15

DATA MINING AND DATA WAREHOUSES

DESCRIPTION

Analysis of large data sets comes at a cost. The cost:benefit ratio is the level of complexity of the analysis versus the compute infrastructure (hardware, software, network, etc.) required to support the analysis. As the complexity of the analysis increases, more extensive infrastructure is required. I will present data warehousing as a use case study to emphasize that the business is not realizing the full potential of the data in the data warehouse. This is due to many reasons, including the physical limitations imposed on the warehouse's ability to perform queries and/or analysis. Examples are in the areas of fraud detection and customer pattern analysis.

There are many ways to construct a data warehouse and methods in which to perform analysis. One method is to query and extract data out of the warehouse and perform the analysis on a system external to the data warehouse. Traditional methods for increasing performance may require the thin distribution of data across increasing numbers of disks or require custom hardware and software combinations that are proprietary to the data warehouse vendor. These alternatives contribute to the increase in both the cost and risk.

The data grid architecture offers an alternative to traditional data mining and warehousing infrastructures. Not all data grid implementations lend themselves to this use case, only those that meet specific criteria, the details of which will be discussed in the following sections.

USE CASES

The topic of data warehousing is rich in research, products, tools, and methodologies. The purpose of this section is not to include all aspects of the topic but to foster a new perspective on the subject that will evolve into new data warehousing techniques so as to increase value with both lower cost of ownership and risk reduction.

At the highest level of the block diagram, a data warehouse is a collection of servers and disks. Data are distributed across multiple disks for storage capacity and can be arranged in specific patterns to increase speed of access. Via the data warehousing servers, a client application can query, perform some level of analysis, and receive the results. Figure 15.1 highlights this high-level process.

The following discussions will focus on business applications querying the data warehouse and performing further analysis on the data. The purpose of data grids in a data warehousing solution is not to replace the data warehouse but to augment it in such a way that will increase the query and analytical performance of the overall process. Therefore, I will not address EAI- and EII-related issues of how the data warehouse is initially populated and updated from the raw datastreams throughout an enterprise.

For this discussion, the workflow model is simple. We will have two branches of processing for this example, with one branch making some level of analysis performed within the data warehouse and the other branch with minimal to no analysis performed in the data warehouse. This processing difference defines the following two use cases:

Use Case 1. Business logic/analytics are performed in the data warehouse.

Use Case 2. Business logic/analytics are performed outside the data warehouse.

These two cases are illustrated in the flow diagram in Figure 15.2.

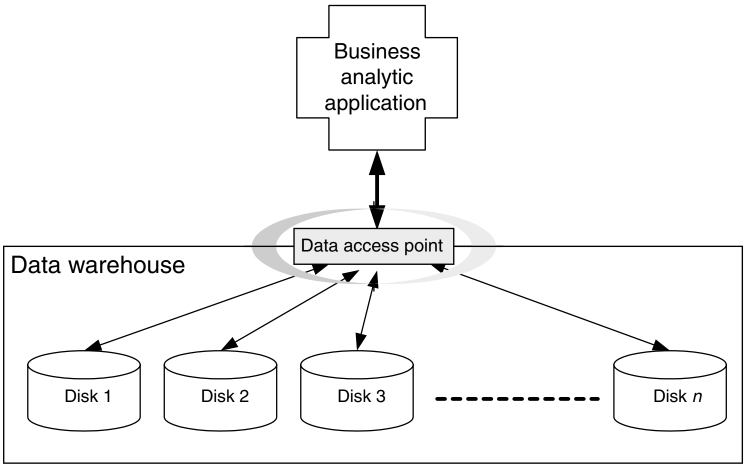

Figure 15.1. Data warehouse high-level block diagram.

