Mixed, Shifting, and High-Concurrency Workloads in Data Warehouse Systems

A Whitepaper

Author:
Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

July, 2012

Sponsored by

Table of Contents

1 Management Summary

Many data warehouse systems contain massive amounts of duplicate data, consist of numerous databases, and deploy various database technologies. In most situations, data warehouse architects have opted for such an architecture to be able to handle the mixed, shifting, and high-concurrency workload that is typical for today's data warehouse systems. However, this duplication of data, and proliferation of databases and technologies make a data warehouse system complex, increase costs, and reduce flexibility. This whitepaper discusses the importance of deploying database technology that can handle a mixed, shifting, and high-concurrency workload and thus minimize, the need for storing duplicate data, and deploying many databases and a multitude of technologies.

The architecture of many data warehouse systems is characterized by a large number of databases, the storage of massive amounts of duplicated data (a large data footprint), and the deployment of a multitude of database technologies—including classic SQL databases, multi-dimensional databases, and data warehouse appliances. The dominant reason for designing such an architecture is usually to improve the performance of queries.

Data warehouse systems are characterized by a large number of databases, the storage of massive amounts of duplicated data, and the deployment of a multitude of database technologies.

This proliferation of data, databases, and database technologies, has these disadvantages:

- The extra costs of storing and updating duplicate data
- Managing and tuning multiple databases,
- The costs of training developers on the multiplicity of database languages and concepts
- Reduced flexibility (undesirable for the time to market)
- The risk of inconsistent data
- A potential decrease of the data quality level

Once, the workload of data warehouse systems was relatively homogeneous; the same types of queries were being executed repeatedly. Due to the changing and expanding reporting and analytical needs of business users, the characteristics of the workload transformed significantly. Users asked for more complex reports, new forms of analytics, more queries, and so on. In addition, the introduction of operational BI, where users require access to (close to) 100 percent up-to-date data, resulted in systems in which data is refreshed more frequently.

From a database perspective, all these new requirements have resulted in a "mixed query workload." Different types of queries *and* data loads must be processed concurrently. For most database servers, it's hard to run such a mixed workload efficiently and predictably. They can handle some query workloads concurrently, but not all of them. One reason is that some database servers have been designed and optimized to process only a few types of query workloads.

What is greatly needed is the ability to simplify the architectures of data warehouse systems. One solution is to deploy database servers that are capable to meet the users' reporting and data requirements of today and tomorrow. Such a database server has to meet these three requirements:

- It must be able to process a *mixed* workload in an efficient and stable way (addressing a wide range of query types).
- It must be able to handle a *shifting* workload (a planned or unplanned increase or change of query workload).
- It must be able to support a *high-concurrency* workload (processing many queries and other database operations concurrently, even if they are of different types).

This whitepaper explains how the BI reporting workload has changed over time and resulted in the need for database servers capable of processing this new mixed, shifting, and high-concurrency workload. It also describes how the Teradata® Database has been designed to support a modern-day BI workload. It also focuses on Teradata Active System Management (TASM), the module that offers DBAs the advanced instruments to manage and control the workload on Teradata Databases. The paper ends with three short case studies. All three of these customers have a large data warehouse environment and all run a modern-day workload. They all benefit from the features offered by TASM to manage the workload.

The BI workload of today requires database servers capable of processing a mixed, shifting, and high-concurrency workload.

2 The Dream – One Large Enterprise Data Warehouse

In an ideal world, all the database applications retrieve data from and manipulate data residing in one data store in which all the data is stored only once, and all the production applications and reporting applications would use one and the same database. Such a simple architecture is the dream of every IT department.

In fact, the reason why database servers were introduced a long time ago, was to get rid of the dispersion and duplication of data in countless files that were becoming an obstacle to the further expansion and maturation of IT systems. Back in 1977, Chris J. Date, one of the founders of the relational model, wrote the following in his best-seller *An Introduction to Database Systems*[1]:

> *An enterprise should store its operational data in an integrated database to provide the enterprise with centralized control of its operational data. … This is in sharp contrast to the situation that prevails in most enterprises today [1977], where typically each application has its own private files … so that the … data is widely dispersed.*

Most of those files are gone by now. Currently, enterprise data is predominantly stored in databases. Unfortunately, it's still distributed and duplicated, though not over files anymore, but over numerous databases. Many organizations struggle with their plethora of applications, databases, and systems that they have developed and bought over the years. The notorious "silos of applications" remain a reality and a constraining factor for making their IT systems ready for the coming decade.

The dream is to have one database, the enterprise data warehouse, in which all the data is stored only once.

The same dream applies to data warehouse systems. Figure 1 represents the architecture of a data warehouse system that all organizations would like to have: an architecture consisting of one database, the *enterprise*

[1] C.J. Date, *An Introduction to Database Systems*, second edition, Addison-Wesley Publishing Company, 1977.

data warehouse, in which all data is stored only once, and from which all the reports retrieve their data. The dream of every data warehouse architect is to have a single database, and if it isn't, it should be.

