Realtime publishers

# The Shortcut Guide™ To

# Large Scale Data Warehousing and Advanced Analytics

*sponsored by*

aster data
more data. big insights.

*Mark Scott*

## Copyright Statement

# Chapter 4: Analytic Considerations for Large Scale Data Warehouses

As the amount of data grows in the data warehouse, so does the need to summarize and use that data in a more manageable fashion. For the purposes of this chapter, analytics will be considered the manipulation of data beyond standard relational database reporting. This will include On-Line Analytical Processing (OLAP), data mining, and the analysis of unstructured data (such as text).

We will explore the building of analytic data structures on a terabyte or petabyte scale. We will look at techniques that utilize the full power of a large scale data warehousing system to draw insight from this information. We will then discuss the processes for making the data available for use within the enterprise.

## Building Analytic Data Structures

One of the most common uses for analytical data is to project the most likely values for unknown data. This can be done through examining trends and data mining to help deduce patterns within the data.

### Predictive Analytic and Data Mining Structures

For most analysts, the ability to review trends and find ways to predict one thing based upon another is their bread and butter. The data warehouse provides all the raw data that they could use, but before they can use it to answer their questions, they need to organize the data into a more easily consumed format.

This process can take several formats. OLAP is commonly used because it typically can organize data and show how data varies over time, by location, by product, and any number of other types of attributes. (These types of attributes are known as *dimensions* in OLAP parlance). So not only can the immediate result be seen but also that information can be placed in the context of how things have occurred over weeks, months, or years, or in comparisons to other regions, products, and so on. It can help highlight whether things are getting better or worse. It allows the analysts to validate the effect of their decisions over a period of time. OLAP techniques are useful when looking for patterns found in broad groupings of data (for example, sales across quarters and product lines). Detecting patterns under more specific conditions requires data mining techniques.

Data mining helps provide analysts with the most probable outcome based on a set of known circumstances. How likely is it that Mrs. Jones, an 85 year old living in a nursing home is purchasing a new $4500 stereo system in a state 600 miles away? Or that the sales for widgets will go up next quarter? Or if Mr. Smith bought a new GPS system, is he more likely to buy a new car stereo or a laptop computer? Data mining can help predict these things and allow better-informed decision making.

## OLAP

OLAP has been used for the past several decades to help enterprises examine trends in data and look at the conditions under key performance indicators (KPIs) for their enterprises. The key to the performance of an OLAP database lies in its design and processing.

OLAP databases are based on multidimensional modeling. Multidimensional models identify interesting facts in quantitative data by showing the entities used to generate the facts. For example, a number of products we produced on Tuesday were produced on specific machines using specific employees, consuming specific materials. The date, the employees, the machines used, the materials used, and the actual products produced are all dimensions.

The dimension can be organized into hierarchies. The hierarchy helps group like items within a dimension. For instance, the machines used to produce the product are part of a line. The line is part of a plant. The plant is at a location. The location is part of a division of the company. Each of these groupings is a level within the hierarchy.