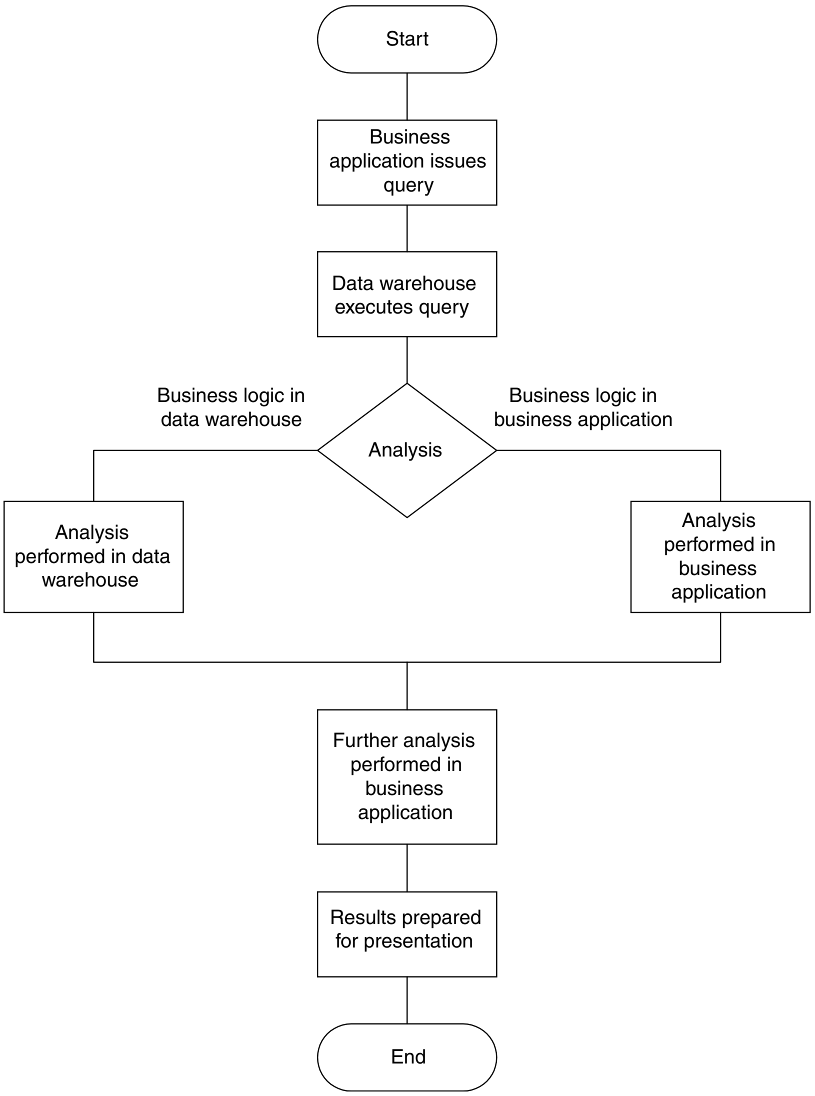
Start
Business application issues query
Data warehouse executes query
Business logic in data warehouse
Analysis
Business logic in business application
Analysis performed in data warehouse
Analysis performed in business application
Further analysis performed in business application
Results prepared for presentation
End

Figure 15.2. Process workflow defines use case definitions.

The type of implementation of the data grid will determine its viability to augment a data warehouse. Data grids that support the following features are candidates, for example

- Bringing the data close, in a networked proximity, to the compute nodes where the business application is running
- Offer full data grid management support
- Data query capability

Certainly level 0 data grids do not meet these criteria and thus will not apply. Some types of level 1 data grids will not apply, either.

An example of a level 1 data grid that will dovetail with a data warehouse is one whose implementation is an "in-memory model" that spans a compute cluster or the compute grid. The benefits of such data grids are performance since the data are kept close to the processing and must offer a distributed memory space that increases capacity as the number of machines in the grid increases.

The economic and logistical concern of staging large volumes of data in a data grid are not daunting hurdles to jump. Both concerns center on the number of machines or nodes required in the data grid to hold large volumes of data. The physical size of the data grid or the number of nodes in the compute grid is dependent on the memory capacity of a single machine of a node and the total data size extracted and analyzed from the data warehouse. Since one of the economic driving forces behind grid computing is to reuse the large numbers of inexpensive machines, the cost of large grid infrastructure is comparatively low in comparison to the large servers that they will be replacing.

The logistical concerns are addressed via new generations of provisioning and management software that automates much of the system administration processes. Therefore a data grid infrastructure is an economical means to dramatically increase the performance and analytical complexity of large data sets.

GENERAL ARCHITECTURE

In both use cases the general architecture is the same. The data grid sits in between the business application and the data warehouse. The business logic operates directly against the data grid. Independent of the use case, the architecture and business application's workflow is the same; the only change is to the capacity (therefore the size) of the data grid. As data capacity requirements increase, the size of the data grid must also increase. The new architecture with the data grid is illustrated in Figure 15.3.

