Realtime publishers

# The Shortcut Guide™ To

# Large Scale Data Warehousing and Advanced Analytics

*sponsored by*

aster data
more data. big insights.

*Mark Scott*

Chapter 2: Choosing the Right Architecture for Your Large Scale Data Warehouse ..... 21
Approaches to Large Scale Data Warehouse Logical Architecture ..... 21
Single Database Warehouses ..... 22
Hub-and-Spoke Architectures ..... 25
In-Database Analytics ..... 27
Approaches to Large Scale Data Warehouse Physical Architectures ..... 28
Single Server Systems ..... 28
Federated Database Implementations ..... 29
Massively Parallel Processing Data Warehouse Systems ..... 31
Approaches to Large Scale Data Warehouse Software ..... 33
Single Vendor Stacks ..... 33
Third-Party Support ..... 34
Best of Breed Software ..... 34
Open, Shared Standards for Data Interaction ..... 35
Approaches to Large Scale Data Warehouse Security ..... 36
Data Entitlement ..... 36
Maintaining Security in Depth ..... 37
Data Protection and Business Continuity ..... 37
Summary ..... 38

## Copyright Statement

# Chapter 2: Choosing the Right Architecture for Your Large Scale Data Warehouse

As data warehouses grow, a number of issues begin to crop up. Time to process data becomes quite lengthy. Tables become so large that the time it takes to query them becomes excessive. Support for in-database analytics overtaxes the available server resources. The service level agreements (SLAs) can no longer be met. As the data warehouse grows to 10TB and beyond, conventional relational database management systems often prove inadequate.

To meet this challenge, the architecture of the data warehouse must be carefully considered. The logical, physical, and security approaches to large volumes of business data are very different from those where the databases are smaller and often less comprehensive.

This chapter will consider the ways in which large scale data warehouses and advanced analytic systems can be designed and implemented. Thinking about the logical and physical architecture, we will look at alternatives to meet the unique needs of large scale data volume. We will consider approaches to finding the correct tools to manage the system, and compare and contrast their relative merits. We will also explore the challenges to securing large data sources and managing business continuity.

## Approaches to Large Scale Data Warehouse Logical Architecture

Modeling a large scale data warehouse is very different than modeling a simple star schema or aggregation found in smaller data warehouses. The large scale warehouse still needs the basic considerations of building conformed dimensions that will be relevant throughout the enterprise. The grain of the fact data still needs to be set accurately. The management of parent-child relationships, slowly changing dimensions, and semi-aggregatable facts remain thorny issues. The problem becomes one of the manner in which sheer size affects performance. The performance issue is manifest primarily on two fronts.

First, data has to be loaded into the system. Many source systems have a rigid window of time in which data can be exported without negatively impacting the response of the system. With multiple source systems—often competing for the same period of time for export—this window can put a lot of pressure on the data warehouse to load a very large amount of data in a relatively short period. International enterprises further exacerbate the problem by requiring the system to be available 24 × 7. They often provide data for loading during times when other countries are in peak query times. The design should facilitate the loading of data without disrupting the ability to respond to queries.

The second issue is raw query response. When tables have hundreds of billions of rows and multiple table joins to collect the data, the task can overwhelm the available disk throughput and processor power. The system loads data slowly and cannot respond to queries in a reasonable timeframe. This often leads to users becoming frustrated or even abandoning the use of the system altogether. System capacity can be expanded, but somehow data volume grows to just over the capacity the system can reasonably support. A design must consider ways in which querying can be optimized so that large data sets can provide analysts with the responsive system they need to do their jobs.

## Single Database Warehouses

Data warehouses can be divided into individual databases, each with its own purpose. A data warehouse is naturally divided in sections, just as a house is divided into several rooms. Each room has a specific purpose and design features that support that purpose. Although a different database or database server could be used for each "room," this setup would be inconvenient, as having a collection of one-room sheds would be much less appealing than having a house. A single data warehouse that can host all the sections will be easier to maintain and operate. Let us consider the specific parts of a typical data warehouse.

The first section of the data warehouse is the staging area. This is where the data comes from the various source systems or systems of record. It might be the Enterprise Resource Planning (ERP) system, the Customer Relationship Management (CRM) system, or the payment transaction system—any number of systems depending on the organization and its needs. The data in staging is modeled to simplify, and often accelerate, the process of receiving data from these source systems. It is usually designed to make it simple to "land" data into the warehouse and to troubleshoot issues when the data loads encounter unforeseen issues. It is not uncommon for some extraction, loading and transformation (ETL) activities to accompany this loading, so the system will have to provide capacity for supporting those activities during the load cycles. These transformations can be time and resource consumptive. It may take longer to perform the transformation operations than actually load the data. All this will add to the demands made on the data warehouse system.

