1

Introduction to Data Warehousing and Analysis Services 2005

A data warehouse is a system that takes data from a company's databases and other data sources and transforms it into a structure conducive to business analysis. Mathematical operations are often performed on the newly structured or organized data to further its usefulness for making business decisions. Finally, the data is made available to the end user for querying and analysis. If the data warehouse is well architected then queries to the data warehouse will return query results quickly (in a matter of seconds). The business decision-maker will have a powerful tool that could never have been effectively used directly from the company's daily operational systems. We consider data analysis to be of two forms. The first requires a person to investigate the data for trends. This method is called On Line Analytical Processing (OLAP). The second form utilizes algorithms to scour the data looking for trends. This method is called Data Mining. Analysis Services 2005 is a business intelligence platform that enables you to use OLAP and Data Mining. Now that you have the big picture of data warehousing, let us look at what you will learn in this chapter.

In this chapter you learn what data warehousing really is and how it relates to business intelligence. This information comes wrapped in a whole load of new concepts, and you get a look at the best known approaches to warehousing with the introduction of those concepts. We explain data warehousing in several different ways and we are sure you will understand it. You will finally see how Analysis Services 2005 puts it all together in terms of architecture — at both client and server levels — based on a new data abstraction layer called Unified Dimensional Model (UDM).

A Closer Look at Data Warehousing

In the book *Building the Data Warehouse*, Bill Inmon described the data warehouse as "a *subject oriented*, *integrated*, *non-volatile*, and *time variant* collection of data in support of management's

decisions." According to Inmon, the subject orientation of a data warehouse differs from the operational orientation seen in *On-Line Transaction Processing* (OLTP) systems; so a subject seen in a data warehouse might relate to customers, whereas an operation in an OLTP system might relate to a specific application like sales processing and all that goes with it.

The word *integrated* means that throughout the enterprise, data points should be defined consistently or there should be some integration methodology to force consistency at the data warehouse level. One example would be how to represent the entity Microsoft. If Microsoft were represented in different databases as MSFT, MS, Microsoft, and MSoft, it would be difficult to meaningfully merge these in a data warehouse. The best-case solution is to have all databases in the enterprise refer to Microsoft as, say, MSFT, thereby making the merger of this data seamless. A less desirable, but equally workable, solution is to force all the variants into one during the process of moving data from the operational system to the data warehouse.

A data warehouse is referred to as non-volatile since it differs from operational systems, which are often transactional in nature and updated regularly. The data warehouse is generally loaded at some preset interval, which may be measured in weeks or even months. This is not to say it is never measured in days; but even if updates do occur daily, that is still a sparse schedule compared to the constant changes being made to transactional systems.

The final element in this definition regards time variance, which is a sophisticated way of saying how far back the stored data in the system reaches. In the case of operational systems, the time period is quite short, perhaps days, weeks, or months. In the case of the warehouse, it is quite long — typically on the order of years. This last item might strike you as fairly self-evident because you would have a hard time analyzing business trends if your data didn't date back further than two months. So, there you have it, the classic definition that no good book on data warehousing should be without.

Taking the analysis one step closer to the nuts and bolts of working systems, consider that a relational database can be represented graphically as an *Entity-Relationship Diagram* (ERD) in a case tool or in SQL Server 2005 itself (see Figure 1-2 for an example). Not only will you see the objects in the database shown in the diagram, but you will also see many join connections which represent the relationships between the objects. Data warehouses can be formed from relational databases or multi-dimensional databases. When your data warehouse is modeled after the relational database model then data is stored in two-dimensional tables and analytical or business queries are normally very slow. When one refers to a data warehouse it is typically OLAP that is being referred to. In the case of OLAP you have a multi-dimensional database with data stored in such a way that business users can view it and efficiently answer business questions — all with fast query response times. There is more to come in this chapter on the differences between relational and OLAP databases.

Data warehousing is the process by which data starting from an OLTP database is transformed and stored so as to facilitate the extraction of business-relevant information from the source data. An OLTP database, like a point-of-sale (POS) database is transaction-based and typically normalized (well optimized for storage) to reduce the amount of redundant data storage generated. The result makes for fast updates, but this speed of update capability is offset by a reduction in speed of information retrieval at query time. For speed of information retrieval, especially for the purpose of business analytics, an OLAP database is called for. An OLAP database is highly denormalized (not well optimized for storage) and therefore has rows of data that may be redundant. This makes for very fast query responses because relatively few joins are involved. And fast responses are what you want while doing business intelligence

work. Figure 1-1 shows information extracted from transactional databases and consolidated into multi-dimensional databases; then stored in data marts or data warehouses. Data marts can be thought of as mini–data warehouses and quite often act as part of a larger warehouse. Data marts are subject-oriented data stores for well-manicured (cleaned) data. Examples include a sales data mart, an inventory data mart, or basically any subject rooted at the departmental level. A data warehouse on the other hand, functions at the enterprise level and typically handles data across the entire organization.

