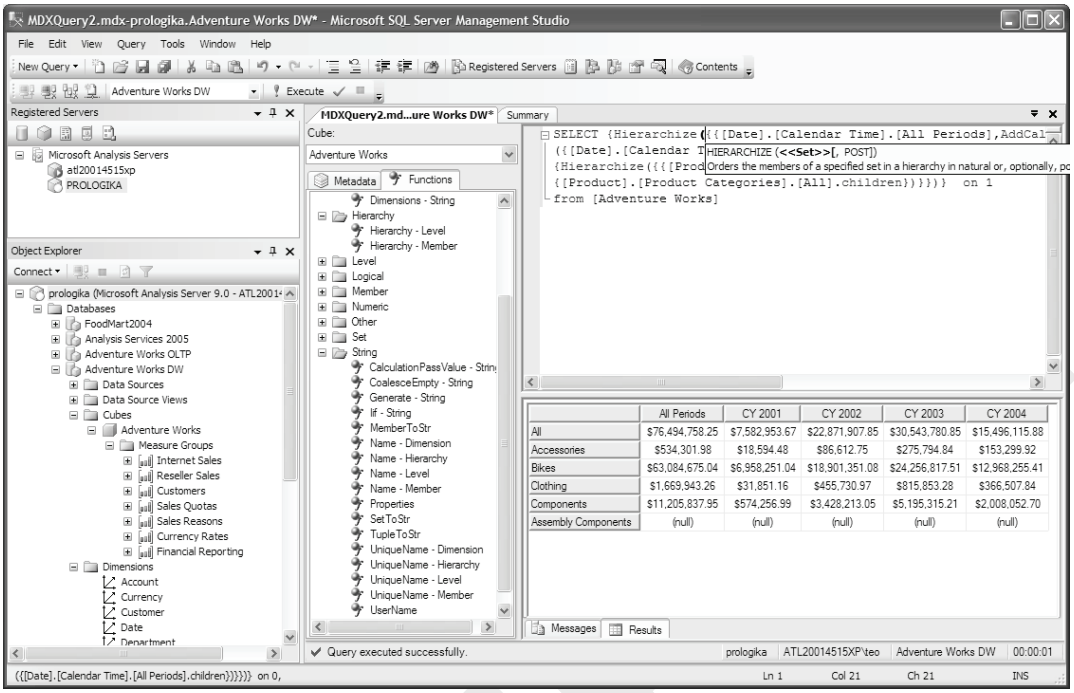


Figure 1.4 Use SQL Server Management Studio to manage all Analysis Services, Reporting Services, Integration Services, and SQL Server installations.

SQL Server

The SQL Server relational database engine forms the foundation of the BI Platform. In my opinion, SQL Server is one of the best products that Microsoft has ever invented. Its relational database has been holding the top TPC (Transaction Processing Council) benchmarks in the price/performance category, as you could see online at *www.tpc.org*. Now that SQL Server comes bundled with so many valuable add-on services, it is indeed "do more with less", as the popular Microsoft slogan goes. While discussing the SQL Server 2005 enhancements and new features may easily fill a whole book, I would like to bring your attention to a couple of SSAS-related enhancements that we will be using throughout this book

SQL Server Management Studio

SQL Server 2000 Enterprise Manager is gone and it is replaced by the new SQL Server Management Studio (see Figure 1.4). The most prominent feature of the SQL Server Management Studio is that it can be used to manage all SQL Server services. Figure 1.4 shows that I've connected to an Analysis Services server called *Prologika* and I've executed an MDX query against the Adventure Works cube.

The central pane shows the metadata of the Adventure Works cube. You can drag and drop objects from the Metadata tab, or MDX standard functions, from the Functions tab. Yes, the query editors support IntelliSense so you could check the function syntax easily! SQL Server Management Studio comes with a slew of editors, templates, designers, and other tools to meet the full spectrum of your query authoring, performance optimization and management needs.

SQL Profiler

Veteran SSAS developers know that, in the past, it was almost impossible to "peek under the hood" of the SSAS server. This has all changed now since the SQL Profiler has been enhanced to support capturing and displaying events raised by SSAS 2005. For example, you can use the SQL Profiler to intercept an MDX query to see how long it takes to execute. We will meet the SQL Profiler in chapter 13 when discussing SSAS management.

Services layer

On top of the relational database, SQL Server provides various services. The three main BI pillars are Reporting Services (SSRS), Integration Services (SSIS), and, of course, Analysis Services (SSAS) which is the subject of this book.

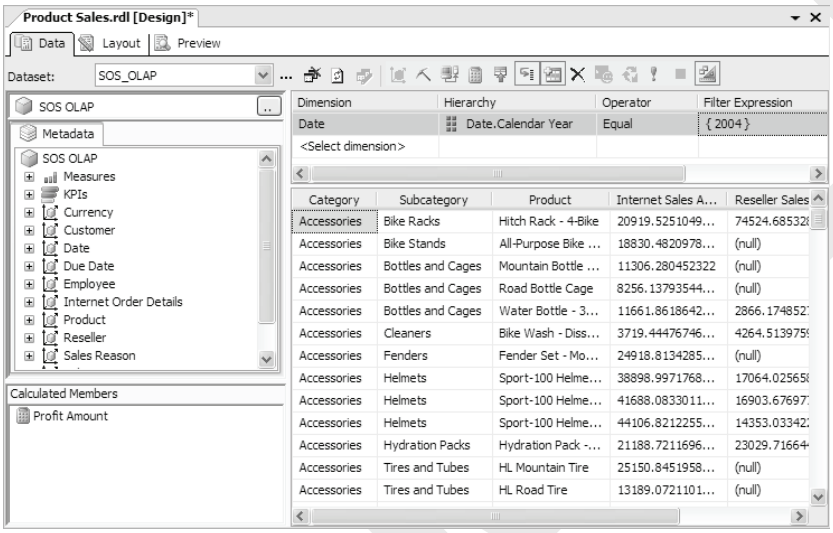


Figure 1.5 The MDX Query Builder makes authoring reports from SSAS a breeze

Reporting Services

SSRS is a server-based platform for authoring, managing, and distributing standard reports. SSRS reports can source data from virtually any data source that exposes its data in a tabular format, including SSAS. A new SSAS-related feature of SSRS 2005 is the excellent MDX Query Builder that you can use to create reports from SSAS cubes easily (see Figure 1.5). Figure 1.5 shows that I've authored a Product Sales report by dragging Product dimension and a few measures from the Metadata pane and dropping them onto the Results pane. In addition, I've parameterized this report by allowing the user to filter the report data for a given calendar year.

I've covered SSRS 2000 in details in my book *Microsoft Reporting Services in Action* (see the Resources section). In chapters 18 and 19 of this book, I will show you the most exciting new features of SSRS 2005 that relate to authoring SSAS-based reports, including the MDX and DMX query builders, the new Windows Forms and ASP.NET report viewer controls, ad-hoc reporting, and SharePoint integration.

Integration Services

Today's enterprise IT shop would typically maintain a hodge-podge of data sources and technologies. These include desktop databases, legacy mainframe systems (that no one dares to touch), RDBMS, etc.

Note One of my projects involved building a data warehouse for a call center of a major financial institution. The data integration requirements called for extracting data from six databases and consolidating it into a central data warehouse repository. Most of the project effort was spent on implementing the ETL data integration processes.

For example, the order tracking data could reside in a SQL Server database, the HR data could be stored in an Oracle database, while the manufacturing data could be located in a mainframe database. Integrating disparate and heterogeneous data sources presents a major challenge for many organizations. This is where SSIS (formerly known as DTS) could be useful. It is typically used for Extracting, Transforming, and Loading (ETL) processes for data integration.

SSIS has been completely revamped in SQL Server 2005. There are a few exciting features in SSIS that specifically target SSAS, including dealing with slowly changing dimensions, implementing low-latency OLAP, and processing partitions. One of the most common OLAP requirements that could benefit from SSIS is data warehousing. We will see how this could be done in chapter 6. There are other SQL Server add-on services that you may find more or less relevant to BI applications. These may include *Replication Services* to clone data, *SQL Server Broker* to raise event notifications, and *Notification Services* to build sophisticated notification application.

Presentation layer

The OLAP technology will be useless if users cannot browse the data. SSAS itself doesn't provide an OLAP browser. The BI platform delegates this role to the Microsoft Office suite, SharePoint, or third-party products.

Microsoft Office

In section 1.1.1 of this chapter, we had a glimpse of how Microsoft Office Web Components (part of the Microsoft Office suite) can be used to build "smart" OLAP clients. Besides OWC, in the last chapter of this book I will show you how to integrate Microsoft Excel and the Office Business Scorecard Manager 2005 with SSAS 2005 to implement interactive reporting and performance management.

SharePoint

Use SharePoint to build enterprise-level portal sites. SSAS doesn't include web parts to browse cube data. However, the Reporting Services team has built two SharePoint web parts, Report Explorer and Report Viewer, which can be used to integrate SSRS reports with a SharePoint portal. We will have a glimpse of how SharePoint Portal Services can be used to disseminate scorecards in chapter 19.

Other applications

Developers can utilize industry-standard connectivity protocols to integrate SSAS and SSRS easily with their applications. In chapter 17, I will show you how you can develop custom OLAP-based applications.

Visual Studio.NET

Finally, developers can use Visual Studio.NET to glue the components of the BI Platform together. Developers can use the excellent Visual Studio.NET IDE to custom solutions or work with BI projects. If you don't have the full-blown version Visual Studio.NET (or you are not willing to purchase a license), the SQL Server 2005 setup program gives you an option to install a scaled-down version of Visual Studio.NET, called *Business Intelligence Development Studio (BI Studio)*.

BI Studio supports Analysis Services, Reporting Services, and Integration Services projects. It gives you the power of the Visual Studio.NET Integrated Development Environment at no additional cost. Using BI studio, you can centralize the design and management of your BI projects. Now that we have reviewed the components of the Microsoft BI Platform, let's find out what's so innovative about SSAS 2005.