The next area is commonly referred to as the Operational Data Store (ODS). This area (at least for the purposes of this discussion) is a relationally modeled data structure cast in third-normal form. This data structure is typical of database designs and is used to help organize the data as it is integrated from multiple source systems

> **Note**
>
> Database normalization is a set of rules that organizes table structures and minimizes redundant data. This can help control database size and minimize data errors within the system. It can also make report queries run somewhat slower. For more information on database normalization, see http://support.microsoft.com/kb/283878.

The ODS will require several other operations before it can be used. First, the data will undergo cleansing and quality checks to ensure that the source systems have provided complete and accurate data (which they frequently do not). They will also need to be conformed to corporate master data.

> **Note**
>
> Master Data Management (MDM) is the process of creating a single view of data shared by multiple sources systems, corporate division, and other groups within an organization. This can include customer or client lists, product information, account details, and a variety of other items that everyone in an enterprise needs to share. You can learn more about MDM at http://www.cio.com/article/106811/Demystifying_Master_Data_Management.

There are additional resources required by the system to move data between areas and perform data quality assessments and data cleansing. These must be accounted for within the architecture.

The next section of this archetypical data warehouse is the reporting database. This database is often dimensionally modeled and designed to simplify and accelerate reporting. Such systems often de-normalize data to allow query results to be returned more quickly. That means these data structures will trade space for speed, organizing data in the way that will get users the quickest responses.

> **Note**
>
> Dimensional modeling helps organize data to group quantitative facts that profile the performance of an organization. The structures help query information more quickly than normalized data models and lend themselves to becoming the foundation of On-Line Analytical Processing (OLAP) databases and data mining models. You can find more information about dimensional modeling at http://www.redbooks.ibm.com/abstracts/sg247138.html?Open.

Building a dimensional model will require processing capacity. It also requires additional space in the drive system.

The next section of the data warehouse contains analytical structures, such as OLAP cubes and data mining models. OLAP cubes pre-aggregate dimensionally modeled data to accelerate queries. If an organization tracks data at a transactional level, most analysts will want to see it aggregated. Rather than each individual sale, they will want to see all the sales for a given day. They will look at the sales for a given store for a given week. Or all the sales for a given group of stores over a quarter. OLAP cube structures will calculate those aggregations in advance and thus be able to respond with a simple materialized result rather than calculating the numbers on the fly when those numbers are requested.

> **Note**
>
> OLAP systems can store data in aggregate tables stored in a common relational database, called Relational OLAP (ROLAP), or in a proprietary structure based on a specific technology, called Multi-dimensional OLAP (MOLAP) or a hybrid of the two (HOLAP). To learn more about these data structures, see http://www.information-management.com/news/1330-1.html.

The final structures in our review of the data warehouse are data mining models. Data mining models apply mathematical algorithms to data to help find underlying patterns. These patterns can help fill the gaps in missing data. They can help group common factors to help enhance the prediction of behavior. They can be used to project future trends and enhance forecasts based on observed trends.

> **Note**
>
> You can find an example of the application of data mining model techniques described at http://www.information-management.com/specialreports/20050215/1019956-1.html.

Both OLAP cubes and data mining models will add to the size of the database. They both require computational power to create and maintain. They both add to the load on the system and require time to build, update, and query.

All these structures require the resources of the data warehousing system: space to store the structures, processing power to move the data, memory to buffer activities and help respond to queries, and CPU cycles to determine how to change data, where to write, and how to find it when requested. As these systems grow, they can overrun the resources of the system that hosts them.

## Hub-and-Spoke Architectures

Since a single, logical database can overwhelm a single server system, it may make sense to break up the model into smaller pieces. Just as a single house makes it easier to go from room to room, there are still many times when adding a shed to the property or a detached garage may serve the purposes of the tenant.

So the question becomes, what belongs in the house and what belongs in the sheds? There is a great deal of interaction between the staging tables and the ODS. The staging tables need the data scrubbed, analyzed, and conformed before it is entered into the ODS data structures. This can be accomplished in distinct logical databases, but such a separation can cause logistical and performance issues. The staging tables need to interact with the reference tables in the ODS to be properly transformed before being added to the system. Most database designers find these processes much simpler if the staging tables, reference tables, and target ODS are all part of the same database structure. Thus, a natural break can be made between the staging databases and the ODS, but it will add a certain layer of complication to the system.

The analytic structures, the dimensional reporting databases, OLAP, and data mining structures are another matter. Often, the analytic databases address a specific subject area. As such, they represent only a subset of the data within the data warehouse. You can separate these subsets into their own databases and use them as data marts. ZDNet defines a data mart as "a subset of a data warehouse for a single department or function." Data marts can be extracted from the enterprise data warehouse. This can reduce the size of the data warehouse itself and scale out the processing demands of the server by placing the data marts on different servers.

The essence of a hub-and-spoke architecture then is to create a central enterprise data store. This is the hub and it serves as the primary data source for the data marts, which represent the spokes.