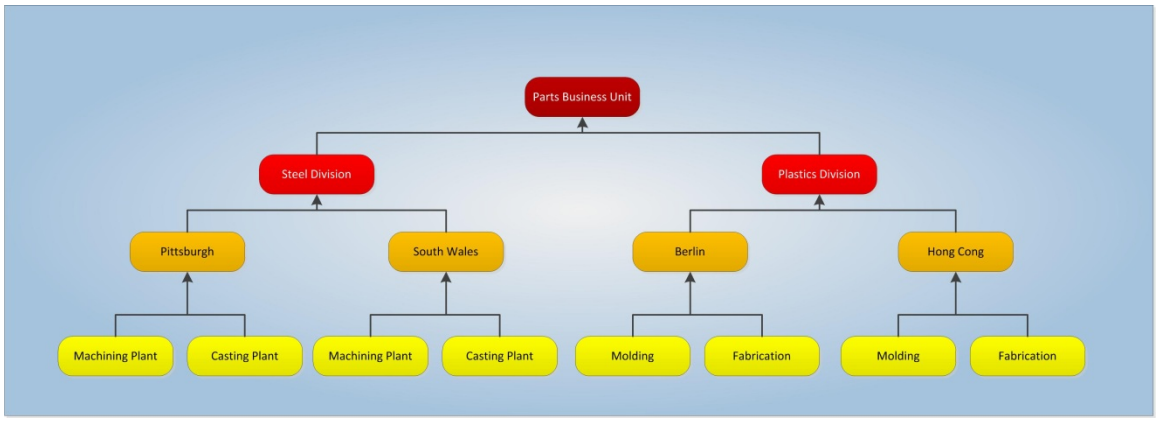


**Figure 4.1: A dimension organized into hierarchies.**

When an analyst approaches the OLAP data, they want to explore the combinations of entities that work together to produce either good or bad results. An OLAP cube will use the hierarchies in the dimensions to pre-aggregate the data so that when an analyst wants to know the production for a specific plant, the answer is already totaled and can be retrieved quickly.

Thus, OLAP cubes are created from aggregation tables. The aggregation tables pre-calculate the information at various levels of the hierarchy so that the information is ready to retrieve on demand. Each cell of the OLAP cube contains measures, such as the count of products sold, the average sale price, the average profit margin, total profit margin, and so on. Processing a cube prepares these aggregation tables by computing each of the measures in all cells. The aggregation can be simply sums of the values of the measures, but they can be more complex measurements, such as minimum or maximum values, averages, or other non-additive processes.

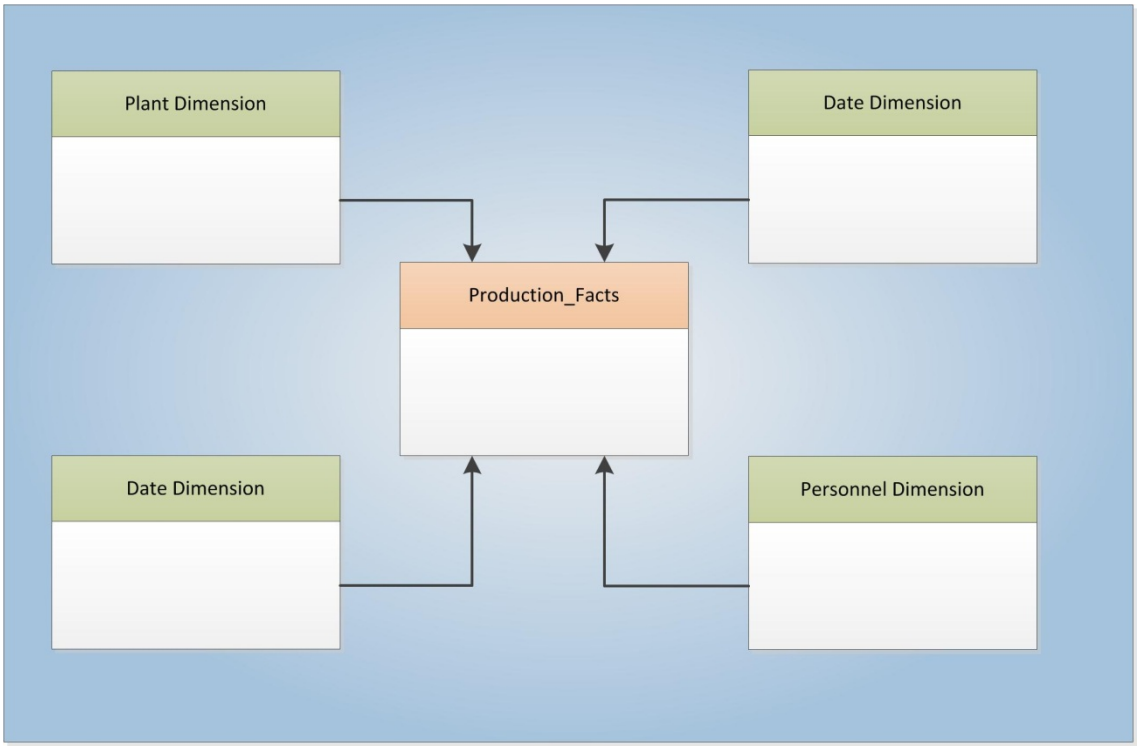


**Figure 4.2: An example OLAP cube.**

The analyst may query for a high-level aggregation, for instance the sum of production by a business unit for the third quarter across all the parts manufactured in all the plants in the enterprise. He may want to drill down from that number to determine which plants produced over their quota. He might then want to look at only machined parts, not molded parts. This information can help determine what is working well, or not so well, within the organization. Pre-aggregating the data so that the analyst can move virtually instantaneously from one level of aggregation to the next makes this approach very productive and helps the analyst reach decisions in a timely manner.