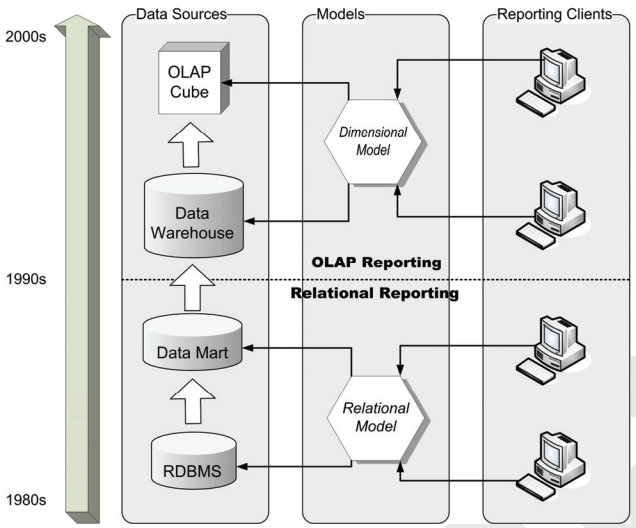


Figure 1.6 Today's BI reality is characterized by the "great divide" between relational and dimensional reporting models.

1.2 Understanding OLAP

SSAS 2005 goes beyond just being an OLAP and Data Mining server. The bold mission of SSAS is to break out of the OLAP space by *unifying* the relational and dimensional reporting models. To understand the new changes in SSAS 2005, it may be useful to take a short trip back in time and discuss the challenges that enterprise business intelligence has been facing for the past two decades. Let's use Figure 1.6 as a roadmap for our tour.

I will be quick to point out that as it currently stands, UDM shouldn't be viewed as a replacement of the relational reporting model or as a competing technology to standard reporting. Considering the unified vision of UDM however, one would expect that eventually both models will be merged in an integrated platform that provides both standard and OLAP reporting services.

Figure 1.6 depicts the evolution of both reporting models, relational and OLAP, and the "great divide" between them that is a common reality for most organizations today. On the one side of the dividing line is relational reporting where reporting processes are performed against relational models. A *relational model* could represent both a relational OLTP schema (normalized in the 3rd Normal Form) and a layer built on top of it (e.g. to serve ad-hoc reporting needs).

Note Strictly speaking, the dividing line between relational and OLAP reporting processes could be somewhat blurred. For example, standard reports can be generated from a dimensional data source (e.g. data mart) if this doesn't lead to performance issues. For this reason, Figure 1.6 shows a data mart in the relational reporting section. Chapter 2 discusses various reporting scenarios in respect to the data source type in more details.

On the other side are OLAP reporting processes that interact with dimensional models. We will use the term *dimensional model* to represent a data source that is specifically structured and optimized to address reporting and data analytics requirements, such as data marts, data warehouses, and OLAP cubes.

1.2.1 Relational Model

In the early 1980s, reporting needs were addressed by sourcing data directly from RDBMS. This model is still popular and widely used today. For example, if your preferred tool of choice for standard reporting is Microsoft Reporting Services, you can source the report data directly from RDBMS. As popular as it is, the relational reporting model has well-known deficiencies.

Not user-oriented

The relational model is designed with the system, not the end user in mind. Consequently, to create a report, the end user has to understand the database relational schema and know SQL. Isn't it strange that one of the job requirements for hiring a business analyst is to know SQL?

Performance challenges

Relational reporting could lead to performance issues. The report performance depends, to a large extent, on the data volume the report needs to process. What's more, running reports directly against RDBMS may very well slow down the performance of the OLTP system itself as a result of locking large number of rows. The reason for this is that pending transactions may be blocked while waiting for the report query to finish executing and releasing the read locks placed on the qualifying rows.

Note Once upon a time, I was called upon to troubleshoot mysterious query timeout errors that a client-server application was experiencing at random. After some troubleshooting, I pinpointed the culprit to be a popular ad-hoc reporting tool. Not only was the tool placing read locks on the SQL Server tables, but it wasn't releasing the locks even after the report was generated.

Lack of conformity

While the ad-hoc reporting models could abstract the underlying data schema to some degree, relational reporting is characterized by a lack of conformity. Business calculations and logic are not centralized in one place. For example, the database developer could define the net dollar amount calculation of a line item in the database itself, while the report designer could re-define it in the ad-hoc reporting model. This may be confusing to end users. Often, users are left to make their own interpretation of the data.

1.2.2 Dimensional Model

To solve some of the challenges of relational reporting, organizations started moving data from OLTP databases to data marts and warehouses. OLAP servers, such as Analysis Services, emerged in the late 1990s to provide the necessary CPU power to process the increased data volumes. A new *dimensional* model was born to make reporting more intuitive for less technically savvy users. SSAS embraces and extends the dimensional model, so let's spend some time explaining its terminology.

Note The terms *data warehousing* and *OLAP* are often used interchangeably but an important distinction exists. As its name suggests, a data warehouse can simply be described as a relational database that stores vast volumes of data. The term *OLAP*, on the other hand, represents the service layer introduced between the warehouse and users to make data available for fast retrieval and analysis. A data warehouse solution may not feature an OLAP server. Similarly, an OLAP server, such as SSAS 2005, may draw its data directly from the OLTP system, instead of from a data warehouse. That said, both data warehousing and OLAP use dimensional modeling as a core technique to organize data and make it more suitable for data analytics and reporting. There are some differences in their terminology though. For example, the data warehouse model refers to business metrics as *facts,* while OLAP uses the term *measures*.

Measures

Measures represent the numerical values (facts) that are used to measure business activity. Let's have a look again at our interactive report shown in Figure 1.7. This report displays the company sales performance. The only measure used in this report is *Sales Amount*. Other measures may include tax amount, discount, profit, order count, etc.

Measures are physically stored in relational tables called *fact* tables. These tables are usually narrow (don't have many columns) but can have thousands to millions rows of historical data. In addition, fact tables have foreign keys that link them to dimension tables.

Dimensions

As its name suggests, the main goal of the dimensional model is to allow users to slice and dice data using different perspectives called *dimensions*. Dimensions reflect the natural way end users would prefer to view and query data. For example, our report allows users to browse data by two common dimensions: *Product* and *Time*.

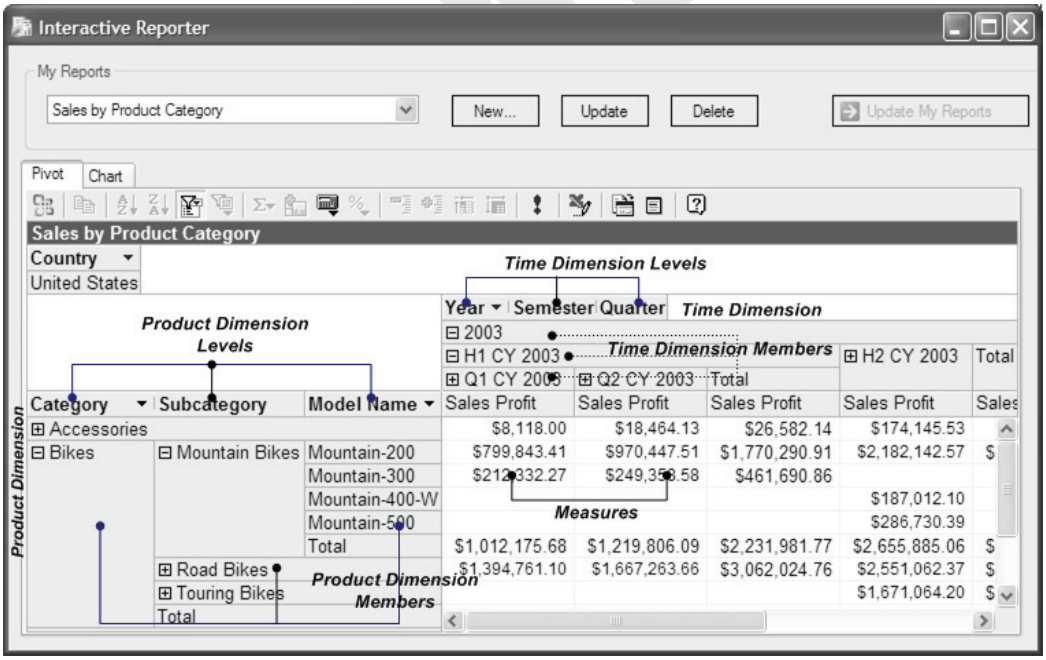


Figure 1.7 Dimensional model is designed to provide intuitive end-user reporting experience.

Dimension hierarchies

To facilitate drilling through data, dimensions may have hierarchies. For example, in our sample report, the time dimension hierarchy consists of the following dimension levels: *Year*, *Seminar*, and *Quarter*. The quarters can be further broken down into more granular levels, e.g. Month and Day. Similarly, the Product dimension hierarchy includes the *Category*, *Subcategory*, and *Model Name* levels. A dimension level summarizes *("aggregates")* data at that level. For example, since the user hasn't expanded the 2003 quarter, 2003 sales data on this report are aggregated at the Quarter level.

Dimension members

The actual dimension entities that belong to each level are called dimension *members*. Thus, the members of the Calendar Year level are *2003* and *2004*, while the members of the Category level are *Accessories*, *Bikes*, *Clothing*, and *Components*. Another way to depict the dimension hierarchy is to use an organizational chart, as the one shown in Figure 1.8.

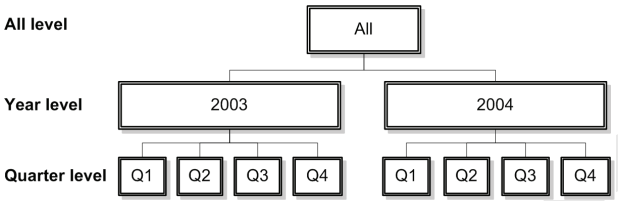


Figure 1.8 Dimension hierarchies organize members in levels.

The top level of a dimension is depicted as *All* level, which is how SSAS terminology refers to it. It is a handy way to retrieve the total aggregated value for the whole dimension. The All level usually serves as the default dimension member. For example, if I am to remove the Time dimension from the report, the report will show the product sales for all time periods. The members of the lowest level of a dimension hierarchy are called *leaf members*. For example, if the Quarter level is the lowest level in our Time dimension, the quarters are the leaf members.

Dimension tables

Dimension data are stored in relational tables called *dimension tables*. Unlike fact tables, dimension tables are usually wide (have many columns) but don't have many rows. The large number of columns is required to accommodate various dimension-related *attributes* that could be of interest to the end users. For example, a product dimension could have attributes such as product description, color, model, list price, listed date, discontinued date, etc.