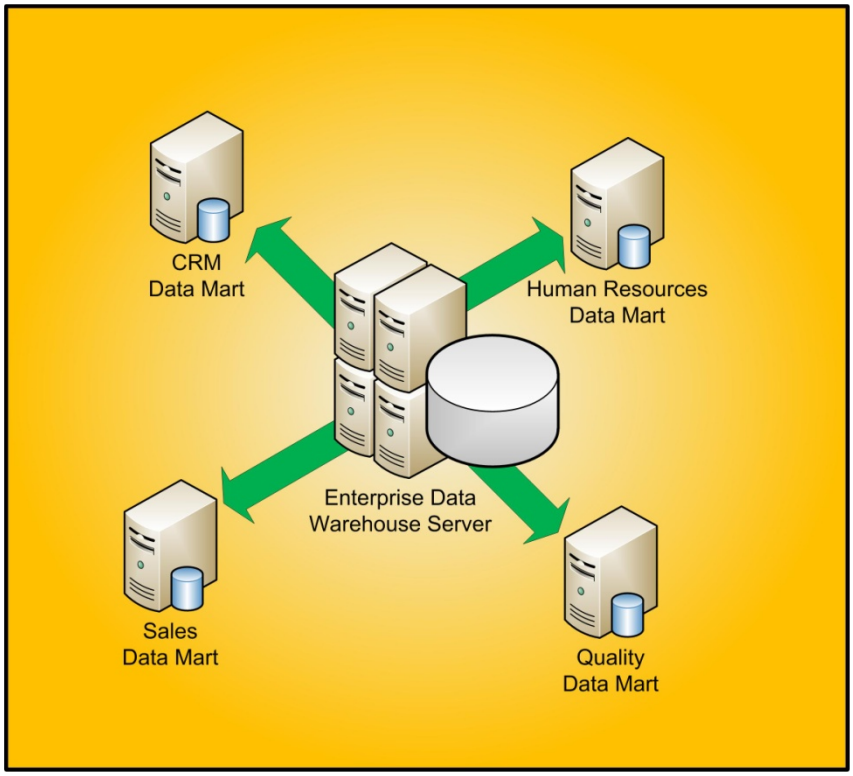


**Figure 2.1: Typical hub-and-spoke architecture.**

This architecture will offer several distinct advantages:

- The load for analysis and query response can be scaled out to multiple servers
- The data marts work with smaller subsets of data and can typically run on more economical hardware
- The data mart servers can be geographically dispersed to reside nearer the users

There are operational considerations with this design pattern. The architecture uses multiple servers, so additional attention needs to be paid to ensuring that the security is transitive throughout the system. Data authentication and authorization needs to be maintained across multiple servers. Often, the spokes are built on different database software platforms and server platforms. This makes a consistent security model even more challenging.

The spokes need to be synchronized with the hub. This will mean another set of data movements that must be coordinated within the enterprise network. If real-time or near real-time reporting is required, coordinating data flow across multiple servers in a continuous fashion can place high demands on network bandwidth and operational components.

Each of the spokes represents a separate database structure. This could be hosted on a separate database server or server cluster to scale out the architecture. These servers need to be backed up, monitored, patched, and maintained. They need business continuity and disaster recovery plans and components. The more moving pieces there are in a system, the harder it is to maintain.

### In-Database Analytics

Although data marts tend to isolate data from the primary data in the data warehouse, many organizations are discovering that keeping the analytic data structures within the enterprise data warehouse is very beneficial. James Kobielus states,

> Visionary organizations are adopting an emerging practice known as "in-database analytics," which supports more pervasive embedding of predictive models in business processes and mission-critical applications. With in-database analytics, enterprises migrate their predictive analysis (PA), data mining (DM), and other compute-intensive analytic functions from separate, standalone applications to execute in the enterprise data warehouse (EDW). Doing so allows IT professionals to leverage the EDW's full parallel-processing, scalability, and optimization features. In-database analytics can help enterprises cut costs, speed development, and tighten governance on advanced analytics initiatives. Business process and applications (BP&A) professionals should implement in-database analytics in conjunction with ongoing efforts to consolidate and scale their EDW (Source: James Kobielus, *In-Database Analytics: The Heart of the Predictive Enterprise*, Forrester Research).

From an operational and design perspective, it is much simpler to build and manage all the components of the enterprise data warehouse on a single platform. It provides a single system to monitor and manage. It simplifies the security model and helps move consistently and quickly between the ODS, dimensional models, and analytic structures to find the correct data on which to report. Breaking down the data warehouse into component pieces is a mechanism devised to mitigate the limitations of physical server implementation.

## Approaches to Large Scale Data Warehouse Physical Architectures

The simplest data warehouse to implement is a single system that can host all the elements of the data warehouse, from staging through ODS, dimensional data, and analytic structures. When data is small enough, this can be handled on a single server. As the data grows, there is a need to scale up or scale out to meet the increasing demands placed on the data warehouse.