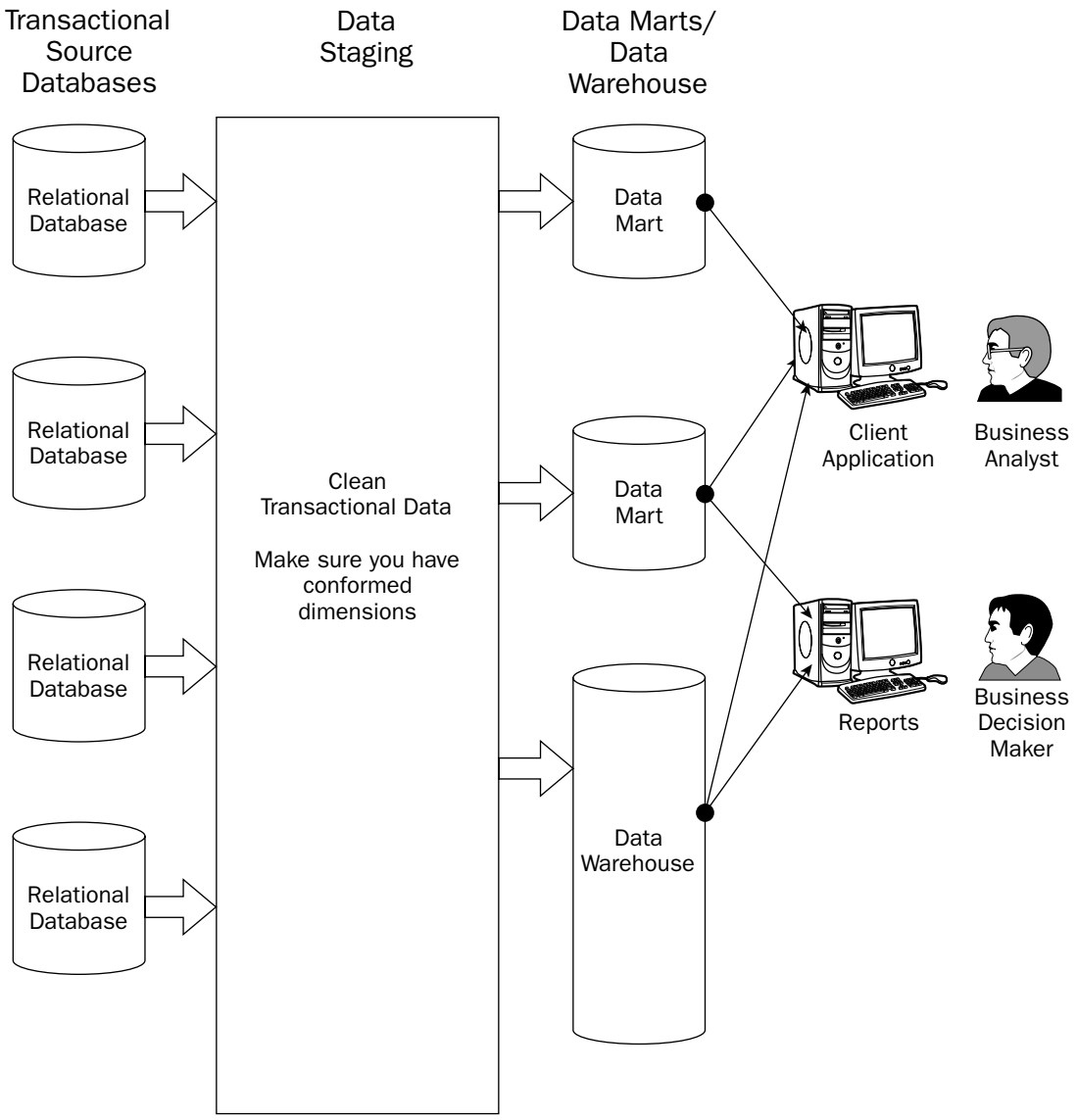

Figure 1-1

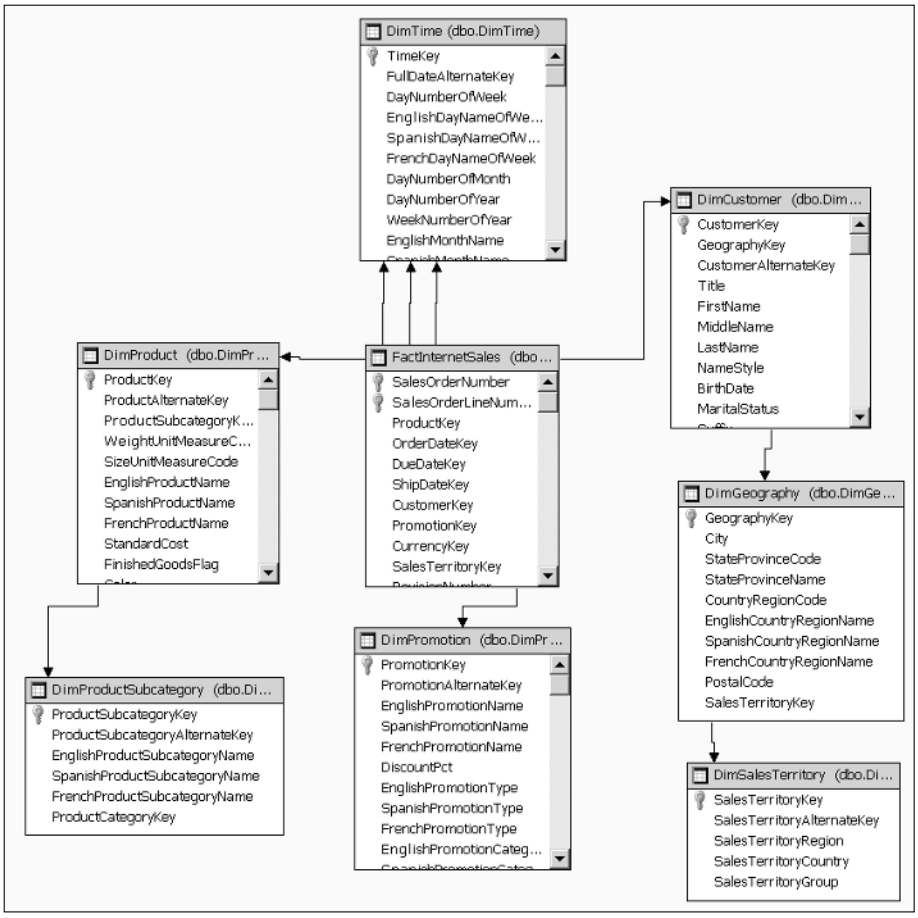

Figure 1-2

Key Elements of a Data Warehouse

Learning the elements of a data warehouse or data mart is, in part, about building a new vocabulary; the vocabulary associated with data warehousing can be less than intuitive, but once you get it, it all makes sense. The challenge, of course, is understanding it in the first place. Two kinds of tables form a data warehouse: fact tables and dimension tables.

Figure 1-3 shows a fact and a dimension table and the relationship between them. A fact table typically contains the business fact data such as sales amount, sales quantity, the number of customers, and the foreign keys to dimension tables. A *foreign key* is a field in a relational table that matches the primary key column of another table. Foreign keys provide a level of indirection between tables that enable you to cross-reference them. One important use of foreign keys is to maintain referential integrity (data integrity) within your database. Dimension tables contain detailed information relevant to specific attributes of the fact data, such as details of the product, customer attributes, store information, and so

on. In Figure 1-3, the dimension table Product contains the information Product SKU and Product Name. The following sections go into more detail about fact and dimension tables.