The classic dimensional model defines two types of relational schemas (see Figure 1.9) that describe the relationship between the dimension and fact tables: *star* and *snowflake*. A star schema requires that a dimension hierarchy be contained within a single table. This requires the dimensional table to be denormalized. For example, going back to the sales report, if a star schema is chosen for the product dimension, the product data may look like this:

```
ProductID  ProductCategory  ProductSubCategory  ModelName
1     Bikes    Montain    Bikes Montain-200
2     Bikes    Montain    Bikes Montain-300
```

If the dimension hierarchy is left normalized, then the schema is of a snowflake type. Over the past decade, dimensional model scholars have staged fierce battles in a quest to find out which schema type reigns supreme. The "classic" dimensional model promotes the star schema. Indeed, if the user queries the relational database (e.g. data warehouse) directly, a snowflake schema will require the user to link the dimension tables together. This assumes that the user has

the necessary technical skills to do so, but wasn't this the problem that the dimensional model was trying to avoid in the first place? Moreover, in comparison with snowflake schemas, stars schemas are easier to maintain and update.

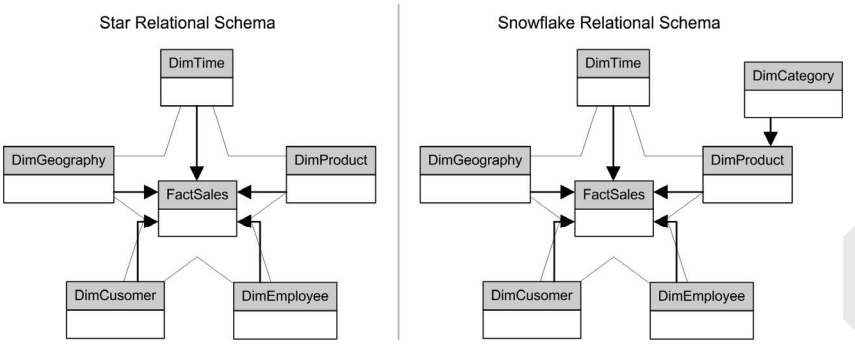


Figure 1.9 The dimension-to-fact relational schema could be star or snowflake.

On the other hand, snowflake schemas could support more flexible relationships, such as referenced and many-to-many relationships (discussed in chapter 5). In addition, they could save storage space with large dimensions. SSAS takes a nonchalant view of this schema debate and supports both schema types. A noticeable exception is the dimension writeback feature which is only supported with star dimensions. In real life you should carefully weigh out the pros and cons of both approaches and choose the schema that best meets your requirements. In general, I would recommend you gravitate toward star dimensions whenever possible and consider "upgrading" them to snowflake dimensions if needed. Dimension terminology has some additional classifications but, for time being, this is all you need to know about dimensions.

Cubes

The Sales by Product Category report (see again Figure 1.7) is an example of a two-dimensional report. SSAS is not limited to storing and displaying information in a two-dimensional format. As I've mentioned, one of the FASMI requirements is that every OLAP system must be *multidimensional*, so users can view data from as many dimensions as they want. In addition, the OLAP multidimensionality shouldn't sacrifice performance.

Definition The cube is the logical storage object in SSAS. It combines dimensions and measures to provide fast multidimensional access to the cube data.

How are cubes implemented?

To achieve these demanding requirements, SSAS employs the logical concept of a *cube* as a main storage object. The term *logical* in our definition means that, unlike relational objects (tables, views, etc.), the cube doesn't have a physical realization. Thus, if you browse the SSAS storage folder (the default is C:\Program Files\Microsoft SQL Server\MSSQL.2\OLAP\Data), you won't find any multidimensional or other exotic structures. Instead, you will find a large number of files that store data in binary format. During runtime, SSAS performs its "magic" and exposes the content of these files to clients as a multidimensional cube. Figure 1.10 shows how you can visualize a cube that has three dimensions.

Suppose that we connect OWC to this cube. We can now browse by three dimensions – Year, Product, and Territory (for the sake of simplicity, let's assume that the three dimensions don't have hierarchies). The intersection of the cube dimensions is called a cube *cell*. For example, the shaded cell in the figure is found at the intersection of the following dimension members

– *2003* (Year dimension), *Bikes* (Product Category dimension), and *North America* (Territory dimension). Each cell of the cube holds a single value.

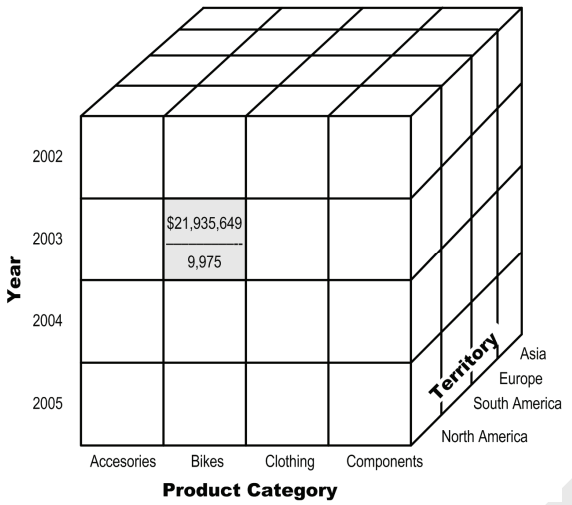


Figure 1.10 The cube is the main storage object in SSAS. The cube is a multidimensional structure that consists of dimensions and measures.

Tip When visualizing the cube, it may be helpful to think of the cube measures as separate dimensions. In our case, we have two measures (sales amount and order count). Each intersection between the cube dimensions and measures will result in a single cell. Since it is difficult to show the cube in four perspectives, I split the cell in two - the upper cell in the figure shows the sales amount; the lower shows the order count.

We live in a three-dimensional space, so it is natural for us to visualize a three-dimensional cube. However, SSAS 2005 cubes can and usually have more than three dimensions. In fact, SSAS 2005 cubes support more dimensions than you will ever need (in the range of billions). Since many of the cube limitations in the past have been removed, an SSAS 2005 cube could really be viewed as a "super-cube". In fact, you are encouraged to build your entire OLAP layer on top of an enterprise-wide data warehouse with a single SSAS 2005 cube.

Cube storage

From a user perspective, a cube appears to store all fact rows and have aggregated values for each cell. For example, if the user hasn't expanded the Calendar Year hierarchy, the cube will show the annual totals. Similarly, if the user drills down to the Quarter level, the cube will readily return the quarter aggregated values.

In reality, the cube may have neither the fact rows, nor the aggregated values stored in it. In this case, the cube will aggregate the values on the fly. As an UDM designer, you can tell SSAS where the fact data and aggregations will be kept – in the relational database (ROLAP storage), in the cube (MOLAP storage), or both (HOLAP storage). SSAS cubes usually perform best when all data (details and aggregations) are stored in the multidimensional store on the server, i.e. when MOLAP storage model is used. In this way, the cube can answer all queries *without* querying (and impacting the performance) of the underlying data source. On the downside, since the server keeps a copy of dimension and fact data (MOLAP storage option), it has to be updated (*processed*) when the dimensional structure or source data changes.

Dimensional model challenges

By now, you are probably convinced that the dimensional model has many advantages over its counterpart, the relational model. It is the cornerstone of the OLAP technology and it is very popular today. At the same time, the dimensional model is not without its shortcomings. Let's mention a couple of them.

Data fidelity lost

While dimensions reflect the natural way end-users prefer to analyze data, important dimension characteristics could be lost when transforming OLTP data to fit into the "classic" dimensional model. For example, the Product dimension we've discussed has a hierarchy that reflects the natural way end users would prefer to browse the product sales data – by category, subcategory and product name.

However, the original Product relational schema may have included columns for product color, size, model, etc. With the classic dimensional model these attributes could be simply "lost" when the Product dimension hierarchy is created or require additional dimensions to be implemented. The end result is that the user may not be able browse or filter data using these attributes.

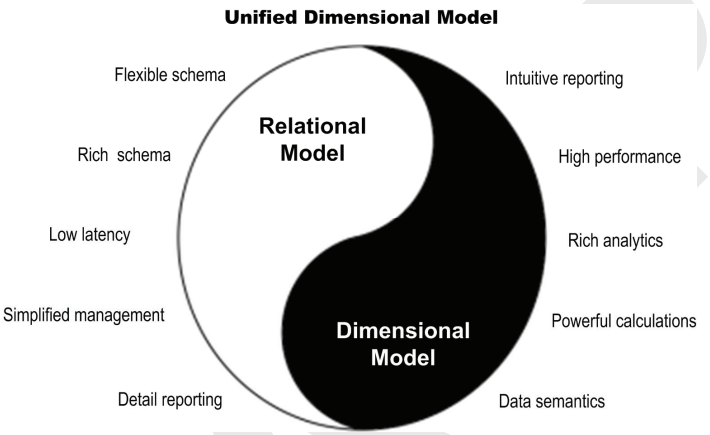


Figure 1.11 The SSAS 2005 Unified Dimensional Model solves some of today's BI challenges by uniting the relational and dimensional models.

Data latency

The second well-known issue surrounding the OLAP model is data latency, since the same data extracts would exist in all the OLAP repositories – data marts, data warehouse, and OLAP cubes. This is further aggravated by latency issues. It is not uncommon for ETL processes to take hours, if not days to complete. By the time data arrives in the OLAP cubes and it is available for reporting, it may be significantly outdated.

1.3 Understanding The Unified Dimensional Model

In summary, most of today's organizations have accumulated a mixture of two distinct storage models, relational and dimensional, and each of them has its own pros and cons. What is exciting about SSAS 2005 is that is starts a novel journey to unite both relational and dimensional models by combining the best aspects from both. This model is called *Unified Dimensional Model* or *UDM*

for short, as shown in Figure 1.11. UDM is *unified* because its goal is to unite the relational and dimensional models. It is *dimensional* because it has its roots in the dimensional model.

Definition The SSAS Unified Dimensional Model (UDM) converges the relational and dimensional models. The physical manifestation of UDM is the SSAS 2005 cube.

1.3.1 Relational Model Features

At this point, some of the OLAP-savvy and perhaps skeptical readers may wonder if the notion of UDM is not too far-fetched. Let's enumerate some the most prominent characteristics of UDM that justify its bold vision to become the "next stage" for OLAP. We will start with the UDM features that bring it closer to the relational reporting model.

Rich schema

As I mentioned, one area where the classic OLAP falls behind compared to the relational model is loss of data fidelity. The end product of the classic dimensional model could be intuitive user hierarchies but at the cost of losing the ability to browse data from other perspectives. One of the most exciting new features of SSAS 2005 is *attribute-based dimensions*. With UDM, each column (attribute) from a dimensional table can be exposed as a hierarchy by itself. In fact, the UDM cube space is a product of attribute-based hierarchies, while multilevel hierarchies are optional.