### Single Server Systems

As single databases are simpler to manage, so are single server systems. Mainframes are still available and they can, indeed, handle vast amounts of data. Microprocessor-based systems continue to grow in capacity. Systems with multi-sockets that host multiple core central processing units can host as many as 128 individual cores. The density of RAM allows more and more memory to be added to these boards. And operating systems (OSs) continue to improve the management of threads and memory address virtualization. If a design starts on a single server database, there is often a route to increase the capacity of the hardware on which it runs.

Of course, there is a limit to how far any system can scale on a single server. Problems rapidly multiply. It becomes harder to provide enough power to the system. The more capacity that a system has in a small space, the more heat it generates. To overcome these issues, more costly hardware is added to provide more power, better cooling, hot-swappable components, and other similar hardware innovations. Nevertheless, the demand will eventually exceed the capacity of a single server.

As the hardware gets larger and more capable, its cost grows at an even greater rate. A single motherboard with 8 sockets and 64GB RAM often costs more than two servers with 4 sockets each and 32GB RAM. The larger the system gets, the higher the price tag relative to its total processing power and storage capacity.

Large single servers are also expensive to protect. To prevent having a single point of failure, an organization needs at least two of them. If the relative price of a single server is high, doubling it to ensure that the organization can get to the information they need when they need it simply doubles the pain.

Once a large expensive server is purchased and commissioned, most enterprises want to get their money's worth from the system. With less expensive, commodity servers, it is reasonable to replace them every 3 to 5 years. With the ongoing affect of Moore's law, the replacement servers are much more capable and have the same or lower relative cost of the replaced servers. However, mainframes seldom get replacement upgrades. Few organizations purchase a 32 socket server and then replace it 3 years later. Large scale systems are often trapped by the high cost of replacement. As these systems age, they become more expensive to operate because the cost of replacement parts go up, OSs go out of main support, and fewer and fewer people know how to keep these systems functional.

Scaling up can be a cost-effective strategy in the short term for data warehouses that start small and do not grow significantly. This option must be carefully considered for its long-term costs over the course of years.

## Federated Database Implementations

To facilitate database scale out, most database software provides some form of distribution or federation. This allows several database servers to work together to manage the same data. By scaling out, many of the issues encountered with single server systems can be addressed:

- Economical, commodity servers can be used as the nodes in the database servers
- As multiple servers are involved, they can be clustered and provide redundancy within the system; a single server failure need not bring down the entire data warehouse
- Additional servers can be added as demand dictates; thus, the cost of the system grows only as the demand grows
- Server maintenance can be applied to individual nodes on a rotating basis, reducing downtime for the entire database for software patches and upgrades

There are several approaches to database federation. Each approach helps address some specific type of bottleneck within the system.

The most common form of distribution is to separate the server that stores the data from the servers that respond to queries. This can be done through replication or the creation of read-only databases. Most commercial database products support some version of this model.

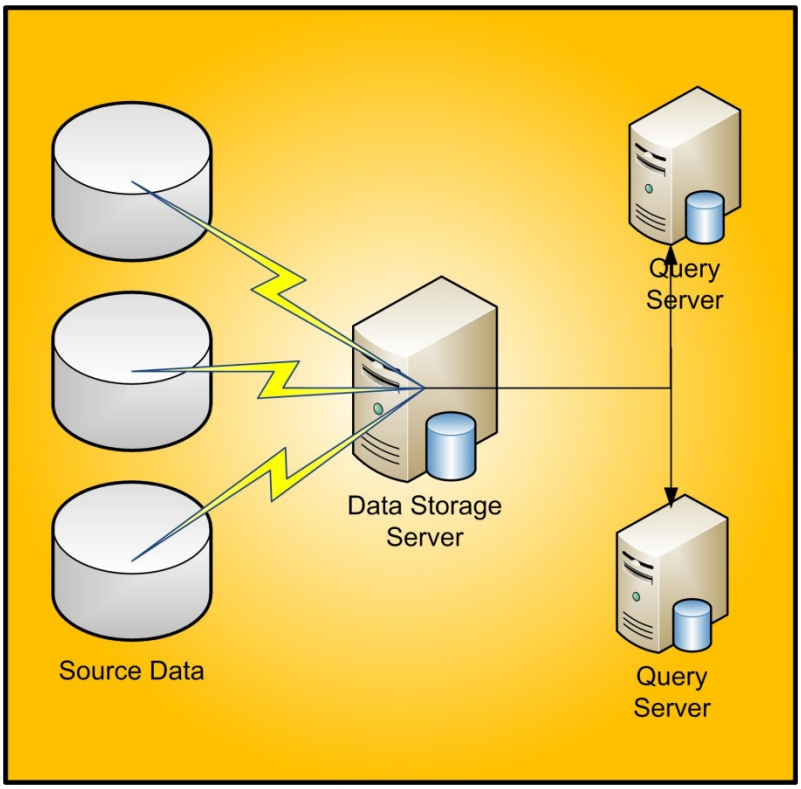


**Figure 2.2: Using query servers.**

The approach allows the separation of the workload of storing data from responding to queries. It can allow data to be loaded into the data handling database without disrupting the query servers. However, it does not solve the problem of helping the databases handle very large data sets. The data storage server becomes a single point of failure for the system. It adds to the complexity of the system by requiring the synchronization of data throughout the system. It also adds a degree of latency to how soon data can be seen in reports.