The analytic system supports the creation and updating of these aggregate tables. The aggregates can be many (depending on the number of dimensions, the number of hierarchies within the dimensions, and the sparseness of cells with non-zero measures), so the aggregate tables can consume quite a bit of disk I/O and processor power to create and maintain. As the amount of data within the warehouse grows, the cubes and aggregate tables grow rapidly. Managing the processing of the cubes and providing the system with the flexibility to modify the cube design to meet user needs is key to the success of an OLAP installation.

> **Note**
>
> I have worked on real-time OLAP applications that traded energy production capacity. They needed to continually monitor changes in demand so that the traders could accurately predict how much energy they would need and place their orders before the prices rose—without buying capacity that they would not require. This put strong demands on the system and required an analytical processing system that could simultaneously respond to the traders and process changes in demand and generation capacity. This is a daunting challenge for all but the most capable systems.

### Data Mining

Data mining analyzes data to find patterns and predictive analysis within bodies of data. Data mining is commonly used in a variety of applications. Predictive analysis uses trends in the data to project upturns and downturns in the business. Classification can help profile the likely preferences of customers. Market basket analysis is used to determine what groups of products are purchased together. Data mining can be used to detect fraud in credit use. The application of these techniques continues to grow.

Data mining algorithms come in different flavors for differing purposes. Classification algorithms can be used to predict a discrete variable (such as a decision tree) based on the attributes of other known variables within the dataset. Regression algorithms are used to project continuous variables, such as profit or loss. Association algorithms are used to find common associations between entities, and are often used in market basket analysis. Segmentation algorithms are used to group entities with common attributes. Sequence analysis algorithms can help summarize frequency or episodes found in data usage and can be implemented in fraud detections. Because different algorithms serve differing purposes, it is not uncommon for several algorithms to be applied to the same base data.

The process of generating a model requires that a portion of the data be analyzed by the mining algorithm to build a pattern. A second subset of the data is used to test the success of that pattern in predicting results with actual, real-world data. Once the model has been refined, it is applied to new data and used to provide projections and classification, offering the most likely result of an unknown value based on the values that are known.

Data mining models work best with large data sets to train the model and test the results. Although smaller samples of the data can be used, the results will tend to be less accurate (particularly if the data sample is inadvertently skewed by the sampling selection process). Large scale data warehouses typically have plenty of data to work with, but providing the processing power and throughput to work with the data and produce the mining model can create a very strong demand.

Furthermore, the model should be refreshed on a regular basis to ensure that the original model is still valid. Comparing a control group against a random group can help determine whether the actions taken on the data mining results have been effective. This process is called *uplift modeling* and can help direct the organization in using the results of the data mining effectively. It also produces additional demand on the data warehouse analytical systems.

> **Resource**
>
> You can find an introduction to data mining and its relationship to OLAP and relational databases at http://homepage.cs.uri.edu/faculty/hamel/pubs/hamel-197-manuscript-final.pdf.

### Building Locally or Remotely

The question then becomes where to build and process the data structures that support analytics. Should they be built locally on the servers where the data is stored or remotely, on distinct analytics servers?

In earlier chapters, we discussed the advantages of constructing a data warehouse where all the elements of the warehouse are contained and managed in a single system. The system can be easier to design because the various components are all contained within a single process boundary. Maintaining security and keeping elements in sync within a single, integrated system is much simpler. Managing and troubleshooting the processes that extract, load, and transform data are much simpler. Data governance is typically simpler in a unified system and can help assure the data is maintained in a compliant manner.

Furthermore, building locally can be much more efficient (providing the system is designed and built correctly). Moving data between nodes within a massively parallel processing data warehouse system is much more efficient than moving data between servers in a typical data center. The connection between nodes is dedicated and optimized for the flow of information between servers. This reduces the demand on other parts of the network and even on the individual data nodes themselves. In a shared-nothing architecture, the individual nodes can work in parallel with one another with fewer bottlenecks. This helps reduce the time it takes to process data and allows the system to remain responsive during load times.