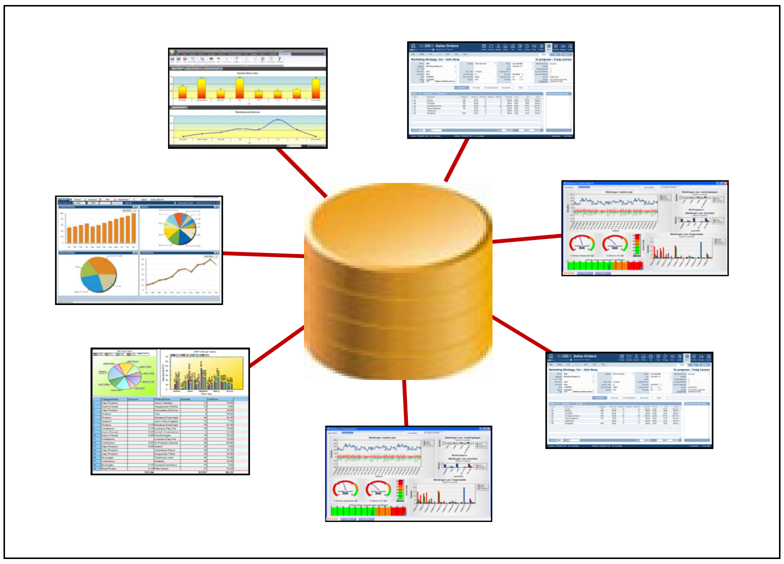

Figure 1 *The dream is to have one database that handles all the data needs and in which all the data is stored only once*

3 The Reality – A Labyrinth of Databases and Database Technologies

The dream may be to have one data warehouse, but the reality for most organizations is different. Unfortunately, the same kind of proliferation that has been created for production systems is also a reality for data warehouse systems. Most current data warehouse systems look like the diagram in Figure 2: a *labyrinth of databases*.

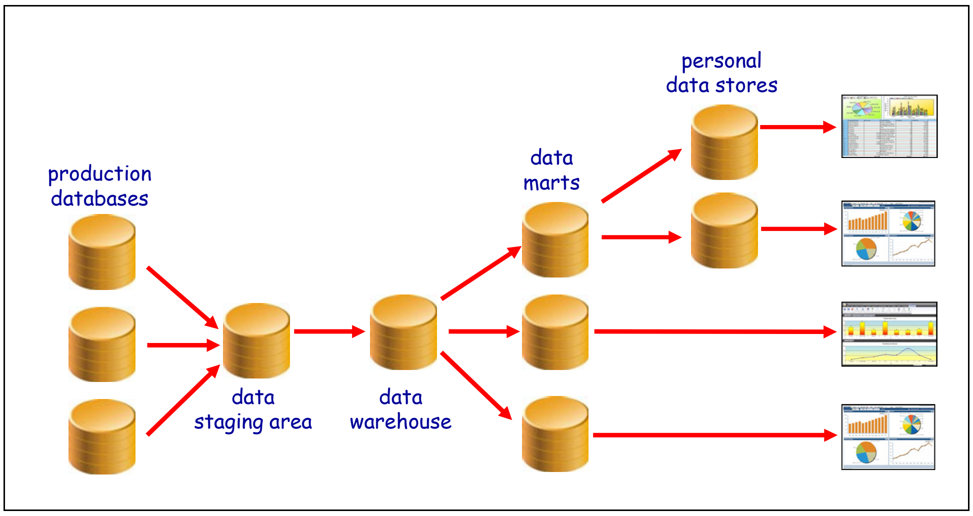

Figure 2 *Most data warehouse systems are like a labyrinth of databases*

In such a labyrinth, data is stored redundantly in multiple databases, and multiple database technologies are used.

Duplicate Data and Multiple Databases – As indicated, in many data warehouse systems, data is stored redundantly. For example, data stored in an enterprise data warehouse is also stored in data marts, and possibly in a set of personal data stores, as well. Usually, these *derived databases* contain a subset of the data coming from the data warehouse and the data is often slightly aggregated. In other words, they contain duplicated data. And even more redundancy can exist in data warehouse systems, in the form of, for example, materialized views and pre-calculated aggregated data stored in data warehouses.

Most often, derived databases, such as data marts, are developed for query performance. There are a variety of reasons why query performance can be improved by introducing data marts. First, the design and contents of the tables in these databases are aimed at the queries executed by the reports accessing them. Second, by creating a data mart, a part of the query workload can be offloaded from the data warehouse, which positively impacts the performance of all concurrent queries. It's like opening up more registers at a store to reduce the waiting time for customers.

Besides the data marts that are developed for large user groups, many personal data stores are also found in data warehouse systems. These are commonly developed for just one or two users and with tools such as Microsoft Access and Excel. Many newer self-service BI tools, such as Qlikview, Spotfire, and Tableau, also create their own copies of the data. Still, these personal data stores, with their redundant data, are also created to improve query performance.

The overall effect of storing redundant data is that (in many data warehouse systems) the real amount of stored data is much higher than the amount of original data, resulting in a *large data footprint*.

Many data warehouse systems have a large data footprint.

Specialized Database Technologies – Distributing and duplicating data is not the only technique used to improve query performance. Another popular technique is to deploy *specialized database servers*. For example, in many data warehouse systems, *multi-dimensional database technology* is used for the data marts and/or personal data stores. In multi-dimensional database servers, data is not organized in tables and columns, but in cubes, dimensions, and hierarchies. Usually, such database technology is deployed for developing data marts and personal data stores if users need fast and iterative OLAP reporting. For example, Microsoft Analysis Server is popular for developing these derived data stores in which the data is stored in multi-dimensional cubes that are accessed by reports using a database language called MDX.