For example, consider a Product dimension table that includes product attributes that are not part of or are not naturally related to the Product Category dimension, such as product name, list price, and color. UDM makes it possible to report off these attributes (see Figure 1.12).

My Sales Report

Drop Filter Fields Here

			Calendar Year			
			⊞ CY 2003		⊞ CY 2004	
Product Name	List Price	Color	Sales Amount	Order Count	Sales Amount	Order Count
⊟ All-Purpose Bike Stand	⊟ 159	NA	$16,798.21	119	$17,081.48	119
⊟ AWC Logo Cap	⊟ 8.6442	Multi	$4,362.08	197		
⊟ AWC Logo Cap	⊟ 8.99	Multi	$15,864.33	1,146	$15,690.91	1,411
⊟ Bike Wash - Dissolver	⊟ 7.95	NA	$8,898.65	621	$7,650.05	664
⊟ Cable Lock	⊟ 25	NA	$5,715.54	111		
⊟ Chain	⊟ 20.24	Silver	$5,377.26	136	$3,469.81	114
⊟ Classic Vest, L	⊟ 63.5	Blue	$4,112.30	73	$6,836.54	120
⊟ Classic Vest, M	⊟ 63.5	Blue	$50,246.23	281	$33,062.58	260
⊟ Classic Vest, S	⊟ 63.5	Blue	$80,456.95	332	$64,306.94	341

Figure 1.12 UDM is an attribute-based model and it supports both multi-level and attribute-based hierarchies.

As Figure 1.12 shows, UDM doesn't force you to use the Product Category dimension when browsing sales data by product. Instead, just like with relational reporting, you can drop the product related attributes side-by-side. To achieve the same effect in SSAS 2000, the cube designer had to create multiple dimensions which led to duplication of definitions and storage.

Flexible schema

One of the trademarks of the relational reporting model is that it enjoys a flexible database schema. Indeed, complex relationships, e.g. one-to-many, many-to-many, outer joins, etc, have been a part of the core relational schema model from its beginning. Similarly, with UDM, you are not confined to star and snowflake schemas anymore. In addition, UDM introduces new dimension roles (referenced, many-to-many, role playing, degenerate, etc) that enable new scenarios. Let's mention one scenario that UDM makes possible.

With the "classic" dimensional model it has been traditionally difficult to join two fact tables that summarize data at different levels (called grain in the OLAP terminology). For example, the grain of the time dimension in your cube may be days. At the same time, you may also have a

sales quota fact table which stores the sales person quotas at a quarter level. Suppose that your requirements call for joining both tables to compare the sales person's performance and her quota side-by-side.

There were a few techniques in SSAS 2000 to address the multi-grain issue, including parent-child dimensions, inserting "fake" members, using virtual cubes, but none of them presented a clean solution. In contrast, UDM addresses this issue gracefully by simply allowing you to define the grain at which a given dimension joins both fact tables, as shown in Figure 1.13. In this case, I have specified that the time dimension will join the fact table at Calendar Quarter level. It can't be easier, really!

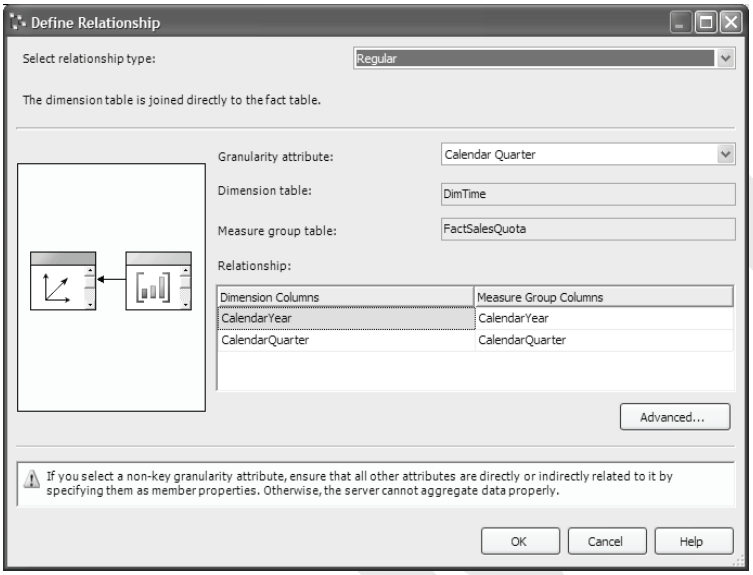


Figure 1.13 With SSAS 2005 you can easily join dimension and fact tables at different grains.

Low latency

The business requirements of today's economics mandate ever-shrinking time windows to make data available for reporting and analysis. UDM makes real-time OLAP and building low latency OLAP applications a possibility. There are two techniques to accomplish this. The first technique involves pushing the data directly into the cube (push-mode processing) without updating the underlying data source. For example, you may have a sales cube built on top of a data warehouse database. The processes of extracting, transforming, and loading the order data into the data warehouse may take significant time to execute. Yet, business requirements may dictate new order data to be available for reporting within a few hours. With SSAS 2005, you can build a light-weight data integration package that runs frequently and trickle-feeds the new orders into a cube completely bypassing the data warehouse tables. Unresolved dimensions can be defaulted to unknown values until the "full-blown" data integration package executes.

The second technique is more suitable for cubes built directly on top of OLTP databases. It allows you to put the cube in "auto-pilot" mode by leveraging the new *proactive caching* feature. When proactive caching is enabled on an SSAS 2005 cube, the cube can detect changes to the underlying data and automatically update its dimensional structures. I will show you how you can implement real-time OLAP in chapter 15.

Simplified management

There are several provisions in SSAS 2005 to simplify the management effort. SSAS 2005 removes the limitation that a cube can have only one fact table. What this means to you is that you can store the entire dimensional model into a single "super-cube". If you worry about what impact this will have on performance and scalability, rest assured that your UDM model scales well. SSAS 2005 gives you the means to scale out large deployments, e.g. by partitioning the cube across multiple servers and load-balancing these servers in a cluster.

To minimize the management and design effort, BI Studio provides a slew of wizards and designers. For example, The Cube Wizard can help you "jumpstart" the cube dimensional model by heuristically examining the relational schema and suggesting measures. As its name suggests, the New Dimension wizard walks you through the process of adding a new dimension. Finally, a brand new .NET-based object model called Analysis Management Objects (AMO) has been introduced to supersede the SSAS 2000 Decision Support Objects (DSO) model and allow developers to implement SSAS management features in their .NET applications.

Detail reporting

You shouldn't view relational and OLAP reporting as competing but, rather, complementary technologies that address different user needs. Therefore, SSAS 2005 will not be my tool of choice for generating OLTP-based relational reports, such as a common order report (order header with line items). Instead, use SQL Server Reporting Services for your standard reporting needs.

Yet, UDM provides several ways to view detail data. First, you can use a feature called *drillthrough* to see the underlying rows under a given dimension member or a cube cell. For example, the cube may aggregate sales order data on a daily basis. Yet, users may want to view the individual orders placed on a given day. You can implement this requirement by enabling drillthrough for that cube. This, of course, assumes that the OLAP browser supports this feature.

Second, you can use UDM *actions*. Considering the above example, if drilldown is not a good fit or is not supported by the OLAP browser, you can implement an action to launch a standard report (e.g. a Reporting Services tabular report) to display the invoices. Once the action is configured, the end-user could right-click on the cube cell in question to launch the report from the dropdown menu.

1.3.2 Dimensional Model Features

While SSAS 2005 comes with exciting new features, its stays close to its roots. The core dimensional concepts are the same. Now let's discuss briefly how UDM leverages and enhances the dimensional model.

Intuitive reporting

Intuitive end-user oriented reporting has been the hallmark of the dimensional model since its inception. As I mentioned, UDM makes the user experience even richer by enabling reporting scenarios that are not part of the core dimensional model, such as actions, drillthrough, and attribute-based reporting.

High Performance

OLAP user experience is directly correlated to the query performance. As noted, there are two factors that contribute most to the SSAS efficiency – the optimized query engine and the cube multidimensional model. SSAS 2005 brings additional performance enhancements. Most of them are related to the fact that SSAS 2005 cubes are not limited to having one fact table anymore. What this means to you is that you don't have to use virtual cubes anymore. Another cause of grievance in the past was that SSAS 2000 required all dimensions to be loaded in memory. To address this issue, SSAS 2005 loads dimensions in memory on as-needed basis.

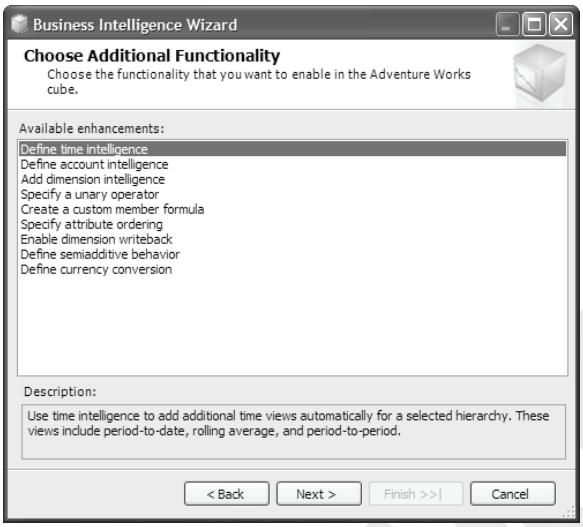


Figure 1.14 Use the Business Intelligence Wizard to add advanced business intelligence features.

Rich analytics

There are many new features in SSAS 2005 that bring rich business intelligence features to end users. Some of them were available in the previous releases as well, but they were not straightforward to implement. To facilitate defining advanced analytics features in SSAS 2005, the Analysis Services team introduces a brand new *Business Intelligence Wizard* (see Figure 1.14). For example, one of the common reporting requirements is to compare data over parallel time periods, e.g. sales figures between the first quarters of two consecutive years. As Figure 1.14 shows, the *Define time intelligence* feature of the Business Intelligence Wizard can help you save time by generating such time-related metrics for you. We will see how to add advanced business intelligence features to UDM in chapter 10.

Powerful calculations

The calculation engine has been completely redesigned in SSAS 2005. You still have to know MDX to define your calculations, but authoring and testing MDX logic is much easier now. Issues that have pestered MDX developers in the past (solve order, pass) have simply disappeared. One welcome enhancement is that all MDX constructs (calculated members, cell calculations, named sets) are centralized in one place (the Calculations tab of the Cube Designer), as shown in Figure 1.15.