Local builds are even more effective if the processes used to build the analytic structures can be distributed across the data nodes. By accessing data locally, processing it locally, and writing the results locally, the processing events can be optimized. This lets each data node run at its highest efficiency. Parallelizing the process across many data nodes allows the data to be generated very quickly and enables massive amounts of information to be analyzed and processed.

Many analytic data systems do not process locally. They retrieve data from the data warehouse and process it on a separate server. This allows the server to be dedicated to analysis—it does not compete for resources with the other activities of the data warehouse. It can help to have separate servers perform different functions if the data warehouse itself cannot be scaled out to meet its demands.

The costs of processing remotely are several. First, the data must be moved from the data warehouse to the analytics servers. This can be costly in terms of network resources. As these are often very large downloads, they can have a major impact on the network segments, and even the SANs, in the data center. The copying of data can also delay the time until the data is available for analysis by the end users. The cost of data governance when data is hosted in multiple diverse systems can be considerable and should also be taken into account.

Analytics processing is not necessarily linear. There is a very strong demand during processing and data loading. As the point of processing is to reduce the effort of answering queries, the servers often are often underutilized while not processing. In a massively parallel processing environment, the servers could be tasked to perform other work, but in a standalone analytics server, this is more difficult.

Scale-up of a separate analytics server can also be an issue. Many servers do not have the inherent scalability of an MPP data warehouse system. This can lead to capacity issues as the system's demands and data size increase.

### Optimizing the Use of Processor Power

In single server systems, to do more work, particularly with multiple cores, you need to write a multi-threaded application. This help to use the processor efficiently and make good use of system resources. It is difficult to write multi-threaded systems so that the individual threads do not block one another when using global resources. As a result, many systems are only written as single threaded systems and cannot take full advantage of the server.

For large scale systems that require multiple servers, there needs to be a method by which multiple servers can be employed against the same tasks. They need to coordinate their efforts so that they do not duplicate one another's efforts. They also need to be able to coordinate and share information with one another. If the servers can work together, the tasks can be made parallel and the working can be done much more quickly and much more efficiently.

The optimum large scale analytical systems will allow parallel execution of code on multiple nodes. It will provide a framework that will help the individual code objects work together so that they can coordinate their efforts. This will make it simple to parallelize tasks; by processing simultaneously on multiple nodes, the system will be able to make good use of all available resources.

The other issue of analytic systems is that there is often an imbalance in how system resources are used. The system tends to require a great deal of resources when it is processing a lot of cubes or working with the data mining models. When it is querying, it often needs fewer resources. The system must be designed to handle the processing needs and have idle resources available when it is not processing.

In a large scale enterprise data warehouse that combines analytics, the resources can be evenly distributed between the tasks of maintaining the data warehouse and providing support for the analytics. This can make better use of the resources because they are more evenly used throughout the day. As long as the system can be scaled to meet the need, the warehouse can provide all the resources required to support the relational database functions as well as the analytic functions. This can lead to a much more efficient use of total system resources.

## Advanced Analytic Structures

Any analytics systems provide standard functions to help analysts harvest data from the system. These systems are meant to meet the most common needs of their client base. They can, however, be insufficient for very specific needs of very specific customers.

> **Note**
>
> I have worked on a number of systems where the analytics server could not support a specific function required by the organization. In one case, I was working on a system that was analyzing specific temperature changes in body fluids. The functions required were not additive but rather a specific function that had to be applied to each data point in the OLAP cube in order to perform the appropriate aggregations. This severely limited the choice of analytics tools that we were able to employ.

Most analytics systems provide the means for developing and deploying customized code to help perform more sophisticated analysis of the data. Although this might work very well in a single server system, it might not be as well-suited for a massively parallel processing data warehouse system.