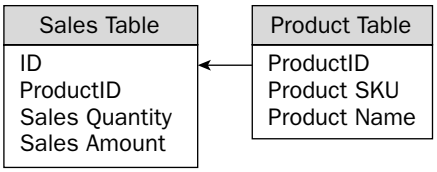

Figure 1-3

Fact Tables

With the end goal of extracting crucial business insights from your data, you will have to structure your data initially in such a way as to facilitate later numeric manipulation. Leaving the data embedded in some normalized database will never do! Your business data, often called detail data or fact data, goes in a de-normalized table called the fact table. Don't let the term "facts" throw you; it literally refers to the facts. In business, the facts are things such as number of products sold and amount received for products sold. Yet another way to describe this type of data is to call them *measures*. Calling the data measures versus detail data is not an important point. What is important is that this type of data is often numeric (though it could be of type string) and the values are quite often subject to aggregation (pre-calculating roll-ups of data over hierarchies, which subsequently yield improved query results). A fact table often contains columns like the ones shown in the following table:

Product ID	Date ID	State ID	Number of Cases	Sales Amount
1	07/01/2005	6	3244	$90,842
1	07/01/2005	33	6439	$184,000
1	07/01/2005	42	4784	$98,399
1	08/01/2005	31	6784	$176,384
1	08/01/2005	6	2097	$59,136
1	08/01/2005	33	7326	$8,635
1	08/01/2005	42	4925	$100,962
1	09/01/2005	31	8548	$176,384
1	09/01/2005	6	945	$26,649
1	09/01/2005	33	8635	$246,961
1	09/01/2005	42	4935	$101,165
1	10/01/2005	31	9284	$257,631
1	10/01/2005	33	9754	$278,965
1	10/01/2005	42	4987	$102,733
...	...	...	...	...

This table shows the sales of different varieties of beer between the months of July and October 2005 in four different states. The product id, date id, and state ids together form the primary key of the fact table. The number of cases of beer sold and the sales amount are facts. The product id, date id, and state id are foreign keys that join to the products, date, and state tables. In this table the state ids 6, 31, 33, and 42 refer to the states MA, CA, OR, and WA, respectively, and represent the order in which these states joined the United States. Building the fact table is an important step towards building your data warehouse.

Dimension Tables

The fact table typically holds quantitative data; for example, transaction data that shows number of units sold per sale and amount charged to the customer for the unit sold. To provide reference to higher-level roll-ups based on things like time, a complementary table can be added that provides linkage to those higher levels through the magic of the join (how you link one table to another). In the case of time, the fact table might only show the date on which some number of cases of beer was sold; to do business analysis at the monthly, quarterly, or yearly level, a time dimension is required. The following table shows what a beer products dimension table would minimally contain. The product id is the primary key in this table. The product id of the fact table shown previously is a foreign key that joins to the product id in the following table:

Product ID	Product SKU	Product Name
1	SBF767	SuperMicro Ale
2	SBH543	SuperMicro Lager
3	SBZ136	SuperMicro Pilsner
4	SBK345	SuperMicro Hefeweizen
...	...	...

For illustrative purposes, assume that you have a dimension table for time that contains monthly, quarterly, and yearly values. There must be a unique key for each value; these unique key values are called *primary keys.* Meanwhile, back in the fact table you have a column of keys with values mapping to the primary keys in the dimension table. These keys in the fact table are called *foreign keys*. For now it is enough if you get the idea that dimension tables connect to fact tables and this connectivity provides you with the ability to extend the usefulness of your low-level facts resident in the fact table.

A multi-dimensional database is created from fact and dimension tables to form objects called dimensions and cubes. Dimensions are objects that are created mostly from dimension tables. Some examples of dimensions are time, geography, and employee which would typically contain additional information about those objects by which users can analyze the fact data. The cube is an object that contains fact data as well as dimensions so that data analysis can be performed by slicing or dicing dimensions. For example, you could view the sales information for the year 2005 in the state of Washington. Each of those slices of information is a dimension.

Dimensions

To make sense of a cube, which is at the heart of business analysis and discussed in the next section, you must first understand the nature of dimensions. We say that OLAP is based on multidimensional

databases because it quite literally is. You do business analysis by observing the relationships between dimensions like Time, Sales, Products, Customers, Employees, Geography, and Accounts. Dimensions are most often made up of several hierarchies. Hierarchies are logical entities by which a business user might want to analyze fact data. Each hierarchy can have one or more levels. A hierarchy in the geography dimension, for example, might have the following levels: Country, State, County, and City.

A hierarchy like the one in the geography dimension would provide a completely balanced hierarchy for the United States. *Completely balanced hierarchy* means that all leaf (end) nodes for cities would be an equal distance from the top level. Some hierarchies in dimensions can have an unbalanced distribution of leaf nodes relative to the top level. Such hierarchies are called *unbalanced hierarchies*. An organization chart is an obvious example of an unbalanced hierarchy. There are different depths to the chain of supervisor to employee; that is, the leaf nodes are different distances from the top-level node. For example, a general manager might have unit managers and an administrative assistant. A unit manager might have additional direct reports such as a dev and a test manager, while the administrative assistant would not have any direct reports. Some hierarchies are typically balanced but are missing a unique characteristic of some members in a level. Such hierarchies are called *ragged hierarchies*. An example of a ragged hierarchy is a geography hierarchy that contains the levels Country, State, and City. Within the Country USA you have State Washington and City Seattle. If you were to add the Country Greece and City Athens to this hierarchy, you would add them to the Country and City levels. However, there are no states in the Country Greece and hence member Athens is directly related to the Country Greece. A hierarchy in which the members descend to members in the lowest level with different paths is referred to as a ragged hierarchy. Figure 1-4 shows an example of a Time dimension with the hierarchy Time. In this example, Year, Quarter, Month, and Date are the levels of the hierarchy. The values 2005 and 2006 are members of the Year level. When a particular level is expanded (indicated by minus sign in the figure) you can see the members of the next level in the hierarchy chain.

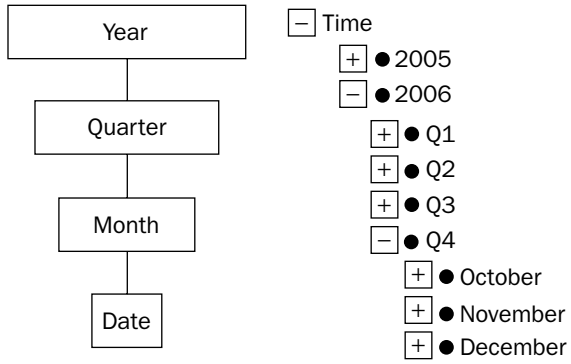

Figure 1-4

To sum up, a dimension is a hierarchical structure that has levels that may or may not be balanced. It has a subject matter of interest and is used as the basis for detailed business analysis.