Another approach is partitioning the data across multiple standard database servers. Each server manages a portion of the data. This approach breaks the database into smaller, more manageable pieces. This allows the database to be balanced across multiple servers and keeps the amount of data each server handles small enough that economical hardware can be employed. Data modification is done as a transaction across all the servers in the federation, and queries are divided across the federation to provide comprehensive results.

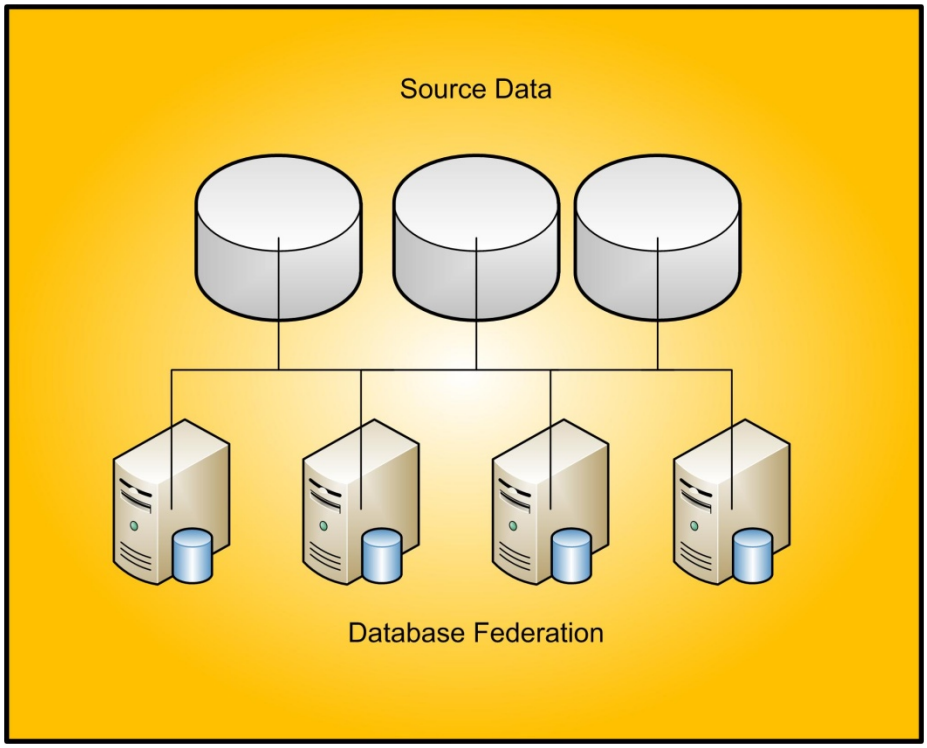


**Figure 2.3: Federation using standard database systems.**

This method was employed by many database software publishers to provide a scale-out approach to their database systems. Although it is conceptually sound, the approach has a number of implementation issues. The servers must update using distributed transactions. This is much slower and more apt to fail than single server transactions. The queries from the servers must be collated into a single result before they can be delivered, and this affects the performance of the federation. If any server in the federation goes offline, the database is effectively offline because the entire database cannot be queried.

They are complex to operate and maintain. For example, if you discover that the partition would work more effectively by changing the key from a geographical partition to a date partition, the database will need to re-distribute the data across all the partitions. This is time consuming and requires system downtime.

## Massively Parallel Processing Data Warehouse Systems

The concept of a federation of databases where each database server handles a portion of the data evolved into the concept of the massively parallel processing data warehouse system. This system also divides the data into logical partitions that can each handle a portion of the data. But rather than using general purpose software for the individual nodes, nodes are specialized to handle specific aspects of the data warehouse.

Some nodes are used to stage and load data into data warehouse storage. The data is partitioned into smaller sections, each section easily handled by the nodes. The data can be partitioned in advance by the data loading nodes and staged to write into storage as quickly as possible. The data loading nodes have no other purpose, so resources used to load data will not affect the query performance of the database.

The storage nodes each manage the data within their own partition. They store data as loaded from the data loading nodes and manage queries. The amount of data handled by any given node is kept relatively small, so commodity hardware can be used in these nodes. This helps keep the overall cost of the hardware down.