The data grid spans the compute space of the business application, thus bringing the entire scope of the data warehouse (depending on the use case) directly into one continually addressable and ready-to-access data space. The advantages for time to process are realized in the ability to pivot data in complex structures and hierarchies in near-in-memory speed, without having to swap data in and out of a local memory space and files system/database, and independent loading of data from any one or multiple data sources.

In this context, the term "data warehouse" refers to the traditional data warehousing infrastructure in place in most organizations today.

First Use Case

In this use case, the data need to be loaded into the data grid. Most if not all of the data of the data warehouse are reflected in the data grid. Thus, any query and

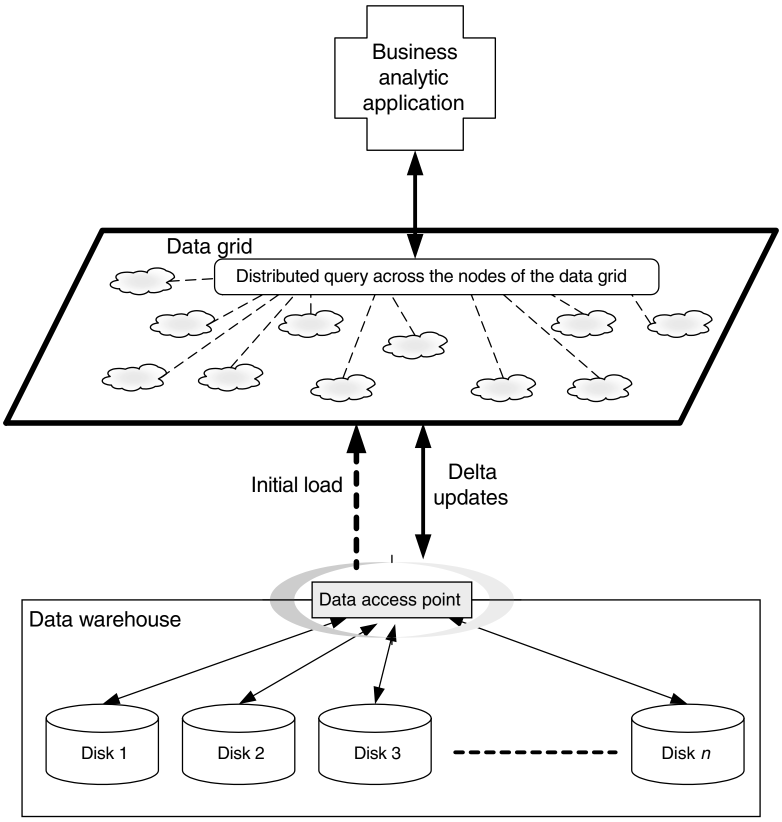

Figure 15.3. Data warehousing with a data grid.

subsequent analysis of the data are run directly against the data grid, not the data warehouse. This implies a one-time data load with periodic data synchronization between the data warehouse and the data grid. Any data integration from external sources into the data warehouse can remain as is (into the data warehouse) or loaded in parallel to both the data warehouse and the data grid. For our discussion, we will assume the former. The EII data load into the data warehouse and subsequent synchronization process into the data grid is independent of and transparent to the business applications. The result is two copies of the data warehouse, one in the data grid for analytics operation and the other in the data warehouse for long-term persistent storage.

As discussed earlier, the class and implementation of the data grid for this use case yield benefits to the business from a cost perspective, reduced processing time, and increased OLAP complexity. Specific potential areas of benefit are increased query speed, savings on analysis processing time, increase the number of business analytics that can be run, and increase the next level of complexity of the analytics that could otherwise not be achieved. However, as with all things, there are tradeoffs. This architecture will expose the risk of large data loss within the data grid. Should enough of the data grid "go down," full or partial data loads could occur, therefore requiring the need to perform data upload from the data warehouse. On such a failure, the required time to reload and recover the system is dependent on the amount of data to be reloaded, the network bandwidth between the data warehouse and the data grid, as well as the network bandwidth within the data grid itself.

The architecture of the data warehouse and data grid transfers the responsibility for direct data access and querying from the data warehouse to the data grid. Therefore, this minimizes the requirements for the expensive data warehousing infrastructure and augments it with a lower cost, faster, and more powerful data grid infrastructure. The cost–benefit difference lies in the data grid's ability to support large quantities of data with an infrastructure that is inexpensive to grow and maintain.