Cubes

The *cube* is a multidimensional data structure from which you can query for business information. You build cubes out of your fact data and the dimensions. A cube can contain fact data from one or more fact

tables and often contains a few dimensions. Any given cube usually has a dominant subject under analysis associated with it. For example, you might build a Sales cube with which you analyze sales by region, or a Call Processing cube with which you analyze length of call by problem category reported. These cubes are what you will be making available to your users for analysis.

Figure 1-5 shows a Beer Sales cube that was created from the fact table data shown previously. Consider the front face of the cube that shows numbers. This cube has three dimensions: Time, Product Line, and State where the product was sold. Each block of the cube is called a *cell* and is uniquely identified by a member in each dimension. For example, analyze the bottom-left corner cell that has the values 4,784 and $98,399. The values indicate the number of sales and the sales amount. This cell refers to the sales of Beer type Ale in the state of Washington (WA) for July 2005. This is represented as [WA, Ale, Jul '05]. Notice that some cells do not have any value; this is because no facts are available for those cells in the fact table.

The whole point of making these cubes involves reducing the query response time for the information worker to extract knowledge from the data. To make that happen, cubes typically contain pre-calculated summary data called *aggregations*. Querying existing aggregated data is close to instantaneous compared to doing cold (no cache) queries with no pre-calculated summaries in place. This is really at the heart of business intelligence, the ability to query data with possibly gigabytes or terabytes of pre-summarized data behind it and yet get an instant response from the server. It is quite the thrill when you realize you have accomplished this feat!

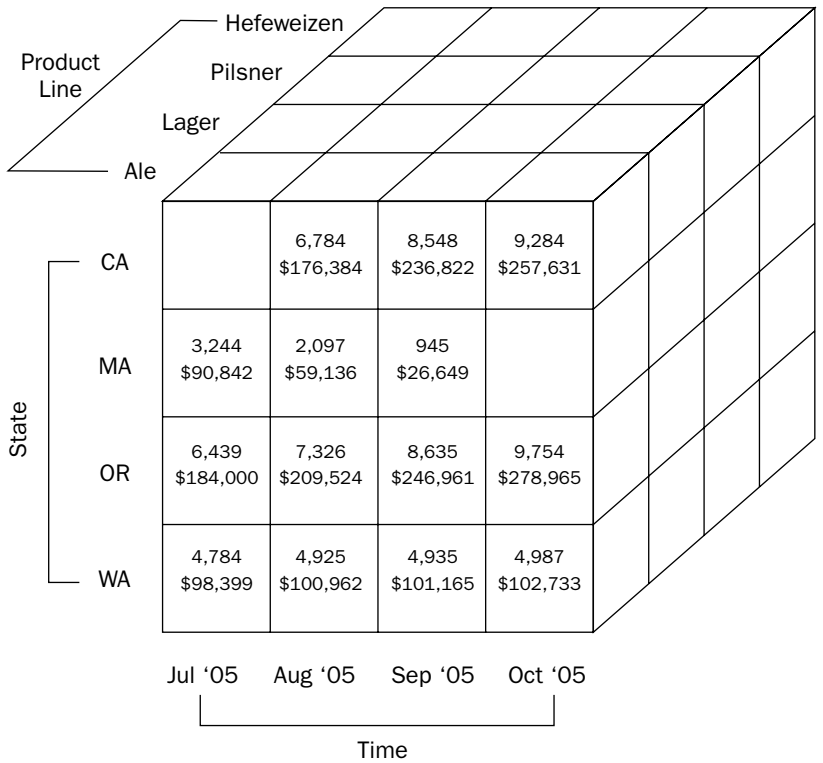

Figure 1-5

You learned about how cubes provide the infrastructure for storing multidimensional data. Well, it doesn't just store multidimensional data from fact tables; it also stores something called *aggregations* of that data. A typical aggregation would be the summing of values up a hierarchy of a dimension. For example, summing of sales figures up from stores level, to district level, to regional level; when querying for those numbers you would get an instant response because the calculations would have already been done when the aggregations were formed. The fact data does not necessarily need to be aggregated as sum of the specific fact data. You can have other ways of aggregating the data such as counting the number of products sold. Again, this count would typically roll up through the hierarchy of a dimension.

The Star Schema

The entity relationship diagram representation of a relational database shows you a different animal altogether as compared to the OLAP (multidimensional) database. It is so different in fact, that there is a name for the types of schemas used to build OLAP databases: the star schema and the snowflake schema. The latter is largely a variation on the first. The main point of difference is the complexity of the schema; the OLTP schema tends to be dramatically more complex than the OLAP schema. Now that you know the infrastructure that goes into forming fact tables, dimension tables, and cubes, the concept of a star schema should offer little resistance. That is because when you configure a fact table with foreign key relationships to one or more of a dimension table's primary keys, as shown in Figure 1-6, you have a star schema. Looks a little like a star, right?

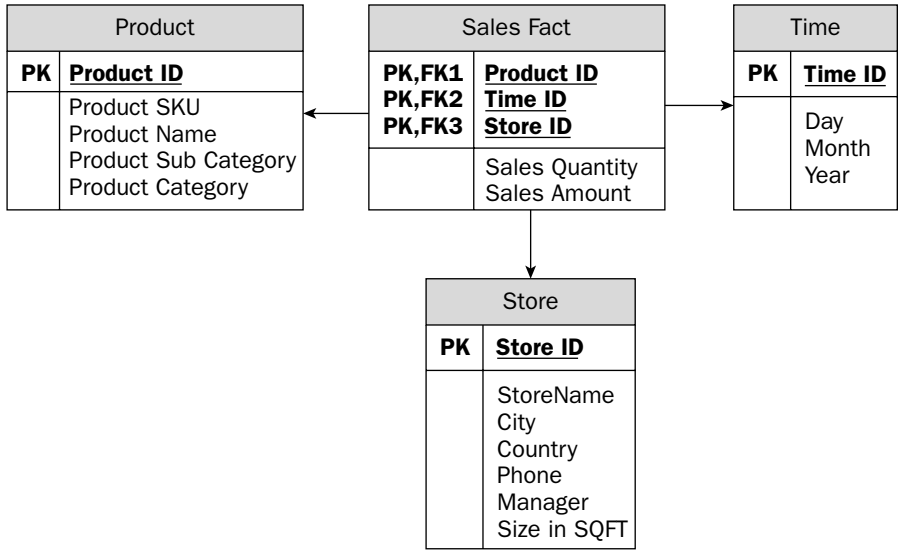

Figure 1-6

The star schema provides you with an illustration of the relationships between business entities in a clear and easy-to-understand fashion. Further, it enables number crunching of the measures in the fact table to progress at amazing speeds.