In this case, I use the Calculations Script View pane to define several calculated members, named sets and MDX scripts. .NET developers will undoubtedly find many similarities between the Script View and Visual Studio.NET. Similar to working with the Visual Studio.NET IDE,

MDX developers can use breakpoints, debug MDX scripts, and see the effect of the executed script. For example, Figure 1.15 shows that I've just stepped out of a breakpoint and the Pivot Table has highlighted the effected cell. The SSAS calculation engine is also extensible. Developers can plug in additional programming logic in the form of SSAS stored procedures that can be written in any .NET-compatible language. We will cover MDX programming in part 3 of this book.

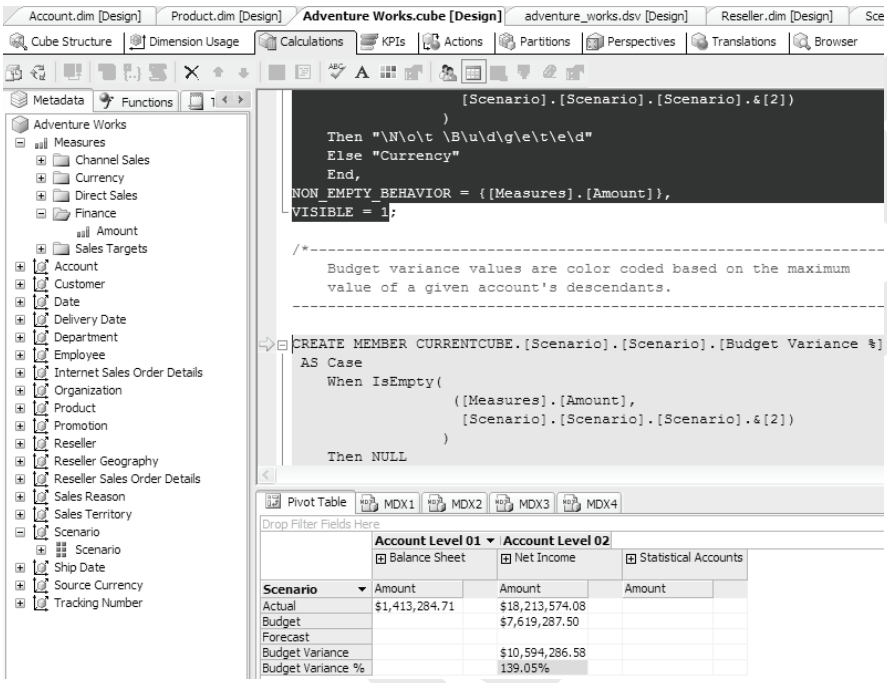


Figure 1.15 Use the MDX Script View to centralize the cube calculations in one place and debug MDX scripts.

Data semantics

Unlike the relational model, which is oblivious to the meaning of data, UDM can be "educated" to understand the data semantics. For example, UDM understands *Time* dimension and chart of accounts. In the later case, UDM knows how to map the account data to pre-defined categories, such as income, expenses, and taxes, and apply the required calculations. We will see how to implement advanced intelligence features in chapter 5.

1.3.3 UDM Components

At this point, you are probably curious about how UDM is implemented. The physical manifestation of UDM is the new Analysis Services 2005 cube. Therefore, I will use the terms *UDM* and *cube* interchangeably throughout the rest of the book. As Shrek would undoubtedly attest, just like ogres, UDM has layers. Figure 1.16 shows how you can visualize UDM. As you could notice, I've stacked the UDM components in the chronological order they will be typically implemented. To reflect this natural flow, the book is organized in the same way and the numbers shown inside the individual layers represent the chapter(s) where the corresponding layer is discussed.

Data source view (DSV)

UDM is based on a logical data schema that seeks to present the data from the relational data store in a standard and intuitive way. UDM implements the schema in the form of a data source view (DSV). Besides providing the UDM data schema, DSV isolates the cube dimensional model from changes in the underlying relational databases.

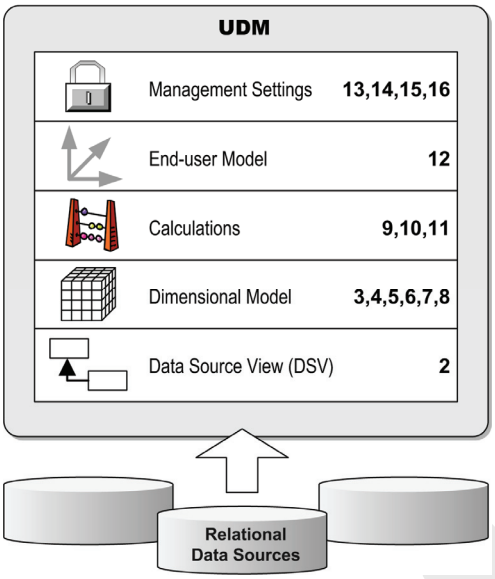


Figure 1.16 The physical manifestation of UDM is the SSAS 2005 "super-cube". Its main building blocks are data source view, dimensional model, calculations, end-user model, and management settings.

Dimensional model

Once DSV is created, the next step will be implementing the cube dimensional model. The end result of this process is the cube definition consisting of measures and dimensions with attribute and/or multilevel hierarchies.

Calculations

Only in rare cases, the dimensional model alone will fully meet your needs. As a UDM designer, you can augment your cube with specific business logic in the form of MDX expressions.

End-user model

As noted, the main design goal of the dimensional model is to provide intuitive end-user reporting and data navigation experience. By "end-user model", we will understand the additional features you can build on top of the dimensional layer to provide even richer data semantics. These features include Key Performance Indicators (KPIs), actions, perspectives, and translations. For example, if the cube will be browsed by international users, dimension levels could be localized by using translations. Or, you can use perspectives to define named subsets of large and complex cubes for easier navigation.

Management settings

At last, the cube is ready for prime time. As a last step, a savvy administrator would configure the cube to meet various operational requirements, including availability, latency, and security. For example, in this stage the cube administrator will configure which users will be able to access the cube, when and how the cube data will be updated, the cube storage model, etc.

1.3.4 To UDM and Beyond

By now, you should be able to understand the UDM goal to unite the best of both worlds (relational and dimensional) and become a bridge between the users and data. One could envision UDM to evolve in time to a point where the other relational and dimensional models are simply not needed and will disappear, as shown in Figure 1.17.

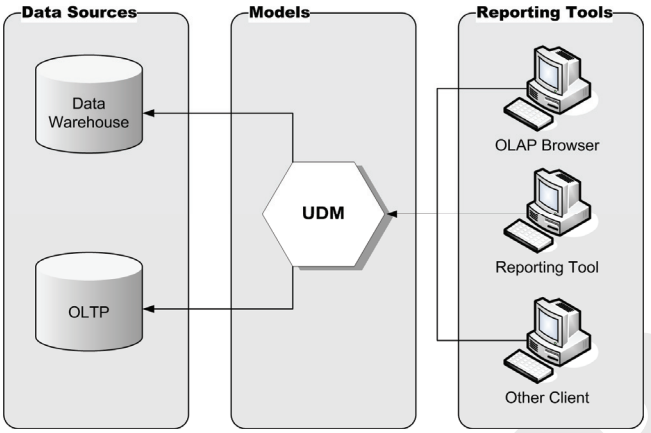


Figure 1.17 In time, UDM could replace the other reporting models to provide both relational and dimensional reporting needs.

When this happens, UDM will be able to serve both relational and dimensional reporting needs. Besides simplicity, having a single model will bring also conformity. Business logic and calculations could be defined in one place. As an added bonus, all reporting clients will be able to benefit from the performance boost they will get from SSAS.

Note To some extent, SQL Server 2005 and the Microsoft BI platform give you the tools to materialize the "unified" vision of having UDM as a focal point for data analytics. For example, as I will demonstrate in chapters 18 and 19, Reporting Services and the Microsoft Office analytics tools integrate well with UDM. However, as noted before, you shouldn't expect UDM to address all reporting requirements equally well. That's why I don't suggest you quickly throw away your reporting tools in favor of UDM. In my opinion, medium to large-size organizations will benefit most from leveraging UDM as a central repository.

1.4 Analysis Services Architecture

Readers who have prior SSAS experience have probably heard the popular saying that all roads to SSAS 2000 go through the PivotTable Service (PTS). PTS was the primary method for interacting with Analysis Services 2000 to perform tasks such as connecting to a cube and retrieving data. It was designed as a client-side component and, as such, it had to be installed on the machine where the client application was installed. PTS helped query performance by providing client-side caching. In many cases, however, PTS was simply getting in the way. For example, PTS wasn't designed to work with server-based applications. The good news is that the SSAS 2005 architecture is entirely server-based, as shown in Figure 1.18. This enables flexible client integration scenarios, e.g. implementing thin clients that require no installation footprint.

Let's discuss the SSAS building blocks starting with the Analysis Services server.

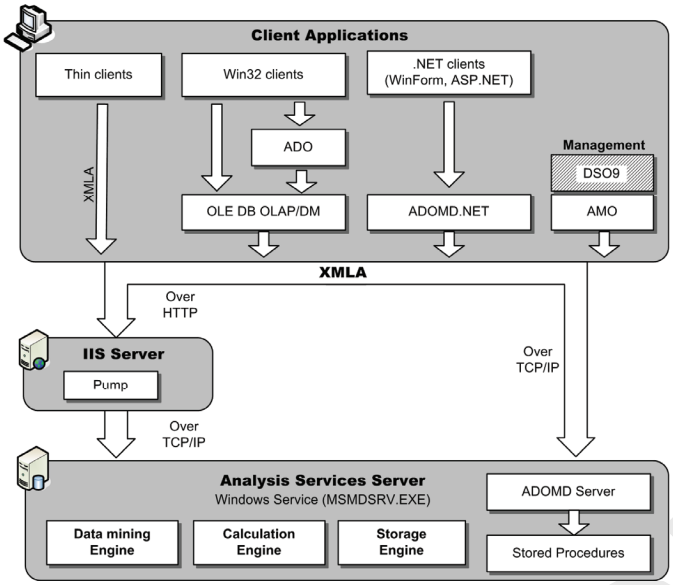


Figure 1.18 SSAS 2005 is implemented as a server-based middle-tier platform. At the heart of the SSAS architecture is the Analysis Services server which provides storage, calculations, and data mining services.

1.4.1 Analysis Services Server

At the heart of the SSAS architecture is the Analysis Services server. The SSAS server provides the following main services:

- *Storage* – The Storage Engine is responsible for storing and processing SSAS objects. It also keeps the UDM object definition (called *metadata*).
- *Calculations* – The Formula Engine handles MDX queries and expressions.
- *Data mining* – The Data Mining Engine processes mining queries and returns prediction results.