Specialized database servers are deployed to improve query performance.

Another example of deploying specialized database servers is when *data warehouse appliances* and dedicated SQL database engines are installed to support analytical environments in which large data sets must be scanned repeatedly. This can be an environment where *data scientists* use high-end analytical tools to create new models and to find unknown patterns and trends in the data.

In fact, the arrival of *big data applications* has led to the introduction of new data store technologies, such as Hadoop, MongoDB, and Cassandra, usually referred to as *NoSQL database servers.* These products specialize in, for example, storing and analyzing extremely large amounts of semi-structured and unstructured data.

Summary – The storage architectures of many of today's data warehouse systems consist of a labyrinth of databases containing a considerable amount of duplicate data and are built up using a multitude of database server technologies. In most cases, query performance is the main reason to opt for data duplication and deploying multiple databases and database technologies.

4 The Disadvantages of Duplicating Data, Running Multiple Databases, and Deploying a Multitude of Database Technologies

Unquestionably, storing redundant data in multiple databases and implementing specialized database technology improve query performance, and probably offer some other advantages, as well. However, this approach also its disadvantages:

Disadvantages of duplicating data (and a large data footprint)

- Extra costs of redundant data storage
- Extra costs for updating the redundant data (if the data in the source database changes)
- The risk of getting inconsistent data, and subsequently inconsistent reporting results
- Potential decrease of the data quality level because of the existence of multiple copies

Disadvantages of having multiple databases (and database servers)

- Extra costs for managing, tuning, and optimizing multiple databases
- Extra costs for retuning databases that may be needed for workload shifts or temporary peaks
- Extra costs for developing and maintaining ETL programs to periodically refresh redundant data
- Decrease of flexibility (for example, changes in source tables structures will require additional changes in target table structures and data integration code)

Disadvantages of deploying a mix of database technologies

- Extra training costs regarding management and tuning of various database technologies
- Extra training costs to give developers experience with multiple database languages and their concepts
- Complexity of migrating reports from one database server to another
- Extra costs for developing and maintaining multiple backup/recovery/fallback procedures

BI systems in which data is not or minimally duplicated, where a limited set of databases is used, and where only one database technology is used, are less expensive, more flexible, and easier to manage.

Summary – Regarding databases and database servers, the old rule "less is more" is very true. A data warehouse system in which data is not (or minimally) duplicated, where a limited set of databases is used, and where (preferably) only one database technology is used, is less expensive, more flexible, and easier to manage with respect to workloads.

5 Today's BI is a Mixed Workload

Good query performance is crucial for the success of a data warehouse system and is directly related to *workload*. The faster queries are executed, the more queries can be processed within a certain period of time. In other words, the faster queries are executed, the bigger the workload a data warehouse system can handle. This section focuses on workloads and explains how the workload of data warehouse systems has changed over time from homogeneous to mixed.

Prior to the turn of the century, the workload on data warehouses was relatively homogeneous—that is to say, the same type of queries was being executed repeatedly. Most of these queries involved data filtering and some aggregations, and they accessed a limited number of tables. On the loading side, data was usually loaded at night or over the weekend when few reports were run. In other words, there were two types of workloads: one query workload consisting of a rather homogeneous set of queries executed during daytime, and one data-load workload executed at night. These two workloads didn't clash, because they were executed at different times. Tuning the database server, therefore, was relatively easy. It was like tuning an engine for a specific type of work, work that would be done over and over again.

Over time, however, because the reporting and analytical needs of business users expanded and changed, the workloads that a data warehouse system had to process changed significantly. More complex reports, new forms of analytics, more queries, and so on, had to be processed. Eventually, this led to a new *mixed query workload*. In addition, the introduction of *operational BI*, where users want to have access to (close to) 100 percent up-to-date data, led to systems where data had to be refreshed more frequently; instead of once every night, data had to be refreshed every 10 or 20 minutes. The consequence of this was that the static query workload became a *mixed workload*, where different types of queries *and* data loads were processed concurrently.

For most database servers running a mixed workload efficiently and in a stable fashion is difficult. They can handle some query workloads concurrently, but not all of them. One reason is that some database servers have been designed and optimized to process a few types of query workloads. An analogy may be minivans that have been designed for transporting small loads, but are not optimized for transporting large loads or driving 200 miles an hour. A second reason is that some database servers can be tuned and optimized in only one way. They can't be set up for a mixed workload, nor can the tuning easily or automatically be adjusted for another workload.

Therefore, to get the required performance levels, many data warehouse architects made the decision to distribute the query workload over multiple databases—such as operational data stores, data marts, and personal data stores—whereby each database could be tuned for a particular query workload. In addition, multiple databases make it possible to deploy specialized database technologies that are usually designed for a specific workload. This way, performance requirements can be met—but it also leads to the labyrinth of databases and results in the disadvantages outlined in Section 4.

Many database servers don't support a mixed workload and can't be tuned for different workloads, resulting in data warehouse systems with a large data footprint.

Summary – Due to the fact that many database servers do not support a mixed workload and can't be tuned for multiple workloads, data warehouse systems have a large data footprint and deploy a multitude of database technologies with the associated costs, complexity, and failings.