The Snowflake Schema

If you think the star schema is nifty, and it is, there is an extension of the concept called the snowflake schema. The snowflake schema is useful when one of your dimension tables starts looking as detailed as the fact table it is connected to. With the snowflake, a level is forked off from one of the dimension tables, so it is separated by one or more tables from the fact table. In Figure 1-7 the Product dimension has yielded a Product Category level. The Product Sub Category level is hence one table removed from the sales fact table. In turn, the Product Sub Category level yields a final level called the Product Category — which has two tables of separation between it and the sales fact table. These levels, which can be used to form a hierarchy in the dimension, do not make for faster processing or query response times, but they can keep a schema sensible.

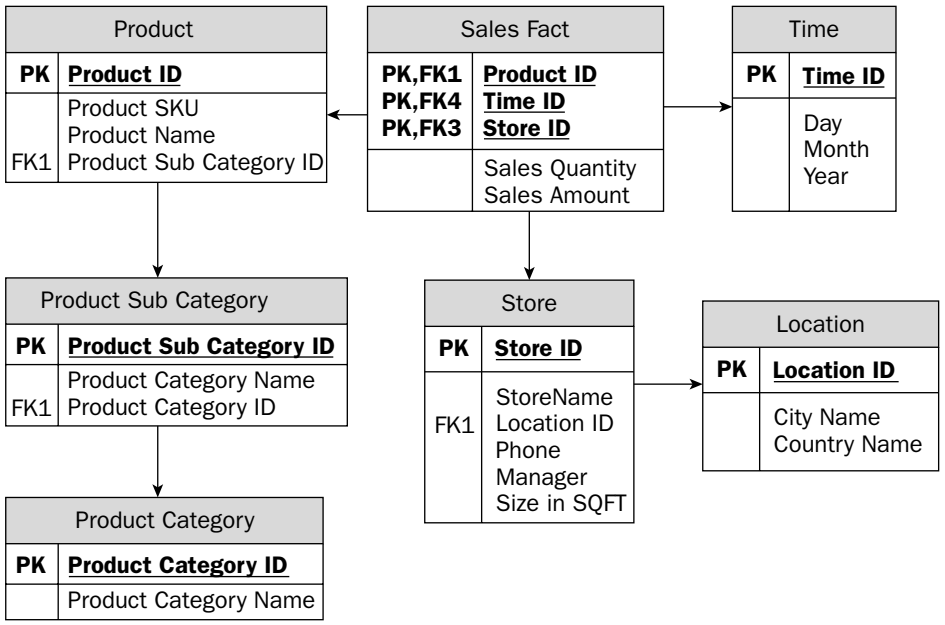

Figure 1-7

You have so far learned the fundamental elements of a data warehouse. The biggest challenge is to understand these well and design and implement your data warehouse to cater to your end-users. There are two main design techniques for implementing data warehouses. These are the Inmon approach and the Kimball approach.

Inmon Versus Kimball Different Approaches

In data warehousing there are two commonly acknowledged approaches to building a decision support infrastructure, and both can be implemented using the tools available in SQL Server 2005 with Analysis Services 2005. It is worth understanding these two approaches and the often-cited difference of views

that result. These views are expressed most overtly in two seminal works: *The Data Warehouse Lifecycle Toolkit* by Ralph Kimball, Laura Reeves, Margy Ross, and Warren Thornthwaite, and *Corporate Information Factory* by Bill Inmon, Claudia Imhoff, and Ryan Sousa.

Kimball identified early on the problem of the stovepipe. A stovepipe is what you get when several independent systems in the enterprise go about identifying and storing data in different ways. Trying to connect these systems or use their data in a warehouse results in something resembling a Rube-Goldberg device. To address this problem, Kimball advocates the use of conformed dimensions. *Conformed* refers to the idea that dimensions of interest — sales, for example — should have the same attributes and rollups (covered in the "Aggregations" section earlier in this chapter) in one data mart as another. Or at least one should be a subset of the other. In this way, a warehouse can be formed from data marts. The real gist of Kimball's approach is that the data warehouse contains dimensional databases for ease of analysis and that the user queries the warehouse directly.

The Inmon approach has the warehouse laid out in third normal form (not dimensional) and the users query data marts, not the warehouse. In this approach the data marts are dimensional in nature. However, they may or may not have conformed dimensions in the sense Kimball talks about.

Happily it is not necessary to become a card-carrying member of either school of thought in order to do work in this field. In fact, this book is not strictly aligned to either approach. What you will find as you work through this book is that by using the product in the ways in which it was meant to be used and are shown here, certain best practices and effective methodologies will naturally emerge.

Business Intelligence Is Data Analysis

Having designed a data warehouse the next step is to understand and make business decisions from your data warehouse. Business intelligence is nothing but analyzing your data. An example of business analytics is shown through the analysis of results from a product placed on sale at a discounted price, as commonly seen in any retail store. If a product is put on sale for a special discounted price, there is an expected outcome: increased sales volume. This is often the case, but whether or not it worked in the company's favor isn't obvious. That is where business analytics come into play. We can use Analysis Services 2005 to find out if the net effect of the special sale was to sell more product units. Suppose you are selling organic honey from genetically unaltered bees; you put the 8-ounce jars on special — two for one — and leave the 10- and 12-ounce jars at regular price. At the end of the special you can calculate the *lift* provided by the special sale — the difference in total sales between a week of sales with no special versus a week of sales with the special. How is it you could sell more 8-ounce jars on special that week, yet realize no lift? It's simple — the customers stopped buying your 10- and 12-ounce jars in favor of the two-for-one deal; and you didn't attract enough new business to cover the difference for a net increase in sales.