When looking at data warehouse systems that embed analytics, it is important to consider the system's ability to deploy custom code in a multi-server environment. To gain the full advantage of the system, it should support mechanisms that allow code to be developed and distributed across the data nodes. If the system allows the developers to write code that can be placed on the data nodes, developers can create a custom system that can take full advantage of the parallel processing power of the system. This will allow the system to employ advanced custom analytics for the enterprise. It will allow the developers to create advanced analytic structures that meet the precise requirements of the enterprise. Data that is not applied uniformly across the nodes could result in errors in the final results and a loss of confidence within the user community (not to mention potentially disastrous decisions made on inaccurate data).

## Analyzing the Data

When using a large scale data warehouse to analyze data, one must consider how the code can be structured and distributed in order to support the use of the massive parallel processing capabilities of the system. Using a framework that supports distributing code objects across the data nodes will help to leverage that power. A system that can help identify bottlenecks and code execution will help optimize the code and provide a robust responsive system. The code should be easy to write in common languages to help developers devise and employ the code simply and rapidly. If that code can be embedded within the system, the system can be used to its full potential.

### Building Analytic Code for Harvesting Data

One of the keys to developing code that can be distributed easily in a massively parallel processing environment is to utilize a framework that is specifically designed to distribute code of multiple servers. Such a framework can be used to simplify the process of getting individual objects to work together when executing on multiple servers simultaneously.

The framework should allow developers to concentrate on the logic of the code that they deploy rather than on the mechanics of distributing the code across multiple servers. The degree that the framework simplifies the process of using a massively parallel processing system frees the developer to concentrate on extracting the precise information that they need from the system. This will help to enhance developer productivity and reduce time-to-market for new analysis projects.

**MapReduce**

One such system for distributing code across multiple processors is MapReduce.This system allows Map functions and Reduce functions to work together to combine the results from code run on multiple servers in parallel. The Map functions allow data to be organized into a key-value pair list. The Reduce functions processes the lists in parallel to produce a set of values—the values that are the desired result of the initial query. For instance, the Reduce functions might return the number of electronic products sold in a region during a specific quarter. This can be returned from each node, resulting in less data to compile to determine the final result. Google has used this framework to process the petabytes of unstructured data that it mines for its search service. The framework has been adopted by major vendors in the massively parallel processing data warehouse market. You can learn more about MapReduce at http://static.googleusercontent.com/external_content/untrusted_dlcp/labs.google.com/en/us/papers/mapreduce-osdi04.pdf.

The framework should allow the code to be executed on each node within the system. The result should be gathered and processed into a single unified result set. The degree to which the data can be processed within the bounds of an individual node and then have the data emerge from all the nodes into the single result set is the degree to which parallel processing can help improve the performance of the system.

## Overcoming Bottlenecks in Code Execution

One of them major obstacles to overcome when working with data is getting the data from the database management system to the code that actually processes it. In most systems, the analytics code is executed in a separate process than the services that actually store and retrieve data. There is necessarily a layer between the two services. This is often a data access layer such as ODBC or JDBC. The imposition of this layer will inherently create bottlenecks between the two systems.

One way to overcome this obstacle is to allow the code that processes the data to run with the same level of authority and access as the code that manages the data. If the analysis services are indeed a different executable than the code that manages the underlying data, this bottleneck cannot be effectively removed. The problem is magnified if the data has to be exported from the data warehouse to a different analysis server.

**Note**

I have worked on several systems where the analysis server and database management server were co-located on the same host. Having both servers execute on the same host did eliminate some of the bottlenecks created by passing data through the network between servers. It did not, however, eliminate the problem that the two services were in competition for the same resources. The contention could have been resolved if both services ran within the same processes and could better coordinate and balance sharing resources within the system.

Providing the functions that analyze the data, produce aggregations, or generate the data mining structures with the greatest level of access to the data will make processing the structures much more efficient. Data warehouse systems then can freely share cumulative resources between the analysis processes, and the data storage and retrieval processes can make the best use of the resources that are distributed throughout the warehouse.

## Simplifying Deployment and Management of Analytic Code

There are two elements to creating and deploying code used to analyze the underlying data. One is the development of the code itself and the other is the process of distributing it across the nodes that implement the code.

### Creating Analytic Code

One of the greatest expenses of developing any application is the cost to developers. Finding developers who are skilled in their trade can be difficult. Providing them with tools that help improve their individual productivity and simplify the process of their work can pay great dividends in terms of the amount of work produced and the cost incurred.