6 The Diversity of Today's BI Reporting and Analytics

The evolution of BI hasn't stopped. New forms of analytics and reporting appear every day. To show what a modern BI workload looks like and to illustrate the extent of the diversity, this section describes some of the known and newer forms of reporting and analytics requested by today's business users. The next section presents the characteristics of the query workload belonging to each of these reporting forms.

Executive Reporting – This is the most classic form of reporting. Business users have a predefined set of queries they can select from through some form of menu or by clicking a button. In most cases, the queries fired off by these tools access a restricted set of tables, don't access that much data, and include simple aggregations and summarizations. Users can't really change their existing reports nor can they develop new ones. Dashboards—which are much more than reports—are the modern manifestation of this requirement.

OLAP – With OLAP (online analytical processing), users have more dynamic reporting capabilities and can define their own queries and dashboards. These reports are not fixed. In most cases, the queries executed by these tools access a restricted set of tables and don't access that much data. The results show the data in an aggregated form, and they require either none or very simple analytical processing.

Operational Reporting – Operational reporting and analytics refer to forms of reporting and analytics in which the data shown to the users has to be close to 100 percent up-to-date (sometimes referred to as operational data, zero-latency data, or real time data). This paper uses the term *operational data* to refer to 100 percent up-to-date data.

Analytics – Analytics is about discovering yet unknown trends and relationships hidden in data. Usually the result of analytics is a model for forecasting, predictive modeling, or optimization. Statistical and data mining algorithms are used to get a better understanding of the business. A typical workload for analytics consists of complex queries, large sets of data being scanned, and complex processing.

Sandboxing – In the world of BI, the term *sandbox* refers to a stand-alone and somewhat isolated environment set up for analysts and data scientists to study data and find answers to unclear or poorly defined questions. In a sandbox, any form of analysis or reporting can be executed without disturbing other users. The analysts can change table structures or they can run queries that can take hours of processing, and so on. It's a playground. It's not uncommon to throw the sandbox away when analysts have their results.

Embedded Analytics – Embedded analytics implies the execution of analytical models implemented in production systems. In most cases, these analytical models are designed and developed off-line, and the resulting forecasting or predictive model is implemented inside the system. These embedded models can be used, for example, to check the correctness of incoming data, optimize business processes, predict cross-selling opportunities, and detect fraudulent activities.

Big Data Analytics – A new form of analytics is called, quite rightly, *big data analytics*. Many traditional information systems exist that store and manage large numbers of records. More recently, new systems have been built that store an amount of data magnitudes larger than those in the more traditional systems. Examples are click-stream, sensor-based, and image processing applications. They all generate massive numbers of records per day. The amount of records here is not measured in millions, but sometimes in trillions. Analyzing this amount of data always leads to a resource intensive query workload.

Unstructured Data Analytics – There is a lot of valuable information hidden in many unstructured and semi-structured data sources. Emails, tweets, contracts, and URL strings may all contain a wealth of information. This form of analytics focuses on analyzing those data sources.

360-Degree Reporting – Some business users need a 360-degree view of certain business objects, such as customers, flights, and bank accounts. To create a 360-degree view, it's not sufficient to present data from the production systems and/or data warehouse, because they probably contain only structured data coming from the production systems. What is needed is for data from all kinds of data sources to be brought together, from both unstructured and structured sources, and from internal as well as external sources. Being able to show a full picture of a specific concept is called *360-degree reporting*. In most cases, 360-degree reporting is done on data's lowest level of detail.

Transaction Reporting – These are the queries executed by production systems to retrieve some data. In most cases, it's detailed data on one or more business objects. This form of reporting is commonly handled by production systems and not so much by data warehouse systems.

Two of these forms, operational reporting and embedded analytics, are discussed in more detail in the next section.

7 Operational Reporting and Embedded Analytics

In this section, we focus on two of the previous forms of reporting: operational reporting and embedded analytics. Both are forms of operational BI. These new forms deserve extra attention, because they present a technological challenge when implemented.

Operational reporting and embedded analytics present a technological challenge.

Many examples exist in which operational BI is needed. For example, a retail company might want to know whether a truck that is already on the road to deliver goods to a specific store should be redirected to another store with a sudden, more urgent need for those products. It would not make sense to execute this analysis with yesterday's data. Many other application areas exist, such as a call center where historical data and 360-degree profile data has to be shown instantly on the screen when a customer calls. Another example is the rerouting of passengers when a flight has been cancelled where several aspects are studied, including the loyalty points of customers, the possible mishaps with customers, and the optimum costs to the airline. All these forms of analysis only make sense when executed on operational data.

It's clear that for operational BI, business users must have access to operational data. A challenge is that most data warehouses offer only once-a-day or –week refresh rates, thus not containing operational data. Another challenge is that in classic data warehouse systems, reports are 'far away' from the database where the operational data is entered. For the data, the road from the source system to the reports is a long one, because it has to pass through many databases before it arrives in the one used for analysis.

The technological challenge of operational BI is always related to getting new data that is entered in the production systems to the database used for reporting and analysis. How do we enable this without clashing or destabilizing the ongoing query workload? Operational BI is difficult to implement, because it almost always implies a mixed workload—queries and inserts occurring concurrently.