You can surface that information using Analysis Services 2005 by creating a Sales cube that has three dimensions: Product, Promotion, and Time. For the sake of simplicity, assume you have only three product sizes for the organic honey (8-ounce, 10-ounce, and 12-ounce) and two promotion states ("no promotion" and a "two-for-one promotion for the 8-ounce jars"). Further, assume the Time dimension contains different levels for Year, Month, Week, and Day. The cube itself contains two measures, "count of products sold" and the "sales amount." By analyzing the sales results each week across the three product sizes you could easily find out that there was an increase in the count of 8-ounce jars of honey sold, but perhaps the total sales across all sizes did not increase due to the promotion. By slicing on the Promotion dimension you would be able to confirm that there was a promotion during the week that caused an

increase in number of 8-ounce jars sold. When looking at the comparison of total sales for that week (promotion week) to the earlier (non-promotion) weeks, lift or lack of lift is seen quite clearly. Business analytics are often easier described than implemented, however.

Analysis Services 2005

Analysis Services 2005 is part of Microsoft's product SQL Server 2005. SQL Server 2005 is the latest SQL Server release from Microsoft in November of 2005. In addition to Analysis Services 2005, SQL Server 2005 contains other services such as Integrations Services, Reporting Services, and Notification Services among other things. Integration Services, Analysis Services, and Reporting Services together form the core of business intelligence platform with SQL Server as the backend. Analysis Services 2005 not only provides you the ability to build dimensions and cubes for data analysis but also supports several data mining algorithms which can provide business insight into your data that are not intuitive. Analysis Services is part of a greater Business Intelligence platform, which leverages not only the rest of SQL Server 2005, but the .NET Framework (Common Language Runtime) and Visual Studio development environment as well. Next you will learn about the overall architecture of Analysis Services 2005 followed by the concept of Unified Dimensional Model (UDM) which helps you to have a unified view of your entire data warehouse.

SQL Server Analysis Services 2005 has been re-architected as both scalable and reliable enterprise class software that provides fine-grain security. So, not only is it quite manageable; but also protects your data from malicious attacks. The architecture of Analysis Services 2005 provides efficient scalability in terms of scale-out and scale-up features. Several instances of Analysis Services 2005 can be integrated together to provide an efficient scale-out solution. On the other hand, the service has been architected with efficient algorithms to handle large dimensions and cubes on a single instance. Analysis Services 2005 provides a rich set of tools for creating OLAP databases; efficient and easy manageability, as well as profiling capabilities.

The *Business Intelligence Development Studio* (BIDS) integrated within Visual Studio is the development tool shipped with Analysis Services 2005 used for creating and updating cubes, dimensions, and Data Mining models. The *SQL Server Management Studio* (SSMS) provides an integrated environment for managing SQL Server, Analysis Services, Integration Services, and Reporting Services. SQL Profiler in the SQL Server 2005 releases supports profiling Analysis Services 2005, which helps in analyzing the types of commands and queries sent from different users or clients to Analysis Services 2005. You learn more about BIDS and SSMS in Chapter 2 with the help of a tutorial. You learn about profiling an instance of Analysis Services using SQL Profiler in Chapter 12. In addition to the above-mentioned tools, Analysis Services 2005 provides two more tools: the Migration Wizard and the Deployment Wizard. The Migration Wizard helps in migrating Analysis Services 2000 databases to Analysis Services 2005. The Deployment Wizard helps in deploying the database files created using BIDS to Analysis Services 2005.

The SSMS provides efficient, enterprise-class manageability features for Analysis Services. Key aspects of an enterprise class service are availability and reliability. Analysis Services 2005 supports fail-over clustering on Windows clusters through an easy setup scheme and fail-over clustering certainly helps provide high availability. In addition, Analysis Services 2005 has the capability of efficiently recovering from failures. You can set up fine-grain security so that you can provide administrative access to an entire service or administrative access to specific databases, process permissions to specific databases, and read-only access to metadata and data. In addition to this, certain features are turned off by default so that the Service is protected from hacker attacks.

Analysis Services 2005 natively supports XML for Analysis specification defined by the XML/A Advisory Council. What this means is that the communication interface to Analysis Services from a client is XML. This facilitates ease of interoperability between different clients and Analysis Services 2005. The architecture of SQL Server Analysis Services 2005 includes various modes of communication to the service as shown in Figure 1-8. Analysis Server 2005 provides three main client connectivity components to communicate to the server. The Analysis Management Objects (AMO) is a new object model that helps you manage Analysis Server 2005 and the databases resident on it. The OLE DB 9.0 is the client connectivity component used to interact with analysis services 2005 instances s for queries that conforms to the OLE DB standard. The ADOMD.Net is dot Net object model support for querying data from Analysis Services 2005. In addition to the three main client connectivity components, two other components are provided by Analysis Services 2005. They are DSO 9.0 (Decision Support Object) and HTTP connectivity through a data pump. DSO 8.0 is the extension of the management object of Analysis Server 2000 so that legacy applications can interact with migrated Analysis Server 2000 databases on Analysis Server 2005. The data pump is a component that is set up with *IIS* (Internet Information System) to provide connection to Analysis Services 2005 over *HTTP* (Hypertext Transfer Protocol).

Even though XML/A helps in interoperability between different clients to Analysis Server, it comes with a cost on performance. If the responses from the server are large, transmission of XML data across the wire may take a long time depending on the type of network connection. Typically slow wide area networks might suffer from performance due to large XML responses. In order to combat this, Analysis Services 2005 supports the options for compression and binary XML so that the XML responses from the server could be reduced. These are optional features supported by Analysis Services 2005 that can be enabled or disabled on the Server.

Analysis Services 2005 stores metadata information of databases in the form of XML. Analysis Services 2005 provides you with the option of storing the data or aggregated data efficiently in a proprietary format on Analysis Services instance or storing them in the relational database. If you choose the data and/or aggregated data to be stored in the proprietary format you can expect better query performance than the case where the data is being retrieved from the relational database.. This proprietary format helps Analysis Services 2005 to retrieve the data efficiently and thereby improves the query performance. Based on where the data and/or aggregated fact data is stored you can classify the storage types as MOLAP (Multi-dimensional OLAP), ROLAP (Relational OLAP), or HOLAP (Hybrid OLAP).

MOLAP is the storage mode in which the data and aggregated data are both stored in proprietary format on the Analysis Services instance. This is the default and recommended storage mode for Analysis Services databases since you get better query performance as compared to the other storage types. The key advantages of this storage mode is fast data retrieval while analyzing sections of data and therefore provides good query performance and the ability to handle complex calculations. Two potential disadvantages of MOLAP mode are storage needed for large databases and the inability to see new data entering your data warehouse.