The cost of mastering a specific database and its specific API must be added to the equation. Each database system will provide its own interfaces to access the data and typically a data layer that abstracts the database engine and its operation from the code that operates on it. Frameworks and development systems that simplify working with these systems can help speed development and ease the task of the developers (as well as get new analyses ready sooner).

A framework that can be implemented with a popular Integrated Development Environment (IDE) will allow the developers to get to work quickly and help them feel more comfortable. If less time can be spent training developers on how to use their tools, more time can be spent on producing useful code.

**Note**

Although most developers can learn any language relatively quickly, I have found that most of them will find a language in which they are most fluent and comfortable. I myself find that when I am using my favorite languages, it is easier for me to write code that is clear and efficient. That does not mean I will not use another language, but it does mean that I do my best work when I am using the languages with which I am most comfortable. Having a system that lets me use my favorite languages will help improve my productivity. Of course, the choice of language should not be left solely to the individual developer. A corporate standard for language development will help the entire development team work together to improve and support one another in their development efforts.

Languages are another issue. Although almost any language can be used for development, most developers will feel more comfortable in one language than another. A framework that is flexible and can support multiple languages will allow developers to work in the language with which they feel most comfortable. This can help improve their productivity and simplify the development activities.

### Deploying Analytic Code

Once the code is written, it will need to be distributed throughout the system. As most developers have little experience in working with massively parallel processing systems, they are unlikely to have developed techniques for placing parallel code on multiple nodes. A well-designed system will help automate this process and extract the mechanics of deployment code across the nodes from the developer.

Once the code is deployed, it will need to be maintained. The simpler the process of updating and distributing new code revisions, the easier the system will be to maintain. Again, a development system that can ease this process will help the developers remain productive without becoming mired in the details of deploying code throughout the system.

The role of the DBA is often impacted by the system. A system that simplifies maintenance and deployment will help reduce the impact on the time spent by the DBA in keeping the system operational as well as lower the total cost of ownership.

Distributing code throughout the nodes within the database system is only part of the code management issue. There need to be development environments and testing environments in addition to the production environment that is used every day. Those of us who have spent a great deal of time developing database systems know the pains of moving code from one environment to the next. A framework that helps simplify moving the code from development to test to production can help reduce those pains and make maintenance of the databases much simpler.

## Embedding Analytics Code inside the Database

When the analytics system and the database system are distinct, there's always the possibility that the interfaces used by the analytics system will be broken when there are changes to the database system. A change to the schema in a table or view can go unnoticed by the database developer, but it will break the analytics system when that system tries to access the view or table. A change to a stored procedure can prevent the analytics system from being able to modify the underlying tables.

A system that can combine the objects that store and retrieve data with the objects that analyze data and build the aggregates or advanced analytical data structures will be easier to maintain. The physical structure of the relational data can be abstracted as a view or stored procedure. Then the analysis system that uses that view or stored procedure must be separately modified. These events must be carefully coordinated and performed on distinct systems. This can create maintenance issues, involve security issues, and, when things are not well coordinated, break the analysis systems.

A warehouse system that uses a combined package to manage the relational data and generate the analysis data structures is inherently simpler to maintain. The changes made to the system can be tested end to end with and a single system, even within a single IDE. This minimizes the risk of breaking components and causing the analysis system to fail because changes were not coordinated across multiple systems.

> **Note**
>
> I have worked in many organizations where the people who work on the analysis systems do not coordinate well with the people who maintain databases systems. The reasons for this can be a simple inability to communicate or highly charged political battles over domain. The communications barriers can lead to hindrances in productivity. Regardless of the reason, having a single system that maintains the data and provides the analysis will often help overcome these barriers. That will help to get products to market sooner and reduce strife within the organization.

A large scale data warehouse system also offers an opportunity to simplify the management of multiple environments. If the system can be scaled to host development, testing, and production, it becomes easier to maintain code and promote changes from one environment to another. As long as each environment can be isolated and security can be maintained, a single system is much simpler.

