Analytics in the Federal Government

White paper series on how to achieve efficiency, responsiveness and transparency.

Gaining the Performance Edge Using a Column-Oriented Database Management System

by David Loshin
President, Knowledge Integrity Inc.

Realize the full value of government data

With statistical data collection and information access at the center of agency responsibility, learning how a column-oriented data platform can overcome the limitations of traditional databases for complex analytics and reporting may be the answer to your mission achievement.

TABLE OF CONTENTS

SPONSORED BY SYBASE • JANUARY 2010

Introduction

Many different kinds of organizations are increasingly recognizing the potential benefits of how analytic databases can support reporting, strategic analysis, and other business intelligence activities. And as the volumes of data used for analysis increase, the sizes of the data warehouses used to support those business intelligence activities must also grow to meet organizational needs, in terms of size, performance, and scalability.

As organizations continue to employ larger data warehouses for purposes ranging from standard reporting to strategic business analytics, complex event processing, and deep-dive data mining, the need for performance will continue to outpace the capabilities of traditional relational databases. In order to satisfy the need for the rapidly exploding need for analytical performance, an alternate database approach, which begins by storing data oriented by columns instead of rows, has proven to sustain the performance and rapid growth requirements of analytical applications. In addition, the simplicity and performance characteristics of the columnar approach provide a cost-effective alternative when implementing specialty analytics servers to support a wide range of users and query types.

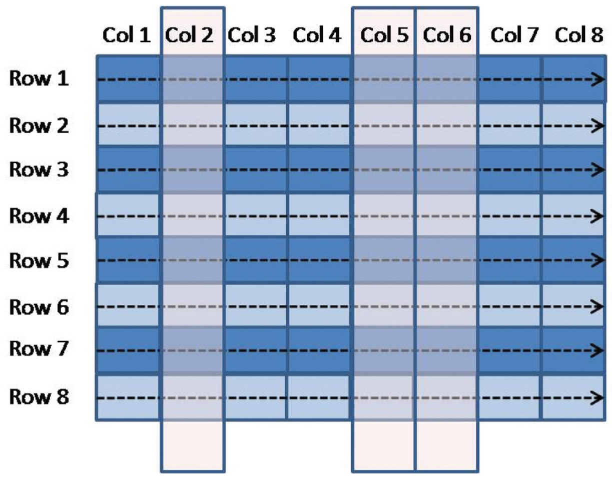
This paper explores the data explosion phenomenon and why column-oriented database systems are able to provide the boost that is needed to gain the performance edge. The paper contrasts row-oriented systems with column oriented systems, and then elaborates the benefits of the columnar approach. A review of how data access latency across the different levels of the memory hierarchy can impact application performance sheds some light on why the column-oriented approach can deliver these benefits, which suggests a number of drivers for increasing analytical application performance, particularly by means of exploiting parallelism. Last, the paper suggests some factors guiding the evaluation process when considering a columnar system.

The data explosion and the challenge of analytic performance

According to Wintercorp's 2005 TopTen Program Summary, during the five year period between 1998 and 2003, the size of the largest data warehouses grew at an exponential rate, from 5TB to 30TB. But in the four year period between 2001 and 2005, that exponential rate increased, with the largest data warehouses growing from 10TB to 100TB. At the same time, the average hourly workload for data warehouses approached 2,000,000 SQL statements per hour, and in cases the number of SQL statements per hour reaching nearly 30,000,000.[1]

1 Auerbach, Kathy, "2005 TopTen Program Summary: Select Findings from the TopTen Program," Winter Corp, Waltham, MA, http://www.wintercorp.com/WhitePapers/WC_TopTenWP.pdf

Figure 1: In a row-oriented database, accessing specific columns requires reading all records



Research indicates that the size of the largest data warehouses doubles every three years. Before considering ways to accommodate the need to consider the emerging volume of unstructured data, trends continue to reflect that growth rates of system hardware performance are being overrun by the burgeoning need for analytic performance. Limiting our scope to only structured data, the rate of data growth is high due to the growing business expectations for reporting and analytics, increased time periods for retention of data, increased numbers of areas of focus for business analytics (it's not just customers anymore!), increased numbers of observations loaded, as well as increasing the number of attributes for each observation. And as organizations also begin to incorporate unstructured data such as audio files, images, videos, etc. the veritable explosion of data becomes an increasingly daunting challenge[2].