As with the previous releases, the SSAS server is implemented as a Windows service called MSMDSRV.EXE written in C++ native code. By default, the setup program installs the Analysis Services Server in C:\Program Files\Microsoft SQL Server\MSSQL.2. You can install more than one instance of SSAS 2005 on a single machine and different versions (e.g. 2000 and 2005) can co-exist side-by-side. Perhaps the most interesting SSAS architectural change is that it embraces XML for Analysis (XMLA) as a native protocol. In fact, you cannot communicate with SSAS in any other way than using XMLA. Given the strategic importance of XMLA, let's spend some time introducing this protocol.

1.4.2 XML for Analysis (XMLA) Specification

As its name suggests, the XMLA protocol conforms to XML-based grammar called XMLA specification. The purpose of this specification is to standardize the data access between OLAP clients and analytical data providers, such as SSAS. Since its debut in mid-2001, XMLA gained support with more than twenty vendors, including the three founding members -- Microsoft, Hyperion, and SAS (see the Resource section). The XMLA specification is managed by the XMLA council (*xmla.org*). As of the time of this writing, the most current version of the XMLA specification is 1.1. This is the version that is implemented in SSAS 2005.

XMLA embraces the SOAP protocol for sending and receiving XMLA messages to a XMLA-capable provider. The actual SOAP grammar is very simple (Figure 1.19). It describes just two methods, *Discover* and *Execute*, which every XMLA provider must support.

Discover

An OLAP client calls the *Discover* method to obtain the metadata that describes OLAP and data mining objects. For example, an OLAP client can ask the SSAS 2005 server to return a list of all cubes defined in an Analysis Services database by invoking the *Discover* method.

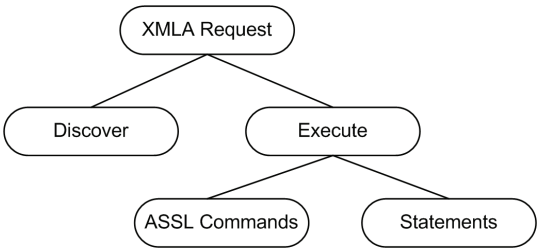


Figure 1.19 The XMLA protocol consists of Discover and Execute methods. A client can use the Execute method to send ASSL commands or statements.

Execute

Execute is an action-oriented method. A client can invoke the *Execute* method to send either ASSL commands or statements.

Analysis Services Scripting Language (ASSL)

Analysis Services Scripting Language (ASSL) is an XML-based grammar that describes the UDM metadata (DDL grammar) and commands.

- *Data Definition Language (DDL) grammar* – DDL is the internal representation of metadata in Analysis Services 2005. DDL describes the object definition, e.g. a cube definition. You can see the DDL grammar by right-clicking on the object in the BI Studio Solution Explorer and choosing *View Code*.
- *Command language grammar* – A subset of ASSL defines some action-oriented commands that could be sent to the server, e.g. for processing, altering, or creating objects. For example, each time you process a cube, BI Studio generates an ASSL script that includes a *Process* ASSL command.

The Execute method can (and most often) is used to send also statements.

Statements

With OLAP, the Execute statements describe MDX queries, while with data mining, they contain DMX queries. The query results are returned as a rowset (for SQL and data mining queries), or in the form of a more complex structure called MDDataSet in the case of OLAP (MDX) queries.

What may be confusing is that both MDX and DMX also define DDL statements. For example, MDX defines a *CREATE MEMBER* construct to create a new calculated member, while DMX supports a *CREATE MINING MODEL* to create a data mining model. It is important to understand that these DDL statements have nothing to do with the ASSL DDL grammar although they have ASSL equivalents. In addition, MDX and DMX DDL statements are less flexible than DDL.

XMLA Connectivity Options

SSAS 2005 gives you two connectivity options to send XMLA messages to an Analysis Services server. By default, the client communicates with the server via TCP/IP. However, SSAS can be configured also for HTTP connectivity to enable web-based integration scenarios.

XMLA over TCP/IP

The XMLA over TCP/IP connectivity option is more suitable for intranet deployments. With this option, the SOAP messages are serialized in binary format and sent over TCP/IP to the SSAS server. You don't need to take any extra steps to configure SSAS to use XMLA over TCP/IP. For example, if you use Office Web Components and set its connection string to use the OLE DB Provider for Analysis Services 9.0, the provider will communicate with SSAS over TCP/IP.

Compared to HTTP connectivity, the XMLA over TCP/IP connectivity option has slightly better performance since no additional layers are introduced between the client and the SSAS server. The tradeoff is that the client has to be able to connect directly to the port the SSAS server is listening to (2383, by default) which may conflict with firewall policies.

XMLA over HTTP

In this case, IIS is used as an intermediary to receive the HTTP requests. To set up SSAS 2005 for HTTP connectivity, you need to set up an IIS virtual root that will host the SSAS XMLA provider (a.k.a. *Pump)*. The purpose of the Pump component is to accept the incoming HTTP requests from IIS and forward them to the SSAS server over TCP/IP. Once the HTTP connectivity is set up, change the connection string to point to the IIS virtual root, e.g. *http://<ServerName>/<VRoot>/msmdpump.dll.*

Consider the XMLA over HTTP option when you need to connect to SSAS over the Internet or when direct connectivity to SSAS is not an option. For example, security requirements may enforce access to SSAS only over port 80 (HTTP) or 443 (SSL). HTTP connectivity could be a good choice when you cannot install programming libraries, e.g. when you need to implement thin or non-Windows clients, e.g. a Java-based OLAP client running on UNIX box. The XMLA over HTTP connectivity option is described in more details in chapter 16.

1.4.3 SSAS Clients

OLAP clients have several available programming interfaces to connect to SSAS 2005. No matter which connectivity option is chosen, the interface library translates the calls to XMLA. Code samples demonstrating different integration options are provided in chapter 17.

Thin clients

Thanks to its entirely server-based architecture and support of industry-standard protocols (HTTP, XMLA, and SOAP), SSAS 2005 can be integrated with any SOAP-capable client running on any platform with no installation footprint. In this case, the client is responsible for constructing SOAP requests conforming to the XMLA specification and interpreting XMLA responses.

Win32 native clients

C++ clients would typically connect to SSAS 2005 using the OLE DB for Analysis Services. This is how OWC connects to SSAS 2005. The provider you need is OLE DB Provider for Analysis Services 9.0 (Provider=MSOLAP;3 in the connection string). You cannot use an older

provider, e.g. version 8.0, because only version 9.0 knows how to translate the OLE DB for Analysis protocol to XMLA. COM-based clients, such as Visual Basic 6.0 clients, can connect to SSAS 2005 by using the ADO **M**ulti**D**imensional library (ADOMD) which is implemented as a COM wrapper on top of the OLE DB provider.

.NET clients

.NET clients can connect to SSAS 2005 using the ADO MultiDimensional for .NET library (ADOMD.NET). ADOMD.NET doesn't require the OLE DB Provider for Analysis Services 9.0 to be installed on the client machine. It is implemented as a light-weight managed wrapper on top of XMLA. Interestingly, SSAS provides also a server-side object model in the form of the ADOMD Server library (ADOMD.NET Server) residing inside the Analysis Services server. The main difference between ADOMD Server and ADOMD.NET is that the former doesn't require the developer to set up a connection with the server explicitly before sending queries or navigating the server objects. Other than that, both libraries provide almost identical set of objects.

For management tasks, .NET developers would use the brand new Analysis Management Objects (AMO) library. With the AMO library, you have access to the full Analysis Services object hierarchy, including servers, databases, data source views, cubes, dimensions, mining models, and roles. Developers would typically use the AMO library to automate routine management tasks, such as database synchronization and processing. AMO supersedes the Decision Support Objects (DSO), the object model of SSAS 2000. DSO is still available for backward compatibly in the form of the DSO9 object library. However, as the documentation states, DSO will be removed in the next version of Microsoft SQL Server and you are strongly encouraged to migrate your management applications to AMO.

1.5 Analysis Services in Action

Let's demonstrate some of the concepts that we've discussed so far in a short hands-on lab. Before we start, let's introduce an imaginary company, called Adventure Works Cycles. *Adventure Works Cycles* (or AWC, for short) manufactures and sells bicycles to resellers and individuals in North America, Europe, and Australia. In 2001, its first year of operation, AWC sales accounted for more than ten million dollars. Since then, the AWC business has been growing exponentially to reach the record high of forty million dollars in total sales in 2003. However, the AWC business took a downturn in 2004 and sales fell below the projected figures. Direct sales to customers remain constant, while resales fell almost fifty percent.

1.5.1 Introducing Adventure Works Sales OLAP System (SOS OLAP)

The AWC management has decided to implement a BI reporting solution to get more insight into the company performance and its customers. And, as you probably guessed it, AWC has hired you as an architect to lead the design and implementation of the strategic Adventure Works **S**ales **O**LAP **S**ystem, or SOS, as the AWC information workers affectionately refer to it to emphasize its much awaited arrival. It is worth mentioning that, as useful as our fictitious system could be, it is not meant to serve as a complete solution for sales analytics. Instead, you should view it as a sample whose main objective is to help you learn SSAS 2005.

The current system

After a series of acquisitions, Adventure Works Cycles operates several software systems spanning different technologies. The employee data is stored in an Oracle-based HR system, while the manufacturing data is captured in an IBM mainframe system. The sales ordering data is stored in a SQL Server database 2005 called *AdventureWorks*.

Note SQL Server 2005 comes with two sample databases. AdventureWorks simulates an OLTP sales order database, while AdventureWorksDW imitates a data warehouse database that sources its data from the AdventureWorks database. You can find the AdventureWorks OLTP Visio schema in the Database\AWC folder of the book source code. As you can see by browsing its seventy tables, the AdventureWorks database is inherently more complex than FoodMart or other SQL Server sample databases that you may have encountered in the past.

The sales representatives use a Windows Form intranet application to capture orders placed through the resale channel. Web customers purchase AWC products online through the AWC Intranet website. In both cases, the sales orders are stored in the *AdventureWorks* OLTP database. A sales order is assigned different status codes as it goes though the order management pipeline, e.g. *In Process*, *Approved*, *Shipped*, or *Cancelled*. AWC has a cutoff period of one month for the order to be considered finalized (*Shipped* or *Cancelled*).