8 The Relationships between Reporting Forms and Query Workloads

Each form of reporting and analytics has its own query workload characteristics. One generates massive amounts of data scans. Some only pick a few rows from a limited set of tables. A few do a lot of complex analytical computations. Others need to access large amounts of data from many tables. In addition, the workload of some reporting forms is very predictable, while that of others isn't.

Table 1 illustrates the differences in query workload generated by the BI forms described in Section 6.

	Complexity of Queries	Amount of Data Accessed	Number of Tables Accessed	Complexity of Analytical Operations	Predictable Query Workload	Predictable Set of Queries	Repetitive Queries
Executive Reporting	Low – Medium	Low – Medium	Low – Medium	Low	Yes	Yes	Yes
Classic OLAP / Reporting	Low – Medium	Low – Medium	Low – Medium	Low	No	Yes	Yes / No
Operational Reporting (BI)	Low	Very Small – Small	Low	Medium	Yes	Yes	Yes
Analytics (forecasting, predictive modeling)	High	Large	High	High	No	No	No
Sandboxing	High	Large	High	High	No	No	No
Embedded Analytics	Medium	Medium	Low – Medium	Medium	Yes	Yes	Yes
Big Data Analytics	High	Extremely large	Low	High	No	No	No
Unstructured Data Analytics	High	Large	Low	Medium	No	No	No
360 Degree Reporting	Low	Large	High	Low	No	Yes	Yes
Transaction Reporting	Low	Very Small	Low	No	Yes	Yes	Yes

Table 1 *An overview of the characteristics of the query workload generated by different forms of reporting and analytics*

9 Requirements for a Database Server

Section 4 describes the disadvantages of duplicating data, implementing multiple databases, and using multiple database technologies. What is needed is the ability to simplify the architectures of data warehouse systems. Again, the dream is to have one database that supports the entire BI workload, thereby reducing the proliferation of databases and deployed database technologies.

A solution is to select a database server that is able to meet users' reporting and data requirements—today and tomorrow. Such a database server has to meet these three requirements:

- It must be able to process a *mixed workload* in an efficient and stable way.
- It must be able to handle a *shifting workload.*
- It must be able to support a *high-concurrency workload.*

These key requirements are described in the next three sections.

10 Requirement 1: Mixed Workload

A *mixed workload* means that queries to be processed have very dissimilar characteristics. Such a workload may contain queries ranging from simple ones (retrieving a few rows from one or two tables and doing little aggregation work) to queries for which many large tables have to be scanned and joined, and complex statistical operations have to be executed. A mixed workload could also mean that at the same time updates, and deletes have to be processed as well.

A mixed workload means that queries and other database operations with very dissimilar characteristics have to be processed.

For a database server to support a mixed workload efficiently, it must be tuned and optimized. Tuning is always aimed at the expected workload. Unfortunately, most database servers can only be tuned for one workload. For example, they can be tuned to handle massive scans of data, for direct access to select small portions of table rows, for large volumes of transactions, or for batch data loading. This is not very different from Formula 1 racecars that can be tuned to perform well on tracks with many long straightaways, or to perform well on a winding track—it's one-or-the-other proposition.

Database tuning is normally done by assigning values to parameters that all somehow influence the database server's performance. Many different parameters exist. Some determine how much internal memory (buffer) and how many processors the database server is allowed to use. There are also those that influence where data can be stored, while others indicate the portion of the buffer that is reserved for locking.

Usually, the values of these parameters are fixed. They can be changed, but the new values become effective only after the database server has been stopped and restarted. Having non-changeable parameters is acceptable when the workload is stable and predictable. However, in more and more data warehouse systems, it's not—it's a mixed workload (see Sections 6 and 7).

Practically, what this means is that for a database server to support a mixed workload, it should allow DBAs to set parameters that determine how the database server should divide its resources over different queries and different types of queries. For example, a DBA should be able to indicate that queries entered by one user or group should have a higher priority than those entered by another, that long-running queries should get less priority than simple queries being processed concurrently, and that unplanned and resource-intensive queries are cancelled if they use too many resources.

11 Requirement 2: Shifting Workload

Most production systems have a very stable and, therefore, predictable workload. The number of users and transactions might increase over time, but it's highly unusual for a production database to suddenly have a different and unexpected workload. This doesn't apply to data warehouse systems. As discussed, the workload of data warehouse systems has always been much more fluctuating, unstable, and unpredictable. The new BI requirements for reporting and analytics have only added to this.

Here are a few examples of how the workload of a BI system can shift:

- New workloads—The new forms of reporting and analytics (listed in Section 6) can lead to new types of queries, and therefore to a new type of workload. Especially now that more users can use self-service BI tools that allow them to create new reports, new workloads can come into existence overnight.

- Expanding workloads—If the business-user community expands due to the success of the data warehouse system or the organization itself, the data warehouse system has to be able to grow accordingly. It should have enough capacity to seamlessly adapt to the expanding workload.

- Temporary new workloads—Some of the new workloads can be temporary. A well-known example is sandboxing, where data scientists need a subset of the data warehouse data for a short period of time for deep analytics. When the exercise is completed and a result obtained, this workload will just disappear.

- Peak workloads—It's not uncommon to tune a database server for the average workload. This is usually sufficient if the real workload fluctuates around that average. But some organizations can have peak workloads that are much higher than the average. Such a peak can jam the whole database server, thus ruining the performance of the entire workload.