The data is stored in a manner that is optimal for retrieval. In a transactional system, it is better to store records in random locations to spread data across the disk and prevent "hot spots." Data warehouse data is better organized for serial reads where similar data is written congruently on the disks. This can allow the storage arrays to be optimized to support serial loading and reading of the data.

Other nodes are specifically designed to handle queries. They break the query into the component portions and distribute the relevant portions of each query to the affected storage nodes. They then collate the result into a single cohesive dataset and return it to the requestor. This helps break up the impact of any given query and allows scale out of the work to gather the appropriate data.

Other nodes specialize in data backup. This allows most of the burden of copying the data to be moved from the individual storage nodes and helps minimize the impact of backing up the data while the system remains in service.

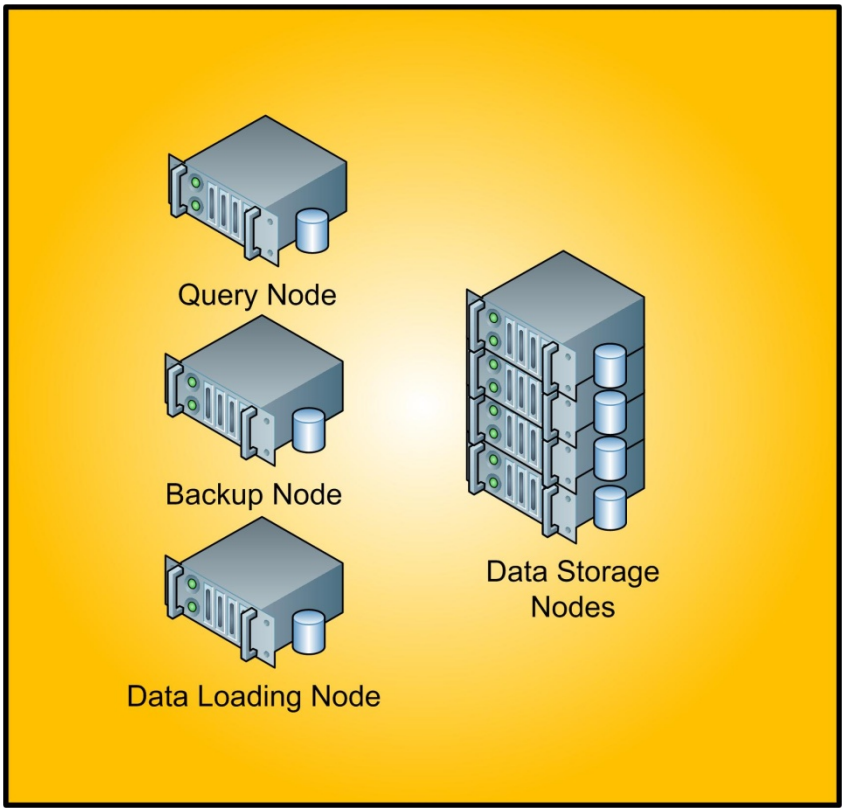


**Figure 2.4: Component nodes in a massively parallel processing data warehouse system.**

The system can implement spare nodes that can be brought online if one of the individual nodes within the system fails. This makes the system much more robust without the full expense of duplicating all the hardware to provide redundancy.

Massively parallel processing data warehouse systems employ software specifically designed to manage the workload across the various component nodes. It makes the system, which is composed of a number of individual server nodes, behave as a single, cohesive system. This helps reduce the burden of management for the system and makes it simpler to operate.

The purpose of a massively parallel processing system, then, is to allow a very large data set to be built as a single logical database and deployed as if it were operating on a single server. The system itself is built of multiple nodes, each being a relatively economical commodity server. The system can be expanded by adding nodes as they are required to meet the growing demand. The system helps overcome many of the challenges inherent in building large scale data warehouses.

## Approaches to Large Scale Data Warehouse Software

When choosing software and hardware for a large scale data warehouse, one must carefully consider the correct approach. Whether purchasing software from a single publisher or a publisher and its partners or assembling a system by acquiring best of breed components for each portion of the system, one must carefully map the tradeoff before an intelligent approach can be determined.

The common components of a data warehouse system typically include the following functional areas:

- ETL software
- Database storage engine—including engines that store the staging and relational tables as well and the OLAP and data mining structures
- Reporting tools
- Analytical tools for working with the analytical data structures; these often include data visualization software

### Single Vendor Stacks

Many software publishers work to provide an end-to-end solution for the needs of their customers. They offer everything from file-based database systems through large scale data warehouse systems and reporting and analytical tools. Virtually all the large, well-known players in the field can lay claim to having something to fit the need of every possible user.

There are some clear advantages to working with a single vendor stack. If a single vendor can be held responsible for all the components in the data warehouse suite of software, there is the proverbial "one throat to choke" when the system does not work properly. Most vendors will help scale from their smaller systems to their larger system as the demands on the system increase. This can allow an organization to start smaller and grow into a large scale data warehouse over time.