Second Use Case

Whereas the first use case is a dual data repository of data warehouse and data grid, this use case is more of a hybrid approach. The initial queries are run against the data warehouse, returning a small subset of data (as compared to the total size of the data warehouse), which is then loaded into the data grid. Once these data are loaded in the data grid, the same benefits of the data grid are realized in the first use case. These benefits include savings on processing time, increase in the number of business analytics that can be run, and increase in the complexity of analytics. On the other hand, the disadvantages of this approach are the increase in the overall time to perform the business application since the data warehouse is involved in the initial query and load of the data into the data grid as business requests occur. However, the downsides are minimized; a catastrophic failure of the data grid will not require a costly reload of the entire data warehouse. Also implied is the size of the data grid infrastructure required to support the OLAP process since an increase in data capacity implies an increase in the number of nodes of the data grid. In summary, the first use case implies a large data grid infrastructure just to support the data warehouse as compared to the second use case, where only a subset of the data warehouse must be stored in the data grid.

The data load process of this use case can be managed in a number of ways: (1) by the business application/data warehouse, (2) by the data grid/data warehouse, and (3) using a method similar to that used in the first use case. If multiple business applications require similar data sets, then keeping those data continually loaded and synchronized with the data warehouse makes sense. If data loss occurs in the data grid, the data load and recovery time is not as catastrophic since it is a subset of data that is required from the data warehouse.

The shifting of fast data access and querying for the data warehouse to the data grid are not as dramatic as in the first use case, but are present nonetheless. In addition, other architectural, processing, and financial benefits are realized from the business application perspective. If the analytic process runs outside the data warehouse and in the memory/processing space of the business analytic, then the data extracted from the data warehouse must be stored somewhere else. When using traditional methods (non-data-grid), there are only a few alternatives; try to fit the data set into the RAM (random access memory) of the user's machine or put the data into a file on the user's machine. Depending on the size of the data, they may not fit in the available RAM, thus eliminating this option in many cases. File sizes also have their limitations. Data may have to be split across multiple files and be swapped in and out of RAM as the business analytics requires. This adds an upper bound to the amount of data that can be extracted. In addition, engineering will need to be involved for managing and splitting the data across multiple files.

The use of a data grid eliminates the need for engineering efforts around large data sets. The data that are extracted out of the data warehouse would easily fit within the data grid, circumventing the need for the application to perform the extra file loading, splitting, and swapping. As a result, the analytics in turn would simply run faster and be available for other nodes and applications if required.

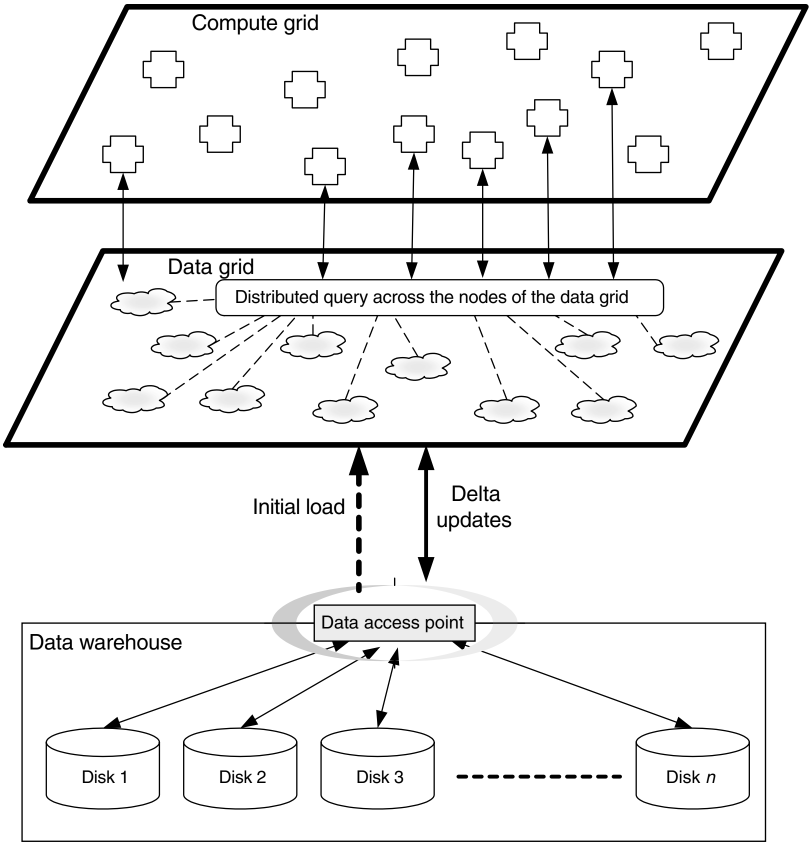

Figure 15.4. Data warehouse enhanced by commute and data grids.