- More complex workload—If users start to understand the data they have access to—if they fully understand what they can do with it, and if they understand their tools better and better—the effect is always that their queries become more complex. This does not automatically lead to *more* queries, but to more *complex* queries.

What is needed is database technology that adapts itself to a *shifting workload*. Like a chameleon adjusts its skin color to the changing environment, a database server should adjust its parameter settings to the changing workload. It should be able to change the parameter settings dynamically based on the changes of the workload. In other words, it should be able to tune itself constantly.

A shifting workload means that the query workload changes over time in an unplanned and planned way.

If we make an analogy with cars, what is needed is a car that can add seats by simply pushing a button (for a shift of workload), and that can also morph into a van, a truck, or maybe even a train as needs change. With cars, however, we'll have to wait a few years for such technology, while some database servers already do support mixed and adjustable workloads.

Technically, this means two things. First, to support a shifting workload, it must be possible to set different parameter settings for different moments during the day, week, or month. For example, on each weekday between 8 p.m. and 5 a.m., the priority for complex queries should be lowered in favor of the batch-oriented queries and data loading. Another example is that at month end, the reports of the CFO department must be favored to close accounting, or the priority of complex queries and dashboards must be boosted on Monday mornings. So, for different time periods, it must be possible to have different parameter settings.

Second, if the workload shifts unexpectedly, a database server should be able to adapt accordingly. Usually, the way to do this is by assigning priorities that help a database server to determine how to divide its resources over the new workload. For example, a server should know that if the number of long-running queries that require a lot of I/O suddenly increases, they should be delayed and access to resources should be limited, because they should not interfere with the queries coming from an embedded analytics environment that are part of an operational environment.

12 Requirement 3: High-Concurrency Workload

The previous sections describe how important it is that database servers know how to handle a mixed and shifting workload. However, a mixed workload could be a diverse number of queries and tasks that do not stress the system. For example, if a particular data warehouse environment executes only 10 queries a day and each one has different characteristics, it still qualifies as a mixed workload. The term mixed workload only indicates that different types of workloads are executed. It does not imply that queries are executed concurrently; every database server can easily run the 10 queries of that mixed workload serially. (Although, a problem in this example may still be that different query types may need different tunings, and when the tuning is wrong for particular queries, it leads to poor individual query performance.)

The keyword here is *concurrent*. Besides being able to handle mixed and shifting workloads, it's essential that a database server deployed in a data warehouse system knows how to efficiently run the queries of a saturated mixed workload *concurrently*. In fact, a workload could be so intense, that many queries and other database operations have to be executed concurrently. This is called a *high-concurrency workload*.

A high-concurrency workload means that many queries and other database operations have to be executed simultaneously.

With a high-concurrency workload, resources, such as CPU time, I/O, and memory, become scarce. In this case, a database server must be able to distribute its resources equally and fairly. It should prevent, for example, certain queries from monopolizing the entire database server, and thus consuming the majority of the resources. This can slow down of all the other queries. It should also prevent low-priority queries from interfering with high-priority ones. It should also divide the resources correctly when data loads are executed concurrently with OLAP-type queries. And the database server should never spend more time on its internal administrative tasks than on query processing due to the large amounts of queries to be processed, which may give users the feeling that the database server has shutdown.

To handle a mixed and high-concurrency workload, a database server should offer DBAs the instruments they need to control how resources are distributed. For example, it should be possible to enter these specifications:

- Queries that unexpectedly consume too many resources should be deprioritized.
- Queries that are part of batch jobs should get less priority than online queries, because there are no online users waiting on the batch queries.
- Operational embedded analytical queries should have a higher priority than executive reporting and OLAP queries.
- Data loads required for operational reporting have a higher priority than long-running, complex queries.

Summary – The architectures of data warehouse systems can be simplified if database technology is deployed that can handle mixed, shifting, and high-concurrency workloads. Such technology should give the DBAs the instruments they need to instruct the database server on how to divide its resources. In a high-concurrency mixed workload system there is less need to duplicate data, develop multiple derived databases, and deploy a mix of database technologies.

13 No Future for Specialized Database Servers?

In the previous sections, the assumption was made that generic database servers that fulfill the three workload requirements, are recommended for data warehouse systems. Does that mean that *specialized database servers*, such as appliances, multi-dimensional databases, graph databases, and NoSQL databases, have no future? The answer is, of course, no. They do indeed have a future.

It's always possible that a specific workload is so intense that, even for a generic database server that supports a mixed, shifting, and high-concurrency workload, this is hard to handle. In this case dedicated and specialized database servers may be the only solution. Some of their vendors have tuned them for such specific workloads. In a way, these specialized database servers are like Formula 1 racecars—they drive very fast, but they are not suitable for any other type of work. They have been built for one workload. Another reason why specialized database servers may be useful is when they can handle a particular workload for less cost. For example, using Hadoop as a storage mechanism for archival data, may make sense from a cost perspective.

But when is a workload so intense that a generic database server can't handle it? What's the breaking point? Even if we are able to identify that breaking point, it's a moving target because with the continuous improvement of hardware and software, it shifts over time. What also impacts the breaking point is that user queries keep increasing in complexity and volume, and the amount of data they analyze keeps growing. So, maybe a generic database server is sufficient today, but it may not be when the workload increases.

Summary – The need for specialized database servers will always remain, especially with a continuously growing BI workload. However, data warehouse architects should be aware of the pros and cons of such a solution.