Of course, the theory of how a single vendor should work is seldom the way they actually work. Many single vendor stacks are assembled through a series of acquisitions. Although all the components may ostensibly be in place, they may not actually work or play well together. If the vendor stack is hobbled together through a mismatch of loosely coupled software, it may not perform as promised.

The individual software must be measured on its merits. Most vendor stacks are better at some things than others. Clearly defining the needs of the organization and quantifying those needs will help determine whether the software will satisfy the demands of the business. A system that can handle 2000 queries per second may not help if it takes 10 hours per day to load and 4 days to recover if a backup needs to be restored. Although the reputation of the vendor and their support model are very important considerations, at the end of the day, the system needs to deliver to the organization on every level—data loading, query time, backup and data security, and support. There is no room for compromise.

## Third-Party Support

Many software vendors build for the larger markets and enlist the aid of partners to provide specialized software to meet the more granular needs of specific customers. The third-party software is often built to fill a particular gap. Often this software meets the need of the market niche quite well.

Finding experts who publish software to meet a specific need can help extend the value of a vendor's software. It can often help fill-in the feature or performance gaps that make specific software unsuited for a need. There are some considerations that should be made when thinking about third-party software, however.

Third-party vendor tools may not integrate into a product as well as the vendor that wrote it. They simply do not have the same level of insight into the internal workings of the base software. Unless the two organizations work in careful lock-step, changes in the versions of either software can cause issues. Often, third-party software publishers do not have the same robust support systems as larger software publishers. This can create a ragged support model. A new service pack on a database system can, occasionally, cause the third-party reporting software, or ETL, or backup software, to fail. This can cause gaps in deploying software patches and require additional testing time before software patches are released.

> **Note**
>
> Open standards for database connectivity, mentioned later in this chapter, can ease many of the issues of working with third-party tools. The willingness of the database publisher to maintain and support those standards should be considered when choosing a vendor.

The viability of the third-party organization must also be considered. If the organization cannot continue to grow and succeed, they may abandon a product. Occasionally, third-party software companies may be bought by competitors and support for the product in the form used in the data warehouse may be discontinued.

Most third-party software is an excellent way to extend the functionality of a base data warehousing system. Before moving in this direction and committing to software from multiple organizations, the purchasing enterprise should consider all the options and risks.

## Best of Breed Software

The next extension of the third-party software concept is the best of breed concept. The functions listed as components in creating a data warehouse and analytic system are very different. Because a company can write a good database engine does not mean that they can write an excellent ETL system. A reporting system that can work with both the source transactional systems and the data warehouse may provide a more consistent reporting experience and help consolidate costs for licenses throughout the enterprise.

Seeking companies that specialize in a particular aspect of the data warehouse system can help improve performance. Finding the best system for each functional area of the warehouse can help ensure that the enterprise gets the best efficiency from each sub-system and optimize performance. If a company concentrates on one thing so that they can be the best at that thing, the results can be remarkable.

Of course, many of the issues with third-party vendors are magnified with best of breed software choices. Some software simply works better with other software. Although an ETL vendor will want their product to work with every database on the marketplace, it will have initially been designed to work with one or two specifically. It will always perform better with those systems.

The more software solutions that are placed together in a system, the more issues may arise as one of the vendors within the solution upgrades. The software patch cycles and upgrade cycles grow increasingly complex as additional publishers work through their own product release cycles.

Product support can also become more challenging. Although most software vendors will work hard with any organization to integrate with other manufacturer's products, there is always some level on which they can declare that it is the other vendor's issue.

Also consider the viability of best of breed software publishers as independent organizations. Recently, many data warehouse and analytical systems have been purchased by larger software publishers. As a software publisher absorbs and continues to develop the best of breed product for their platform, it may not work as optimally on a third-party data warehouse. In other terms, it may no longer be "best of breed" for your database platform.

Best of breed software solutions can provide the optimal solution in performance and cost effectiveness. As long as the risks are understood and mitigated along the way, this can provide a performant, viable data warehousing solution.

### Open, Shared Standards for Data Interaction

Data warehouse subsystems that adhere to shared database standards provide a great deal of protection when designing a data warehouse solution. Although proprietary protocols are often more performant in the short run, they can limit options in the long term and may close the door to alternative approaches to data warehouse software solutions.

A system built on commonly accepted standards, such as Open Database Connectivity (ODBC), can allow software from many sources—Java, .NET, COBOL and other platforms—to exchange data. These standard interfaces help systems work together and allow those who employ them to use many different options to work with their data and deliver it to a wider range of clients in a more tailored and appropriate format.

A strong common shared standard for database interaction can allow the database to be constructed as a "black box," with the hardware and software configured and optimized for this use. The hardware and software implementing the database become abstracted by the data interaction layers. This allows the data warehouse to be configured in an optimal manner, and changed as required, without affecting the source systems that feed the data warehouse or the reporting applications that consume the data.