Enter the Compute Grid

Up to this point we have discussed how the data grid can benefit an analytic application. The performance of the business application can be enhanced even further by running the OLAP process in a compute grid. The benefits gained will depend on how well the OLAP process of the business application can be parallelized across a compute grid (recall the discussions in Chapter 14, on calculation-intensive applications). In these instances the benefits are manyfolds: (1) the benefits of the data grid–data warehouse combination as described above, (2) the ability to split a serial process into worklets running in parallel across the compute grid, and (3) the turboboost (if you will) of adding data affinity to the mixture. Figure 15.4 illustrates the new architecture that evolves with the compute grid and the data grid for the data warehousing solution.

DATA GRID ANALYSIS

For the data grid analysis of data mining and data warehouses, I will use the application definition expressions as introduced in Chapter 5. For both use cases discussed above, the general architecture is very similar; thus the general equations will also be similar. However, within each use case there are numerous variations on implementation that are dependent on the specific environment in which the use case is to be deployed. These differences in environment and implementation will appear in subareas of the application definition expressions and in the data management policies. Therefore, the expressions shown below are a template; it is left to the reader as an exercise to architect in each case given the variations in definition and policy expressions:

- The application definition equation for a distributed environment is

$$OLAPProcess\begin{pmatrix} Work(\ ), \\ Data\_output(\ ), Data\_S1(\ ), \ldots Data\_Sn(\ ), \\ Time(\ ), \\ Geography(\ ), \\ Query(\ ) \end{pmatrix}$$

where

$$Work(atomic, nonsynchronous)$$

The output data surfaces are small in comparison to the data surfaces that will be analyzed from the data warehouse.

$$Data\_output(``x" \ kbits, ``y" \ bits, transactional, nontransient, queryable)$$
$$Data\_S1(``z" \ Tbits, ``k" \ Mbits, nontransactional, transient, queryable)$$
$$Data\_Sn(``z" \ Tbits, ``k" \ Mbits, nontransactional, transient, queryable)$$

The ability to run complex analysis over large data sets in short periods of time provides the business with a better view into the prevailing economic forces and consumer demands of the current environment. Armed with better and in-depth quality views of the "bigger picture," the business can target specific consumer groups and manage manufacturing and supply chain in finer detail otherwise not possible. This OLAP process is to run in the confines of a single data center and the applications requirement to analyze (complex queries) of any of the data sets is not essential to the business.

$$\begin{aligned}&Time(NearReal\text{-}Time)\\&Geography(DataCenter, 1GbitEthernet)\\&Query(complex)\end{aligned}$$

- Data management policies are expressed as

$$DataDistributionPolicy = DDP\left(\begin{array}{l}DataWarehouse\_DDP,\\DWRegion,\\Scope(ALL),\\Pattern\left(Automatic, Random\left(\begin{array}{l}DWDDPPattern,\\WhiteNoise(\,),\\NULL,\\NULL,\\NULL,\\NULL\end{array}\right)\right)\end{array}\right)$$

The data replication policy shows a lower number of replicas per data atom since the overall size of the data in the data grid is quite large. Each replica increases the overall storage capacity requirements of the data grid by the size of the data loaded from the data warehouse. The downside to a lower replication size per data atom is resilience of the data in case of failure. These tradeoffs must be considered in each use case and architecture variation.

$$DataReplicationPolicy = DRP\left(\begin{array}{l}DataWarehouse\_DRP,\\DWRegion,\\3,\\Scope(ALL)\end{array}\right)$$

$$SynchronizationPolicy = SP\left(\begin{array}{l}DataWarehouse\_SP,\\DWRegion,\\Scope(Boundary(\text{``}intra\text{''}), NULL),\\Transactionality(\text{``}nontransactional\text{''}),\\LoadStore(List(\text{``}DataWarehouse\_DLP\text{''}), NULL),\\Events(NULL)\end{array}\right)$$