## Distribution of Analytic Data

Once the analytic data structures are created, you'll want to put that data in front of the users. There are several required processes to accomplish this. You can distribute analytical data throughout your enterprise, using remote analytic servers to service the needs of users located across the globe. In other circumstances, you may want to keep the data centrally located. Regardless of the method by which you provide data access to the users, you'll need to carefully manage the way the data artifacts are created and synchronized. You'll want to control the governance of those artifacts. You'll also need to maintain the data artifacts to ensure that they are up to date and accurate.

### Distributed Data Artifacts

Processing the analytical data structures often requires a great deal of computer resources. Large scale data warehouses can be designed to provide that horsepower. However, if users are widely distributed, it can be advantageous to distribute the analytical data structures to remote servers in remote locations throughout the world. This is commonly referred to as a *hub-and-spoke architecture*.

In previous chapters, we have discussed the advantages of using a single central location to house and process analytical data. Once the data has been generated, you can export it through your WAN connections to remote servers. For remote locations that have high-cost or low-speed connections, this architecture can serve very well to allow the analytic data to be located near the users who consume it.

Another advantage of creating smaller analytic data marts is that the data marts can contain a subset of the total analytic data in the warehouse. This can serve as a security mechanism. You can limit the data in a data mart to a specific set of data that a defined group of users is authorized to see. The smaller data sets can often be handled by a single server. However, if the resulting data mart is large enough, you might want to consider implementing an additional large scale data warehouse to host the data mart.

> **Note**
>
> I worked on a project where several regional managers needed to analyze the performance of their individual regions. The regional managers were very competitive. Even though the data was for a single organization, the regional managers were not allowed to view one another's data. By creating a single central analytical database for use by corporate management and data marts for each of the regional managers, we were able to secure the data used by each regional manager and still maintain consistency across the entire data set. The user data marts can help as a security mechanism as well as a means of creating smaller targeted analytic data sets.

Data marts can be useful for distributing subsets of the analytic data and can address several enterprise network issues. They do, however, introduce their own challenges. When creating data marts, it is important to ensure that the data is kept refreshed and synchronized. The processes for distributing the data to the individual data marts must be carefully monitored. This adds operational overhead and can pose challenges for the operations team. The use of data marts should be carefully considered by balancing the effort to maintain the marts with the advantages of using them within the enterprise. Some analytical systems will need to store the results of their analysis for later use. This can add considerable overhead to the maintenance of distributed systems.

## Centralized Data Artifacts

The simplest way to maintain the data is to keep all the data stored in a single system. Frequently, data marts and other mechanisms for distributing data through the enterprise are not based on the advantages of data marts per se, but rather the limitations of the servers that host the data warehouse in the analytics data. We have discussed at length the advantages of keeping all the data in a single cohesive system. These advantages include:

- The ability to drill through from analytic data to detailed relational data
- The simplicity of maintaining data within a single system
- The ability to secure data by using a single comprehensive data model across all related data structures
- The improved governance of dealing with sensitive and confidential data within a single system

Having worked on many data warehousing systems, I have seen the impact of developing a data warehouse and analytic data systems when heterogeneous mix of servers, data management, and data analysis tools are used to maintain the data. This not only slows the development process but also makes it difficult to maintain a unified vision across the enterprise.

A single central core that provides the basis for reporting within the enterprise can help the data stewards and IT management personnel to control the integrity and veracity of that data. The use of the centralized system to maintain this data often results in analytical data being brought to market in a much shorter time, with a much lower cost of development.

A common reason for not maintaining a single central data store is the limitations on the systems that maintain the data. With systems such as massively parallel processing data warehouses, the enterprise has a platform that can be dynamically expanded to meet the demands of the central warehouse in a practical and cost-effective manner. The advantage of maintaining all the elements of the data warehouse, including the analytics systems, in a single platform can be realized with such systems.

## Management and Governance of Data Artifacts

The data harvested by an organization represents great opportunity for serving the needs of the organization. It also represents a serious responsibility. The data is often confidential and represents a trust between the organization and its client base. The proper handling of that data is very important to maintaining that trust.

As we develop analytic systems, it is easy to set different rules for protecting the analytic data than are used to protect detailed relational data. This can lead to unintentional breaches in the security of the data used in the analysis. If a single system is used, a single security model applied to all the data within the system, it becomes a much simpler task to ensure that the data is secure. This can help protect the personal and confidential information maintained by the organization and to prove that that data is indeed secure.