Column-oriented analytic platforms—a high-performance approach

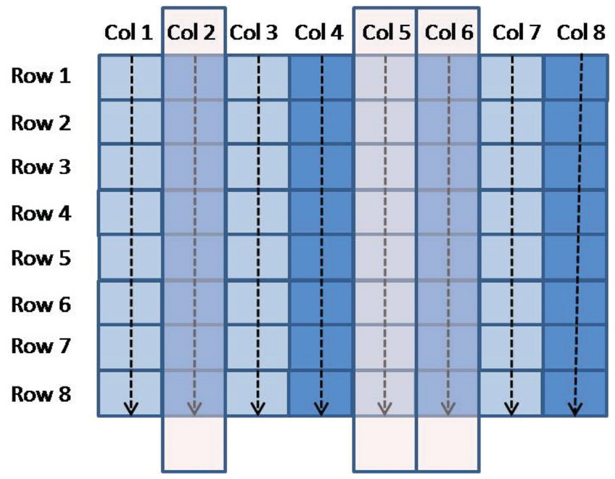
The growing pervasiveness of business intelligence across the organizational spectrum shows a growing reliance on enterprise data warehouses for purposes ranging from standard reporting to strategic business analytics, complex event processing, and deep-dive data mining. And with the rapidly increasing demand for reporting and analysis, the performance and efficiency expectations are likely to outpace the capabilities of traditional platforms. In order to satisfy the need for analytical performance, it is becoming clear that traditional row-oriented relational database management systems (RDBMS) are not be the most effective deployment platform for analytics and related BI activities. What has emerged is a mature, capable alternate database platform, one that manages data by columns instead of rows to meet the cost and performance requirements of the growing enterprise analytic and data warehousing infrastructures.

Row-oriented databases and the analysis challenge

In a typical relational database management system, data values are collected and managed as individual rows and events containing related rows (customer and order for example). This reflects the history wherein most data begins life in transactional applications which generally create or modify one or a few records at a time for performance reasons. Conversely, business intelligence and analytic applications, generated reports, and ad hoc queries often call upon the database to analyze selected attributes of vast number of rows or records, needing only those columns or aggregates of those columns to support the user's needs.

2 Winter, Richard, "Why are Data Warehouses Growing So Fast?," Business Intelligence Network April 2008, accessible via http://www.b-eye-network.com/view/7188

Figure 2: In a column-oriented database, only the columns in the query need to be retrieved



Because of their row-based functions, a row-oriented database must read the entire record or "row" in order to access the needed attributes or column data. As a result, analytic and BI queries most often end up reading significantly more data than is needed to satisfy the request. This creates very large I/O burdens. In addition, the row-oriented RDBMS, having been designed for transactional activities, is most often built for optimum retrieval and joining of small data sets rather than large ones, further burdening the I/O subsystems that support the analytic store.

In response, system architects and DBAs often tune the environment for the different queries by building additional indexes, pre-aggregating data, and creating special materialized views and cubes. These require yet more processing time and consume additional persistent data storage. Because they are often quite query-specific, these tunings only address the performance of the queries that are known, and do not even touch upon general performance of ad hoc queries.

A different approach: Column-oriented data management systems
If the issues associated with poor performance can be attributed to the row-oriented, horizontal layout of the data, then consider the alternative: organizing the data vertically along the columns. As the name implies, a column-oriented database has its data organized and stored by columns. Because each column can be stored separately, for any query, the system can evaluate which columns are being accessed and retrieve only the values requested from the specific columns. Instead of requiring separate indexes for optimally tuned queries, the data values themselves within each column form the index, reducing I/O, enabling rapid access to the data without the need for expanding the database footprint, all while simultaneously and dramatically improving query performance.

Benefits of the column-oriented approach

"With the rapidly increasing demand for reporting and analysis, the performance and efficiency expectations are likely to outpace the capabilities of traditional platforms."

David Loshin

From the simplicity of the columnar approach, accrue many benefits, especially for those seeking a high-performance environment to meet the growing needs of extremely large analytic databases. These key factors are seamlessly engineered into a column-oriented database, which enable reasonably-priced, benchmark-busting performance to meet an organization's business intelligence needs.