Below are three scenarios for data integration (EII) and data consistency between the data warehouse and the data grid. Starting with the data source it can update both the data warehouse and the data grid simultaneously. Assuming that there is already an integration of data source to the data warehouse, this would imply a second integration path into the data grid. If EII is achieved via a middleware bus, such as a queuing system, this may be the preferable path. Alternatively, the EII into the data warehouse remains unchanged, leaving the data warehouse to manage synchronization with the data grid, possibly via event/trigger mechanisms. The third alternative is to have all data sources input directly into the data grid, and leverage the data grid's event processing to manage synchronization with the data warehouse. The example presented here assumes that the EII functions of the data warehouse remain within the data warehouse. Should one choose to transfer the EII responsibility from the data warehouse to the data grid, then the data grid event notification policies will need to be established to maintain data consistency to the data warehouse.

$$EventNotificationPolicy = N/A$$

The data load policy is required only in the second use case, and if the implementation offers the business application the data grid as the medium to instantiate queries to the data warehouse. Then the data load policy will manage the data warehouse interface for data query and loading result sets into the data grid as defined in the equation below:

$$DataLoadPolicy = DLP\left(\begin{array}{l} DataWarehouse\_DLP, \\ MCRegion, \\ Granularity(Grouping(1), N/A), \\ DataWarehouseAdapter(\,) \end{array}\right)$$

$$DataStorePolicy = N/A$$

The QoS–application requirement quadrant graph in Figure 15.5 shows application characteristics of nontransactional complex data analysis that are large in size.

BENEFITS AND DATA GRID SPECIFICS

The benefits of applying data grid technology to a data mining/data warehouse application are twofold: the speed of processing and the complexity of the process. The use of the data grids and the compute grids enable complex analysis to be performed in a comparatively shorter timeframe. Therefore, this opens the door for the feasibility of creating increasingly sophisticated analysis programs and running more of them. These advances in business possibilities and sophistication are not

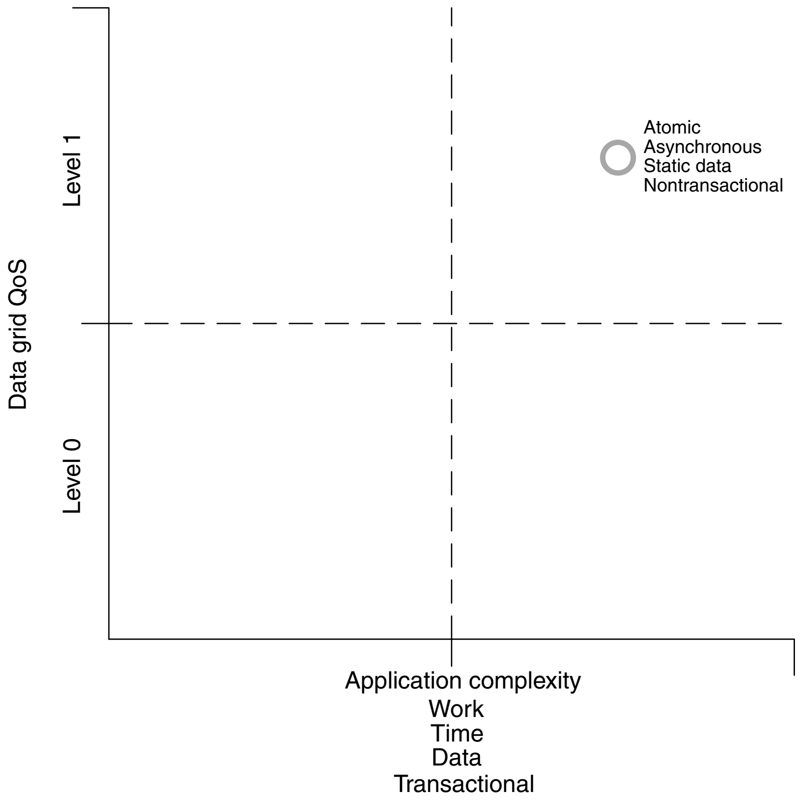

Figure 15.5. OLAP processing with a data warehouse and data grid.

possible without a data grid infrastructure. Using an implementation without the data grid architecture would expand to your traditional data warehouse with more disks and CPUs to perform queries and/or analytics but would still be unable to achieve the levels of complexity that you would desire and need with the ever-increasing business requirements.