14 Teradata Corporation

Teradata was founded in 1979. From day one, the company has focused on developing parallel database servers to support data warehousing and business intelligence. In 1983 they shipped their first beta system to Wells Fargo bank, one of their early customers. The decision was made to build the software and the hardware in conjunction so that the former could make an optimal use of the latter.

The current version of Teradata® Database is 14. Today, the product is available as a software solution or as a database machine (meaning software plus hardware). For different types of workloads, special versions of the Teradata Database are available—ranging from the Teradata Active Enterprise Data Warehouse via the Extreme Data Appliance to the Data Mart Appliance—all running the same database software. They refer to this as the Teradata Workload-Specific Platform Family. For quite a number of consecutive years, Gartner has positioned the Teradata Database in the #1 Leaders Quadrant in their Magic Quadrant for Data Warehouse Database Management Systems.

The Teradata Database has been designed to support mixed, shifting, and a high-concurrency workloads. It's not a specialized database server aimed at a limited set of workloads. It's not a platform designed only for query workloads. Instead, it can handle a real mixed workload with diverse queries and inserts going on at the same time, making it highly suitable for operational BI environments. To emphasize this capability, Teradata introduced the term *active data warehouse* a few years ago. In addition, it supports advanced features for the DBAs to keep a high-concurrency workload under control.

Since 1980, Teradata Database has been designed to exploit a parallel hardware architecture. This makes it possible to distribute query and load processing over many processors and distribute I/O over many disks. This is the foundation for its scalability; it makes it suitable to run large workloads on extremely large databases.

The internal architecture of the hardware platform on which the database server runs plays an important role. For years, the discussion has been whether SMP or MPP is the better internal hardware architecture to support a database server running a BI workload. First, the fact that a database server runs on one of those hardware architectures is no guarantee for high performance. The first hurdle to take is always how well the database server exploits the architecture. This is also analogous to racecars: an untrained driver with the fastest Formula 1 car will lose to a highly trained driver driving a Formula 2 car. In other words, "throwing hardware at it" is just half the story. It's also about how the hardware is deployed and used.

By designing and developing the software *and* hardware, Teradata can develop and tune the hardware including high bandwith I/O storage from NetApp for the database server. Because the hardware is used only for database processing, database processing is not disturbed or unbalanced by other applications. Additionally, the database software knows more about the hardware setup, making it possible for the database software to have more control over the workload. Compare this to, for example, a cloud solution where the database server has no or limited knowledge of the location and distribution of the stored data.

A crucial module for the Teradata Database is a module called TASM (Teradata Active System Management). This module is the key instrument for DBAs to keep the processing of the workload under control, even if it's mixed, shifting, and/or high-concurrency. Queries can be classified based on common characteristics, such as the source of the query, the tables it's accessing, the processing style, and the utility executing a task. Queries can be prioritized based on business needs. Exceptions can be specified so that when a query is expected to consume too much resources, it's rejected, its processing is delayed, or its priority is demoted.

The utility called Teradata Viewpoint allows the status of each Teradata Database server to be monitored live. This portal-based system management tool shows the health of each database server. It can also show whether query goals are being met. But most important, DBAs can easily see whether the settings they entered have had the correct impact on handling the workload.

Summary – Given its support for mixed, shifting, and high-concurrency workloads, Teradata Database is ready for the new BI requirements demanded by today's and tomorrow's data warehouse systems. Using Teradata Database, less need exists to develop additional derived databases, such as data marts and personal data stores; it minimizes the data footprint, and stops the proliferation of different database technologies in a data warehouse system.

15 Three Short Case Studies

This section contains three short case studies of organizations that all faced challenges with respect to their BI workloads. All have used the Teradata Database for implementing their data warehouse environment.

Large Financial Organization – This financial organization switched to Teradata more than 7 years ago. Currently, the entire data warehouse environment totals 100 terabytes, consisting of more than 10,000 tables and hundreds of thousands of columns.

The architecture of their data warehouse system consists of a non-persistent operational data store in which data is kept for a maximum of 30 days. In the enterprise data warehouse, data has different refresh rates. The highest refresh rate is 30 minutes, implying that the data latency is at maximum 30 minutes and that data is loaded into the data warehouse every half hour. The environment contains data marts that are not developed as separate databases, but are defined in the same database environment as the data warehouse itself. In other words, they share the same database platform.

Some data marts are implemented as views on the data warehouse tables and others as physical tables. The logical design of most tables is normalized, although for performance reasons the physical design of a table may be denormalized.

The query workload is generated by approximately 1,000 concurrent users and on average 50-80 queries are executed concurrently. They process a mixed and high-concurrency workload. All the queries, ranging from simple to complex, are processed by one and the same Teradata Database server.

TASM is heavily used to make the platform handle the workload. Simply put, in the daytime the platform is tuned towards queries and in the evening towards running data load jobs. Priorities are assigned to query categories to handle the mixed and high-concurrency workload in the daytime.

Because TASM allows the DBAs to assign priorities and make the system ready for different workloads, they estimate that they save 25 percent on hardware. This means that without TASM they would need 25 percent more hardware power; this would mean they would have to upgrade to a larger platform much sooner.