> **Note**
>
> I worked on an analytics system that helped property owners project the potential operating expenses of commercial properties by comparing the operational statements of similar properties within a geographic region. It was important to ensure that the specific data for single property could not be isolated by carefully controlling the parameters used in the analytics system. The system needed to be designed to ensure that no data would be displayed unless a minimum number of operating statements were included within the aggregation. Because we used a single centralized system, the security model was much easier to build. It helped us complete the project on time and within budget.

Beyond the simple management of the data, many organizations deal with strict corporate and government compliance. Whether dealing with industry standards such as PCI DSS, government regulations such as HIPPA and SOX, or internal corporate standards for controlling access and handling of critical data, a single system is the most direct way of maintaining the standards.

For many of these systems, there are requirements for auditing the movement and access of data. This auditing must be maintained as a consistent requirement for demonstrating compliance to these regulations. A single system that can audit the access to data from the moment it is staged to the point where it is delivered to the end users will simplify this process and help ensure that the enterprise maintains compliance. If a breach does occur, it can also help quickly identify such a breach and bring the issue to a swifter resolution.

In many organizations, data lineage is also a critical issue. It can become very important to track from analytic data back to the source system from which the data was derived. This can help validate the authenticity of the analytic data and may be crucial in identifying errors in the source systems or the ETL process used to produce the analytic data from which decisions are made. A single system that maintains the metadata used to track the data lineage from staging through the development of the analytic data structures will enhance the ability of the organization to track data end to end.

### Maintenance of Analytic Data

There are server factors involved in maintaining the analytic data. Initially, the processing of the analytic data is part of the overall ETL process used within the data warehouse. The processing of OLAP cubes and data mining structures needs to be coordinated with the loading of the source data in the relational systems. Managing that process within a single system will help ensure its consistency and make it much easier to maintain. These processes can be maintained within a single set of jobs, so coordination of the steps (staging data for source systems, ETL into the relational tables, and the processing of analytic structures) can be more closely coordinated. This can shorten the time it takes to get the data from the source systems into the analytics system each day.

When changes occur in the source systems, the data warehouse and analytic data will need to be adjusted accordingly. Providing those changes within a single system will help ensure that the changes are made consistently across all the dependent objects within the data warehouse and analytic system. This can prevent analytic systems from becoming unavailable due to changes in the backend systems that supply them data. This will allow the analytics system to remain more agile and able to keep up with rapid changes in many dynamic organizations.

Large amounts of analytic data need to be carefully protected. It is easier to back up the analytic data in a single large scale system than to coordinate backup of many diverse systems. This of course is dependent on having a system that can backup the data quickly without impacting the ability of the system to serve the users. The massively parallel processing data warehouse systems that we have described use distinct nodes for backup and take the pressure off the nodes that deliver that information to the users. Data backup can be contained within a single, easily monitored system.

When single server analytic systems are implemented, they are often not clustered or farmed to provide business continuity in the event of server failure. A massively parallel processing data warehouse system inherently provides internal clustering across the nodes. This makes the system inherently more resilient when single servers fail. The system will automatically recover from the failure of a single node within the system. This ensures that the analytic data is available to the users if a server fails.

## Conclusion

In this chapter, we looked at the advantages of housing and processing the analytic data within the same system as the relational data that has been brought in from the source systems. We have laid out the advantages of processing the analytic data within the same systems that maintain the relational data in the data warehouse.

This guide has discussed the implications of maintaining large amounts of data in a single data warehouse. We discussed the advantages of keeping that data in a single data warehouse structure. We discussed the challenges of maintaining large scale systems and the technologies available to allow terabytes and petabytes of data to be maintained and placed at the fingertips of the decision-makers within an organization. We explored the architectural options for building these large scale systems. We discovered that systems that use multiple commodity servers working together often provide the most economical and effective solution for the systems. Massively parallel processing data warehouse solutions provide many advantages.

We discussed the operational advantages of basing your data warehouse in a single federated system. Maintaining and securing data within a single system is much simpler than maintaining the same body of information across multiple systems. When the processes of importing and organizing data are contained within a single system, they are more efficient and easier to monitor. A central data warehouse system provides these advantages.