Large data warehouses are unique structures. Although they are built on the same underlying principles as smaller transactional and analytical databases, they have unique challenges and require flexibility in approach and design to overcome those challenges. A large scale data warehouse will have a large diversity of source systems and more distinct types of clients than smaller systems. This makes adherence to common, open standards more important. The large scale data warehouse needs to interact fully with the diverse range of enterprise systems without compromising its own internal need to meet the demands placed upon it. Implementing open standards will help it rise to that challenge.

## Approaches to Large Scale Data Warehouse Security

Enterprise data warehouses contain the wealth of information that the entire organization has gleaned, all put in one place. That data may contain personally identifiable information (PII), corporate and trade secrets, marketing intelligence, financial records, and a wide range of other important information. Making that information available to the right users while keeping it away from malefactors is critical to the success of the warehouse itself.

All of the information in the data warehouse is not for the eyes of each individual who can access the data warehouse, so the system must engineer some form of data entitlement. That data entitlement must be easily maintained and must extend throughout all the elements of the data warehouse system. The data must also be protected to ensure that it can be restored in a timely manner in the event of a disaster.

## Data Entitlement

The data warehouse should be designed to allow certain users access to some data while allowing different users access to other data. This means it requires some form of user access control. Such control is typically manifest in data entitlement tables. These tables indicate which users or groups of users can access which levels of data. To successfully implement this level of control, each query must be evaluated to determine whether the inquirer has permission to see the data that they request.

Some systems provide an internal mechanism that grants permission on certain rows or certain fields in the tables. This security mechanism is invoked as a portion of each query. To use these systems, database roles are created. Each role is restricted to accessing only the data to which its users are entitled. To maintain these systems, the data to which a certain role is entitled must be maintained and membership in the role must be updated to ensure that only authorized users are members of the role.

The same type of system can be employed with user access tables maintained within the database. Each query includes a join to the user access table that will indicate which data can be accessed. This system requires the identity of the user to be defined as a parameter within the query. Also, the queries must be protected so that no one can access any data without submitting their identity and joining to the user access tables.

Part of the security of the data warehouse system must include an evaluation of how users are identified by the system and the support the system provides for implementing data entitlement. Architects who do not plan for appropriate data entitlement in the beginning may discover it to be quite a problem going forward as new data sources are added, new government and corporate regulations are enforced, or changes in the organization rapidly shift who can or cannot see portions of the data within the warehouse.

### Maintaining Security in Depth

Beyond data entitlement, the security of the data must be considered at every phase of the data storage. An unfortunately common mistake is to export sensitive data to the data warehouse over an unsecured FTP channel and then leave it as an unencrypted file on a file share until the ETL system imports it. PII is left in unencrypted columns in the database. Data entitlement is carefully organized within the ODS, but the OLAP cubes and data mining structures may require a different security structure. It is important to maintain security through the system to make certain each data structure and its individual derivatives are adequately secured.

Most database designers do not think in terms of threat analysis. They do not look for the ways in which a database can be breached, list the threats in order of likelihood, and then develop a plan to mitigate the threats. The news frequently warns of databases that have been hacked, credit cards stolen, Social Security numbers compromised, and identities stolen.

The large scale data warehouse will often contain the most sensitive and high-risk information within the company. Understanding the support mechanisms for securing data through encryption, secure ETL, and data entitlement in every phase of the data life cycle can help protect the data from incursion and compromise.

### Data Protection and Business Continuity

The data warehouse provides vital information to the entire organization. The cost of not having that data available is often incalculable. Data warehouses need to be designed to run continually without interruption and designers must plan for disaster.

The first consideration is system maintenance and downtime. Any system requires maintenance windows during which the entire system will not be available. The shorter those windows, the more costly the data warehouse system. Before committing to a system, the acceptable downtime should be evaluated and factored into the price of the entire system. Because of the size and capacity of these systems, the cost of downtime is magnified.

The second consideration is system backup. It can take years to populate a data warehouse and once they have matured, it may be impossible to re-construct the data therein contained from source data. Backups can have a major impact on the performance of the system and potentially create expensive maintenance and performance windows within the operation of the system. The impact of backups on the operation of the system should be planned into the system from the onset. Systems that offer high-performance backup and restoration solutions can help provide additional options when planning for disaster recovery.

System resiliency must also be considered. Where are the single points of failure and how can they be parallelized or protected? Using mirrored systems and clustered servers can help reduce risk, but this option can be expensive to implement and maintain. The total cost of the system must include the standby and redundant components that ensure that the system will meet the SLAs of the business.

## Summary

This chapter has reviewed many of the approaches to designing a large scale data warehouse and analytical system. The next chapter in the series will address the operational challenges faced by large scale data warehouse implementations.