AWC has already built a data warehouse to archive the sales history. Data from relevant systems is extracted, transformed, and loaded in the data warehouse. Shipped sales orders that are older than a month are extracted from the *AdventureWorks* OLTP system and offloaded to the warehouse. The role of the data warehouse is fulfilled by the *AdventureWorksDW* sample database, which can be installed by running the SQL Server setup program.

Reporting challenges

Currently, enterprise reporting needs are addressed by running standard reports directly against the warehouse database. This reporting model is characterized by several of the standard reporting deficiencies we enumerated in 1.2.1, including:

- *Inadequate reporting experience* – Business analysts complain that they cannot slice and dice data from different perspectives easily. Different reporting tools are used based on the user skill set, ranging from Excel spreadsheets to high-level reporting tools, such as Reporting Services.
- *Performance issues* – Reports that aggregate large volumes of data take a long time to execute.
- *Insufficient data analytics* – Complex business logic and calculations cannot be easily implemented on top of the data warehouse relational schema. Subsequently, they are often redefined from one report to another and stored as part of the report, instead of in a central repository. In addition, the current reporting model doesn't support pattern discovery and forecasting.

To address the current report deficiencies, you've decided to use SSAS 2005 as an OLAP engine that will power the new SOS system.

The solution

You envision the SOS system to provide three major functional areas – a historical layer, a real-time UDM layer, and a reporting layer.

Historical UDM

The main purpose of the SOS system is to provide fast and uniform access to the data stored in the data warehouse. This objective will be achieved by building a UDM layer in the form of an Analysis Services 2005 cube on top of the warehouse database. The historical UDM layer will serve most of the OLAP requirements and all data mining requirements.

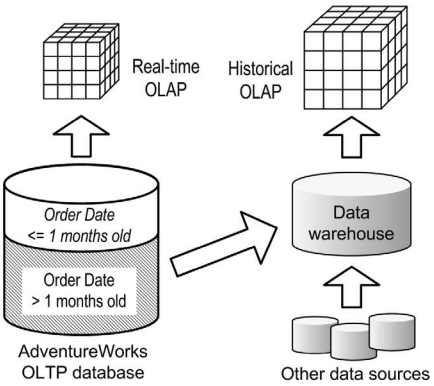


Figure 1.20 The SOS solution will feature the real-time and historical UDM layers.

Real-time UDM

To address real-time BI needs for reporting off volatile order data that hasn't been offloaded to the data warehouse, a real-time (hot) OLAP layer will be built directly on top of the Adventure-Works OLTP system. The real-time UDM layer will be implemented as a second SSAS 2005 cube that will provide a subset of the data analytics feature set of the historical UDM.

Note We will keep the real-time UDM light-weight on purpose. From a learning perspective, there is no point duplicating the same feature set in both the historical and real-time UDM models. Instead, when implementing the real-time UDM, our focus will be demonstrating UDM features that are particularly relevant to low-latency OLAP solutions, such as data source views and proactive caching. In real life, of course, you can have a more sophisticated and feature-rich real-time layer if required.

The term *real-time* here means that the cube will pick up changes in the transactional data almost instantaneously, instead of requiring explicit processing.

Reporting layer

Since the AWC business analysts have different reporting needs, you envision leveraging several BI reporting tools for presenting data to the end users, including custom applications, Reporting Services, and Microsoft Office.

1.5.2 Your First OLAP Report

Suppose the Adventure Works business analysts would like to be able to generate interactive sales reports to slice and dice data from different angles. Let's see how we can address this requirement by using two reporting technologies: standard reporting and OLAP reporting.

Standard reporting

In the absence of an OLAP reporting solution, the most common option is to author standard or ad-hoc reports that submit SELECT SQL statements directly to the OLTP database. These SELECT queries are typically multi-join statements that link several relational tables together to

fetch the report data. An example of a SQL SELECT statement that will produce a standard report similar to the interactive report shown in Figure 1.1 is included in SQLQuery.sql file.

There may be several potential issues with generating reports sourced directly from an OLTP database. To start with, the report query may impact the performance of the OLTP database. The query may take long a time to execute. On my machine (HP NW8000, 1.8 GHz Pentium M single CPU, 2 GB RAM) the query in the SQLQuery.sql file takes about three seconds to execute. Not that bad, you may say. Of course, we need to factor in the amount of data processed. In our case, the SalesOrderDetail table in the sample AdventureWorks database has 121,317 order line items. Now, imagine that the same query is fired against a much bigger transactional or warehouse database. Assuming linear regression of performance, the same query will take about 30 seconds to complete if we have ten times more records. I doubt that your users will be willing to wait for that long!

If you would like to empower your end users to generate their own reports in ad-hoc fashion, they have to know quite a bit about the relational (ER) model and SQL. They have to know which tables to join and they have to know how to join them. True, ad-hoc reporting tools may abstract to a certain extent the technicalities of the relational model but they have issues of their own. Finally, standard reports are not interactive. The user cannot drill down data, e.g. double-click on a given year column to see data broken down by quarters.

Deploying the Unified Dimensional Model

Now, let's see how OLAP and UDM change the reporting experience. We will use Excel reporting capabilities to build a simple reporting solution with SSAS 2005 that will resemble the report shown in Figure 1.1. The report will source its data from an SSAS 2005 cube. To build the cube, we will use the AdventureWorks Analysis Services Project sample that comes with the SQL Server 2005 samples. It includes a sample cube called *Adventure Works*. The Adventure Works cube draws data from the Adventure Works warehouse database (*AdventureWorksDW*)

If you have installed the SQL Server 2005 samples (see Appendix A), the project will be located in C:\Program Files\Microsoft SQL Server\90\Tools\Samples\AdventureWorks Analysis Services Project folder.

Note Both Standard and Enterprise versions of the project will do fine for our demo. Choose one based on the SSAS version you are running.

Opening an Analysis Services project in BI Studio

If you haven't deployed the sample AdventureWorks Analysis Services Project sample, follow these steps:

1. Start SQL Server Business Intelligence Development Studio (found in the Microsoft SQL Server 2005 program group). Readers familiar with Visual Studio.NET will undoubtedly notice that the Business Intelligence Development Studio IDE looks similar. As I've mentioned in section 1.1.4, you will use BI Studio as a primary tool to design and maintain UDM.
2. From the File menu, choose Open, then Project/Solution… and Open the Adventure Works solution (Adventure Works.sln). This solution includes a single project (Adventure Works DW.dwproj). Don't worry if the concepts of SSAS database and projects are not immediately obvious. It will all become clear in Chapter 2. For the time being, note that the AdventureWorks DW project includes the definitions of all objects in the Adventure Works UDM.

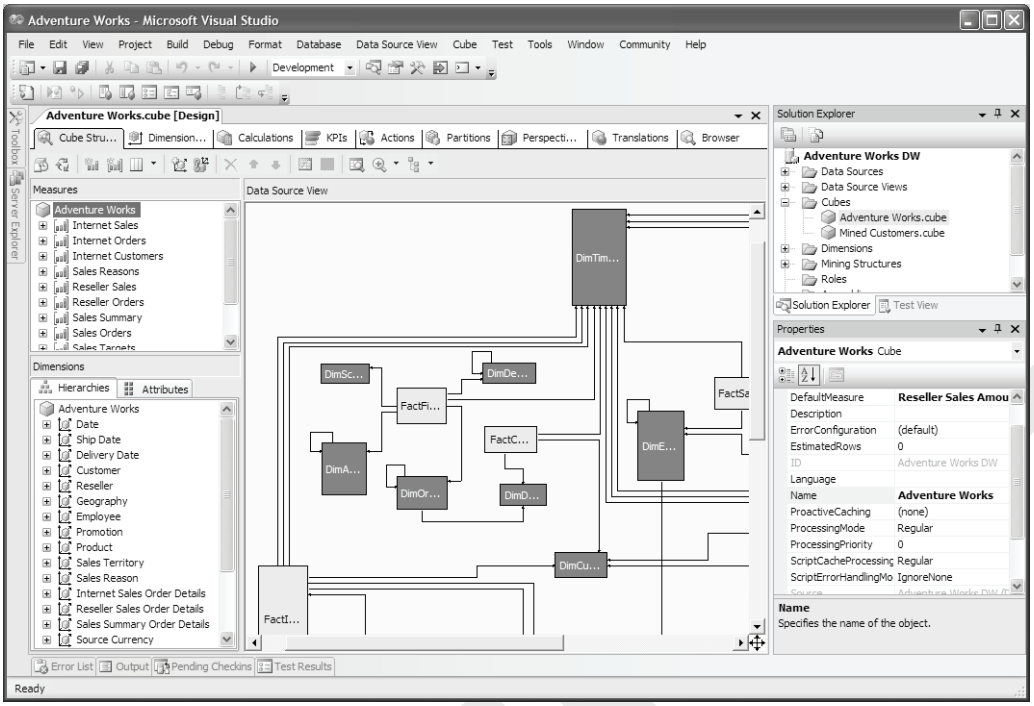


Figure 1.21 Use the Business Intelligence Studio to design SSAS objects.

3. If the Solution Explorer window is not shown, click on the Solution Explorer (View menu) or press Ctrl-Alt-L. The Solution Explorer shows the SSAS objects defined in the Adventure Works DW project in a tree view, as shown in Figure 1.21.

   Double-click on the Adventure Works cube to open the Cube Designer. The Cube Designer uses the same colors (blue for dimension tables and yellow for fact tables) as Analysis Manager 2000. Note that the dimension and fact tables are linked to each other, just like relational tables are joined via referential integrity constraints. However, the big difference is that UDM enforces these relationships at a metadata level. As a result, the end user doesn't have to explicitly join UDM objects. Instead, producing an OLAP report is as easy as dragging and dropping UDM objects using your favorite OLAP client, which could be Microsoft Excel, as we will demonstrate shortly.

4. In the Solution Explorer, expand the Cube node, right-click on the Adventure Works cube and choose View Code. BI studio shows the definition of the cube described in Analysis Services Scripting Language (ASSL). When you work in project mode (default), changes are persisted locally.

Deploying projects

5. To propagate changes made in project mode, you need to deploy the project. Back to the Solution Explorer, right-click on the Adventure Works DW project node (not to be confused with the topmost solution node) and choose *Properties*. Expand the Configuration Properties and click the *Deployment* node (Figure 1.22).