ROLAP is the storage mode in which the data is left in the relational database. Aggregated or summary data is also stored in the relational database. Queries against the Analysis Services are appropriately changed to queries to the relational database to retrieve the right section of data requested. The key advantage of this mode is that the ability to handle large cubes is limited by the relational backend only. The most important disadvantage of the ROLAP storage mode are slow query performance. You will encounter slower query performance in ROLAP mode due to the fact that each query to the Analysis Services is translated into one or more queries to the relational backend.

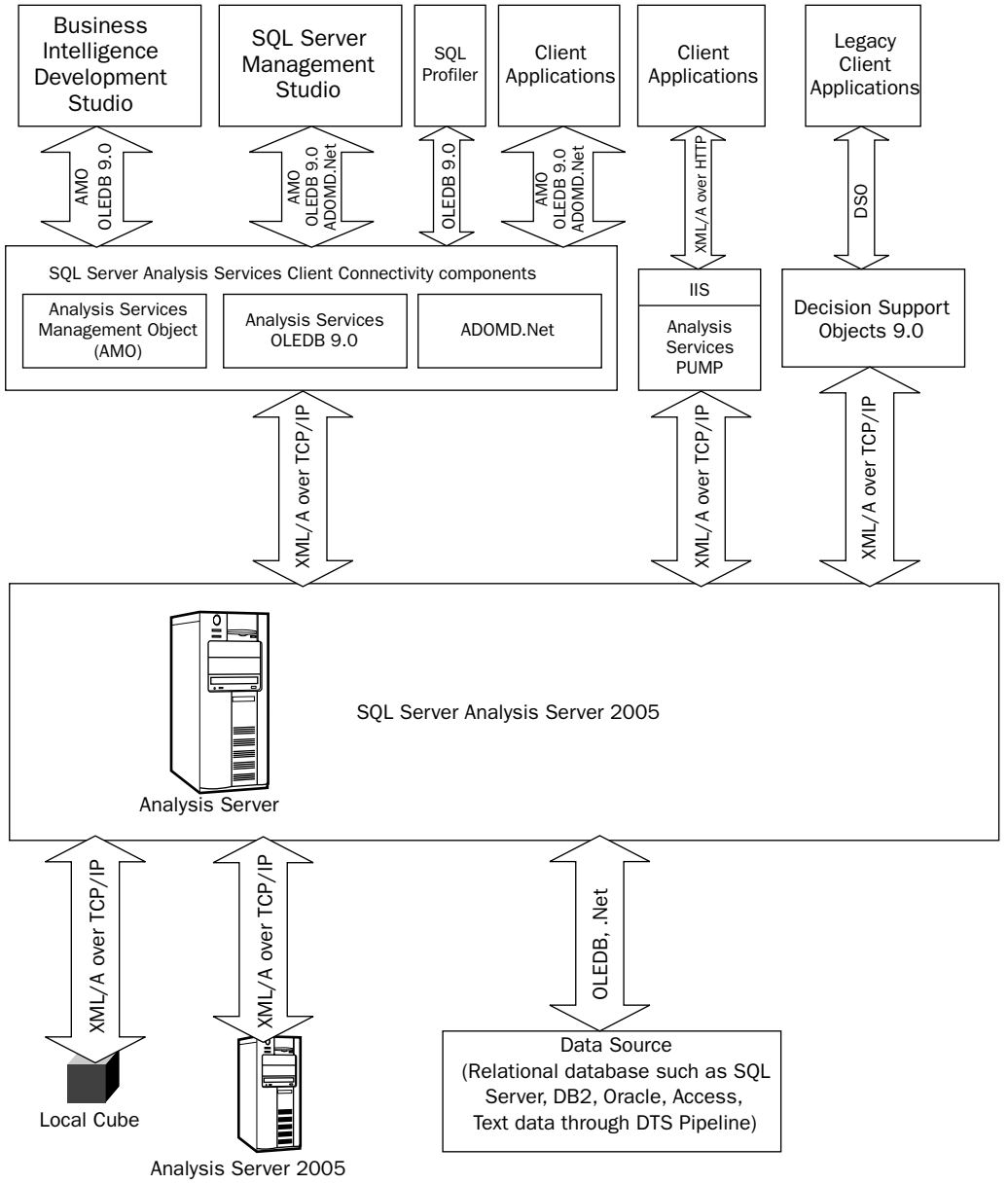

Figure 1-8

The HOLAP storage mode combines the best of MOLAP and ROLAP modes. The data in the relational database is not touched while the aggregated or summary data is stored on the Analysis Services instance in a proprietary format. If the queries to Analysis Services request aggregated data, they are retrieved from the summary data stored on the Analysis Services instance and they would be faster than data being retrieved from the relational backend. If the queries request detailed data, appropriate queries are sent to the relational backend and these queries can take a long time based on the relational backend.

Based on your requirements and maintainability costs you need to choose the storage mode that is appropriate for your business. Analysis Services 2005 supports all three storage modes.

The Unified Dimensional Model

Central to the architecture is the concept of the Unified Dimensional Model (UDM) which, by the way, is unique to this release of the product. UDM, as the name suggests, provides you with a way to encapsulate access to multiple heterogeneous data sources into a single model. In fact, with the UDM, you will be buffered from the difficulties previously presented by multiple data sources. Those difficulties were often associated with cross–data-source calculations and queries — so, do not be daunted by projects with lots of disparate data sources. The UDM can handle it! The UDM itself is more than a multiple data-source cube on steroids; it actually defines the relational schema upon which your cubes and dimensions are built. Think of the UDM as providing you with the best of the OLAP and relational worlds. UDM provides you with the rich metadata needed for analyzing and exploring data along with the functionality like the complex calculations and aggregations of the OLAP world. It supports complex schemas, and is capable of supporting ad-hoc queries that are needed for reporting in the relational world. Unlike the traditional OLAP world that allows you to define a single fact table within a cube, the UDM allows you to have multiple fact tables. The UDM is your friend and helps you have a single model that will support all your business needs. Figure 1-9 shows a UDM within Analysis Services 2005 that retrieves data from heterogeneous data sources and serves various types of clients.