To indicate that Teradata is capable of running a non-mixed and high-concurrency workload, as well, they have a separate Teradata platform for running simple queries that are not part of the central data warehouse system, but rather belong to the operational system. This platform processes 30 million queries per day (which is equal to 250 queries per second).

Large Engineering Organization – This longtime Teradata customer has all its engineering and manufacturing data along with other functional areas running on one Teradata platform. All the databases comprising the data warehouse environment reside on that one box. The architecture includes a non-persistent staging area and an enterprise data warehouse. It data from 38 data sources, of which one is a core engineering system. The environment is not just a data warehouse for reporting. The organization considers it a mission-critical system; it's crucial for their day-to-day business operations. If this data warehouse environment is down, the production lines stop within minutes.

The size of the whole data warehouse environment is approximately 40 terabytes, including all the derived and redundant data. Their entire query workload expressed in I/O is 4 petabytes per day. This means that the amount of data retrieved from disk is 100 times the size of the entire data warehouse itself. This is an indication of the data intensity of the query workload being executed. The NetApp storage technology enables this I/O bandwidth-intensive workload.

Most of the tables in the data warehouse are normalized (or lightly denormalized) using base normalized temporal tables. Some tables are primarily developed to deal with recursive queries that are executed on recursive bill of material data structures. In these optimized tables, recursive data is flattened to speed up queries. A typical query would use 5-10 base tables and 1-5 optimized tables.

This organization is running a mixed and high-concurrency workload on this platform. First, 14 different applications make use of this environment, some for more than 18 years. The query characteristics of these applications are very different. Second, for a large percentage of the data, the latency is at maximum 5 to 10 minutes, but some data requires a data latency of a few seconds. This implies that loading and querying take place concurrently. Third, on average, there are 70 queries being processed concurrently and at peak times more than 100 queries run simultaneously. Many of the queries are driven by web services.

They use TASM heavily to stabilize that mixed and high-concurrency workload. Different groups of queries have been identified and priorities and service level goals have been defined for queries. At different times of the day, different settings are active to handle the shifting workload. The customer claims that because of TASM they need approximately 23 percent less hardware. This number is pretty much the same as for the previous case study.

High-Tech Manufacturing Organization – Their data warehouse environment is approximately 45 terabytes and runs on one Teradata machine. Only 1 percent of that is reserved for the staging area. The base tables of the enterprise data warehouse itself occupy a substantial 42 terabytes. They use data marts for a few situations and together these occupy 2 terabytes. The environment is 24x7 operational. Users employ a wide range of tools for reporting and analytics, including Microsoft Excel, Oracle/Hyperion, and SAS/JMP.

Data is loaded in different ways. Part is loaded in real time, giving users access to zero-latency data. A second part is loaded every 10 minutes (they call this micro-batch). A third part is refreshed every hour, and finally, there is data coming from an SAP environment that is loaded every night.

It's a mixed and high-concurrent workload. First, on average, data has a very low latency, meaning that data is constantly being loaded concurrently with the execution of queries. Second, different types of queries are processed concurrently. For example, resource intensive queries coming from SAS have to be executed concurrently with more simple queries coming from Excel.

As with the previous two customers, they are using TASM to prioritize workloads and to balance the mixed and high-concurrency workload.

Because manufacturing and testing their new products generates more and more data, their expectation is that the data warehouse will at least double in size the next four years. They store unstructured data, such as weblogs, but their expectation is that this data source will not grow considerably.

Summary – Despite the sizes of these data warehouse environments, the architecture of each is relatively simple: one database platform. They all have a mixed, shifting, and high-concurrency workload. All three are proof that it's possible to have a rather simple—and thus flexible—architecture and still handle a mixed and high-concurrency workload on a large database. For them, the dream of a single platform is a reality. They see no need to add more databases or specialized database technologies.

About the Author: Rick F. van der Lans

Rick F. van der Lans is an independent analyst, consultant, author, and lecturer specializing in data warehousing, business intelligence, and database technology. He works for R20/Consultancy (www.r20.nl), a consultancy company he founded in 1987.

Rick is chairman of the annual European Data Warehouse and Business Intelligence Conference (organized in London). He writes for B-eye-Network.com[2]. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[3] all published at BeyeNetwork.com.

He has written several books on SQL. His popular *Introduction to SQL*[4], published in 1987, was the first English book on the market devoted entirely to SQL. After more than 25 years, this book is still being sold and has been translated into several languages, including Chinese, German, and Italian.

In recent years he has focused on applying data virtualization in business intelligence systems resulting in his upcoming book *Data Virtualization for Business Intelligence Systems*, which will be released in the summer of 2012.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn (http://www.linkedin.com/pub/rick-van-der-lans/9/207/223) or via Twitter (http://twitter.com/Rick_vanderlans).

About Teradata Corporation

Teradata Corporation is the world's leading analytic data solutions company, focused on integrated data warehousing, big data analytics, and business applications. Teradata's innovative products and services deliver data integration and business insight to empower organizations to make the best decisions possible and achieve competitive advantage. Visit teradata.com for details.

[2] See http://www.b-eye-network.com/channels/5087/articles/
[3] See http://www.b-eye-network.com/channels/5087/view/12495
[4] See http://www.amazon.com/Introduction-SQL-Mastering-Relational-Database/dp/0321305965/ref=sr_1_1?ie=UTF8&s=books&qid=1268730173&sr=8-1