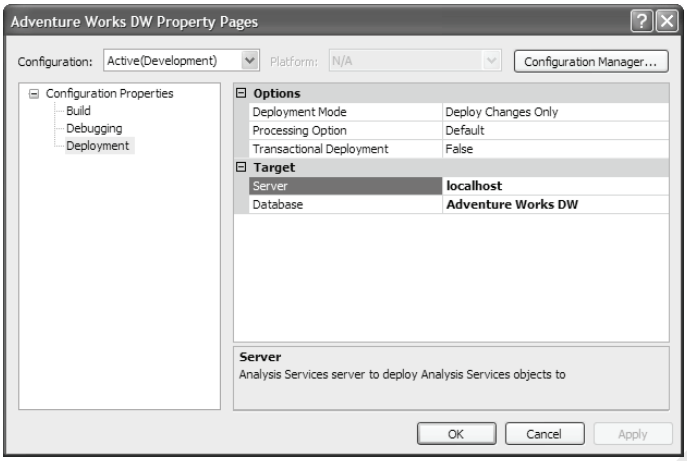


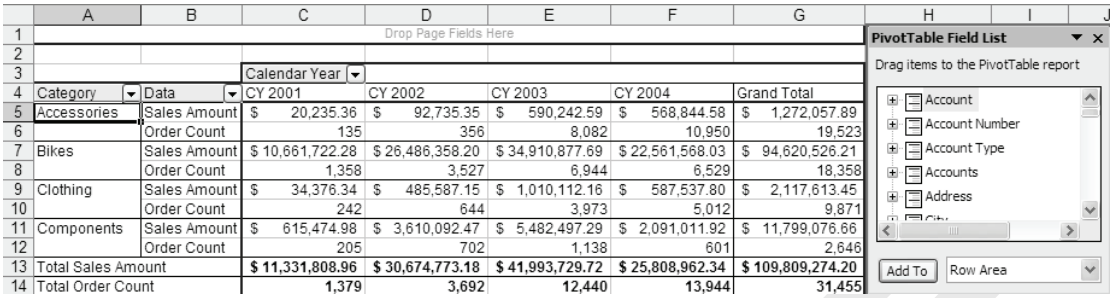
Figure 1.22 To propagate changes made in project mode, you need to deploy the project to the SSAS server specified in the Deployment Options window.

6. Verify the server name (enter **localhost** to deploy to the local server).
7. Close the Property Pages window. If you haven't deployed the Adventure Works DW project yet, right-click on the project node and choose *Deploy*. BI Studio builds and deploys the project. The end result of this process will be the creation of a new SSAS database called *Adventure Works DW*.
8. To verify that the deployment process has completed successfully, open SQL Server Management Studio (Microsoft SQL Server 2005 Program group).
9. In the Object Explorer pane, choose *Connect* ⇨ *Analysis Services* to connect to the SSAS server that you deployed the project to.
10. Expand the Databases folder and check that there is a database named *Adventure Works DW*.
11. Expand the Adventure Works DW folder and take some time to familiarize yourself with the database content. For example, expand the Adventure Works cube, then the Measure Groups measures and notice that there are eleven measure groups (with the Enterprise version).

Building OLAP Report

At this point, the SSAS database is built and its only cube has been processed. Let's now use Microsoft Excel as a reporting tool to browse the cube data. We will generate a report that shows sales data broken by product and time, as shown in Figure 1.23.

1. Start Microsoft Excel 2003
2. Create a new PivotTable report by selecting PivotTable and PivotChart Report from the Data menu.
3. In Step 1 of the PivotTable and PivotChart Report Wizard, select the "External data source" option since you will be retrieving the data from an SSAS server. Click Next.
4. In Step 2, click on the Get Data button to configure Microsoft Query. In the Choose Data Source dialog, click on the OLAP Cubes tab. Make sure the New Data Source item is selected. Click OK.



	A	B	C	D	E	F	G
1	Drop Page Fields Here						
2							
3			Calendar Year				
4	Category	Data	CY 2001	CY 2002	CY 2003	CY 2004	Grand Total
5	Accessories	Sales Amount	$ 20,235.36	$ 92,735.35	$ 590,242.59	$ 568,844.58	$ 1,272,057.89
6		Order Count	135	356	8,082	10,950	19,523
7	Bikes	Sales Amount	$ 10,661,722.28	$ 26,486,358.20	$ 34,910,877.69	$ 22,561,568.03	$ 94,620,526.21
8		Order Count	1,358	3,527	6,944	6,529	18,358
9	Clothing	Sales Amount	$ 34,376.34	$ 485,587.15	$ 1,010,112.16	$ 587,537.80	$ 2,117,613.45
10		Order Count	242	644	3,973	5,012	9,871
11	Components	Sales Amount	$ 615,474.98	$ 3,610,092.47	$ 5,482,497.29	$ 2,091,011.92	$ 11,799,076.66
12		Order Count	205	702	1,138	601	2,646
13	Total Sales Amount		$ 11,331,808.96	$ 30,674,773.18	$ 41,993,729.72	$ 25,808,962.34	$ 109,809,274.20
14	Total Order Count		1,379	3,692	12,440	13,944	31,455

Figure 1.23 Use Microsoft Excel to create SSAS 2005 interactive reports.

5. In the *Create New Data Source* dialog, name the data source Adventure Works. In the "Select an OLAP provider for the database you want to access" dropdown select Microsoft OLE DB Provider for Analysis Services 9.0 (see Figure 1.24). Recall that Win32 clients must use version 9.0 of the OLE DB Provider for Analysis Services to connect to SSAS 2005.

6. Click the Connect button. On the *MultiDimensional Connection* dialog, select the *Analysis Server* radio button, enter the machine name where SSAS 2005 is installed (Server field). Leave the credentials fields blank to use Windows Authentication. Click Next. The Database listbox appear. Select the *Adventure Works DW* database and click Finish. You are now taken back to the Create New Data Source dialog.

7. Expand the last dropdown (Figure 1.24) and select the Adventure Works cube. Click OK to close the *Create New Data Source* dialog and OK to close the *Choose Data Source* dialog. You are back to the PivotTable and PivotChart Wizard. Click Next to advance to Step 3.

8. Accept the defaults in Step 3 and click Finish.

9. A blank pivot report appears in the Excel spreadsheet. A PivotTable Field List pane contains all measures and dimensions defined in the Adventure Works cube.

Note At this point, you are probably confused by the sheer number of items shown in the PivotTable Field List pane. Most of the items are attribute-based dimensions which are derived directly from columns in the underlying dimension tables. For example, the *Color* dimension corresponds to the Color column in the DimProduct dimension table. Unfortunately, Excel 2003 was released before SSAS 2005 and it is unaware of the new features. Subsequently, the Field List is not capable of organizing the attribute hierarchies in folders, as the Cube Browser does.

10. Scroll down the PivotTable Field List pane until you locate the *Date.Calendar* hierarchy. This dimension represents a natural time hierarchy with Year, Semester, Quarter, Month and Date levels. Drag the *Date.Calendar* hierarchy to the *Drop Column Fields Here* area of the pivot report.

11. Scroll further down the PivotTable Field List pane until you locate the *Product Categories* dimension. This dimension represents a natural product hierarchy with Category, Subcategory, and Product Name levels. Drag the *Product Categories* hierarchy to the *Drop Row Fields Here* area of the pivot report.

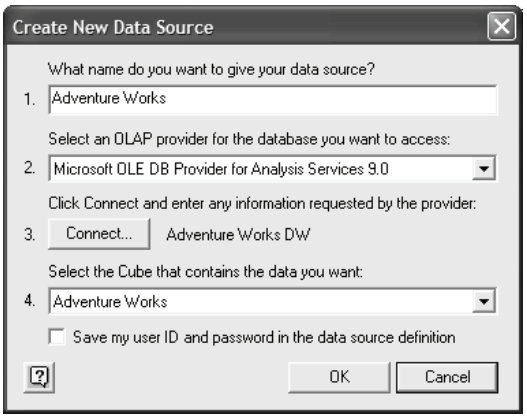


Figure 1.24 Use the Microsoft OLE DB Provider for Analysis Services 9.0 to connect to SSAS 2005.

12. Let's now add some measures to the report. Scroll the PivotTable Field List pane all the way down until you locate the *Sales Amount* measure. In the pane, measures have a different icon (0110) than dimensions. Drag the *Sales Amount* measure to the *Drop Data Items Here* report area. Do the same with the *Order Count* measure. Although the PivotTable Field lists doesn't have a special icon for MDX expressions, note that there are many calculated measures we can use in the report, such as *Reseller Ratio to All Products*, *Internet Gross Profit Margin*, etc.

13. If you wish, you can spend some time to pretty up the report by changing format, font, and color settings. At the end, your report may look like the one shown in Figure 1.23.

We are done! Feel free to experiment with the Excel PivotTable report. For example, double-click on any member of the Product Categories dimension to drill down sales data to the product subcategory and product name levels. Drag and drop other dimensions and measures. Once the cube is designed and deployed, there are many ways to build interactive reports that provide the needed level of business intelligence.

1.6 Summary

This chapter has been a whirlwind tour of the SSAS 2005 and OLAP technology. By now, you should view SSAS 2005 as a sophisticated server-based platform that provides OLAP and data mining services. Empowered with SSAS 2005, you can build intuitive and efficient BI applications. We've seen how SSAS 2005 fits into the Microsoft BI initiative. We've emphasized the ambitious goal of SSAS 2005 to converge the relational and dimensional models into a single Unified Dimensional Model.

We've also looked at the high-level of the SSAS 2005 architecture and emphasized the fact that XMLA is the native protocol of SSAS 2005. To help readers who have prior SSAS experience, I've provided a side-by-side comparison map between versions 2000 and 2005. Finally, we've put into practice what we've learned by building an interactive Microsoft Excel PivotTable report which sourced its data from SSAS 2005. Having laid the SSAS foundation, we are ready to "drill down" the UDM layers. Let's start by finding out how we can work with data.

1.7 Resources

Microsoft SSAS home page
(http://shrinkster.com/895) – First stop for the latest on SSAS.

The OLAP Report website
(http://www.olapreport.com/) – The OLAP Report is an independent research resource for organizations buying and implementing OLAP applications.

SQL Server 2005 Features Comparison
(http://shrinkster.com/62q) – Compares side by side the editions of the SQL Server 2005 products.

SSAS and the competition
(http://www.olapreport.com/market.html) – Market share analysis of the top OLAP vendors.

Microsoft Reporting Services in Action Book
(http://www.manning.com/lachev) – Following the report lifecycle, my book teaches you the necessary skills to create, manage, and deliver SSRS reports.

XML for Analysis home page
(http://www.xmla.org/) – Visit to access the latest XML/A specification, FAQ, discussion forum, and samples.