Key elements of the UDM are as follows:

- **Heterogeneous data access support:** UDM helps you to integrate and encapsulate data from heterogeneous data sources. It helps you combine various schemas into a single unified model that gives end users the capability of sending queries to a single model.
- **Real-time data access with high performance:** The UDM provides end users with real-time data access. The UDM creates a MOLAP cache of the underlying data. Whenever there are changes in the underlying relational database, a new MOLAP cache is built. When users query the model, it provides the results from the MOLAP cache. During the time the cache is being built, results are retrieved from the relational database. UDM helps in providing real-time data access with the speed of an OLAP database due to the MOLAP cache. This feature is called proactive caching. You learn more about proactive caching in Chapter 18.
- **Rich metadata, ease of use for exploration, and navigation of data:** UDM provides a consolidated view of the underlying data sources with the richness of metadata provided by the OLAP world. Due to rich metadata supported by OLAP, end users are able to exploit this metadata to navigate and explore data in support of making business decisions. UDM also provides you with the ability to view specific sections of the unified model based on your business analysis needs.
- **Rich analytics support:** In addition to the rich metadata support, the UDM provides you with the ability to specify complex calculations to be applied to the underlying data; in this way you can embed business logic. You can specify the complex calculations by a script-based calculation model using the language called MDX (Multi-Dimensional eXpressions). UDM provides rich analytics such as Key Performance Indicators and Actions that help in understanding your business with ease and automatically take appropriate actions based on changes in data.
- **Model for Reporting and Analysis:** The UDM provides the best functionality for relating to both relational and OLAP worlds. UDM provides you with the capability of not only querying the aggregated data that are typically used for analysis, but also has the ability to provide for detailed reporting up to the transaction level across multiple heterogeneous data sources.

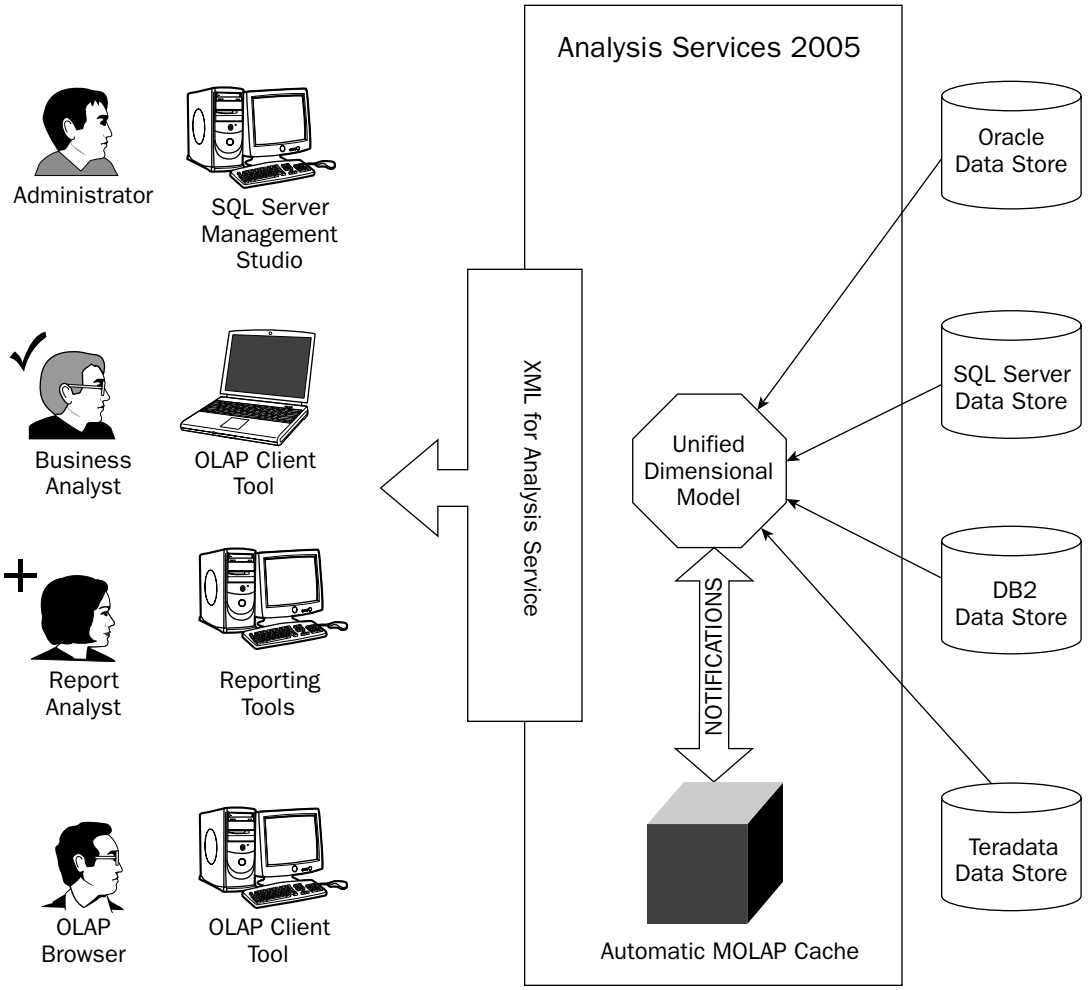

Figure 1-9

Another handy aspect of using the UDM is the storage of foreign language translations for both data and metadata. This is handled seamlessly by the UDM such that a connecting user gets the metadata and data of interest customized to his or her locale. Of course, somebody has to enter those translations into the UDM in the first place; it is not actually a foreign language translation system.

Summary

Reading this chapter may have felt like the linguistic equivalent of drinking from a fire hose; it is good you hung in there because now you have a foundation from which to build as you work through the rest of the book. Now you know data warehousing is all about structuring data for decision support. The data is consumed by the business analyst and business decision-maker and can be analyzed through OLAP and Data Mining techniques.

OLAP is a multidimensional database format that is a world apart in form and function when compared to an OLTP relational database system. You saw how OLAP uses a structure called a cube, which in turn relies on fact tables (which are populated with data called facts) and dimension tables. These dimension tables can be configured around one or more fact tables to create a star schema. If a dimension table is deconstructed to point to a chain of sub-dimension tables, the schema is called a snowflake schema.

By choosing Analysis Services 2005 you have chosen a business intelligence platform with awesome innovations built right in; like the UDM. Also, there is an advantage that Analysis Services 2005 offers — it comes from a particularly strong and reliable company that had the highest market share with its earlier product, Analysis Services 2000. The rest of this book illustrates the power of the platform quite clearly.

In the unlikely event that you didn't read the introduction, mention was made that you should read at least the first three chapters serially before attempting to tackle the rest of the book. So, please do not skip Chapter 2, an introduction to Analysis Services and Chapter 3, an introduction to the technology behind the most famous acronym in business analytics, MDX.