- **Engineered for analytic performance**: Because of the limitations described previously, there are limits to the performance which row-oriented systems can deliver when tasked with a large number of simultaneous, diverse queries. The usual approach of adding increasing numbers of space-consuming indexes to accelerate queries becomes untenable with diverse query loads, because of storage and CPU time required to load and maintain those indexes. With column-oriented systems, indexes are of a fundamentally different design. They are engineered to store the data, rather than as a reference mechanism pointing to another storage area containing the row data. As a result, only the columns used in a query need be fetched from storage. This I/O is conducted in parallel, because large columns are automatically distributed across multiple storage RAID groups. Once retrieved, columns are maintained in cache using caching algorithms intended to optimize memory access behavior patterns, further reducing storage traffic.
- **Rapid joins and aggregation**: In addition, data access streaming along column-oriented data allows for incrementally computing the results of aggregate functions, which is critical for business intelligence applications. In addition, there is no requirement for different columns of data to be stored together; allocating columnar data across multiple processing units and storage allows for parallel accesses and aggregations as well, increasing the overall query performance. An underlying query analyzer can evaluate ways to break the query down in ways that not only exploits the availability of multiple CPUs for parallel operations, but also arranges the execution steps to leverage the ability to interleave multiple operations across the available computational and memory resources, creating additional opportunities

for parallelization. For example, complex aggregations and groupings can be "pipelined" by streaming the results of parallelized data scans into joins that in turn feed groupings, all happening simultaneously.

- **Smaller storage footprint**: One of the dependent factors among row-based systems to accommodate the aforementioned "data explosion" is the need for additional structures beyond the row-based data storage. These include the addition of indexes, tables for pre-aggregation and materialized views to the already burdensome storage requirements. A more efficient design is used in column-oriented systems, where data is stored within the indexes, eliminating the storage penalty of the types of indexes used in row-based systems. Some column-oriented systems not only store data by column, but store the data values comprising a column within the index itself; efficient bit-mapped indexes are selected to optimize storage, movement and query efficiency for each individual column's data type.
- **Suitability for compression**: The columnar storage of data not only eliminates storage of multiple indexes, views and aggregations, but also facilitates vast improvements in compression, which can result in an additional reduction in storage while maintaining high performance. One example, typically employed where a column contains often-repeating values, involves tokenizing the commonly used data values, mapping those values to tokens that are stored for each row in the column. Mapping the original form of the data to a "token", and storing the column as a list of these tokens requires much less storage space and yet, to the application, is totally transparent. For example, instead of maintaining copies of city name values in address records, each city name ("Phoenix, AZ") can be mapped to an integer value ("3449") which requires 2 bytes to store, rather than 10-12 bytes. The resulting compression is 4-6 times in this example. As another example, run length encoding is a technique that represents runs of data using counts, and this can be used to reduce the space needs for columns that maintain a limited set of values (such as flags or codes). For each column type and profile, specific indexing and compression techniques used as well, which contribute to further reducing the storage requirements.
- **Optimized for query efficiency**: The bitmap-based data structures used in some column-oriented analytic stores provide superior query performance by using sophisticated aggregation and bitmap operations within and across columns. If we use a bitmap to describe data, we can make use of the mapping to also count unique occurrences at loading time, and therefore we can provide that pre-aggregated count rather than analyzing the actual column data. The resulting systems may provide orders of magnitude improvements in query processing performance. As an example, consider the use of a tokenized column for listing cities. Some queries, such as those that count occurrences, never need to access the data itself, because some amount of metadata (such as "counts") that is an inherent result of preparing the column for storage can be captured at loading time.
- **Rapid data loading**: The typical process for loading data into a data warehouse involves extracting data into a staging area, performing transformations, joining data to create denormalized representations and loading the data into the warehouse as fact and dimension tables, and then creating the collection of required indexes and views. In a row-based arrangement, all of the data values in each row need to be stored together, and then indexes must be constructed by reviewing all the row data. In a columnar arrangement the system effectively allows one to segregate storage by column. This means that each column is built in one pass, and stored separately, allowing the database system to load columns in parallel using multiple threads. Further, related performance characteristics of join processing built atop a column store is often sufficiently fast that the load-time joining required to create fact tables is unnecessary, shortening the latency from receipt of new data to availability for query processing. Finally, since columns are stored separately, entire table columns can be added and dropped without downing the system, and without the need to re-tuning the system following the change.

The combination of these benefits establishes a highly scalable analytical database environment suited for extremely large data sets. Reducing the storage footprint and optimizing data access means that the I/O channel load is significantly decreased, which opens up critical bandwidth for supporting many simultaneous queries. Therefore, the column-oriented approach delivers the kind of performance necessary for emerging business intelligence activities such as real-time analysis or embedded predictive analytics.

Managing the memory hierarchy

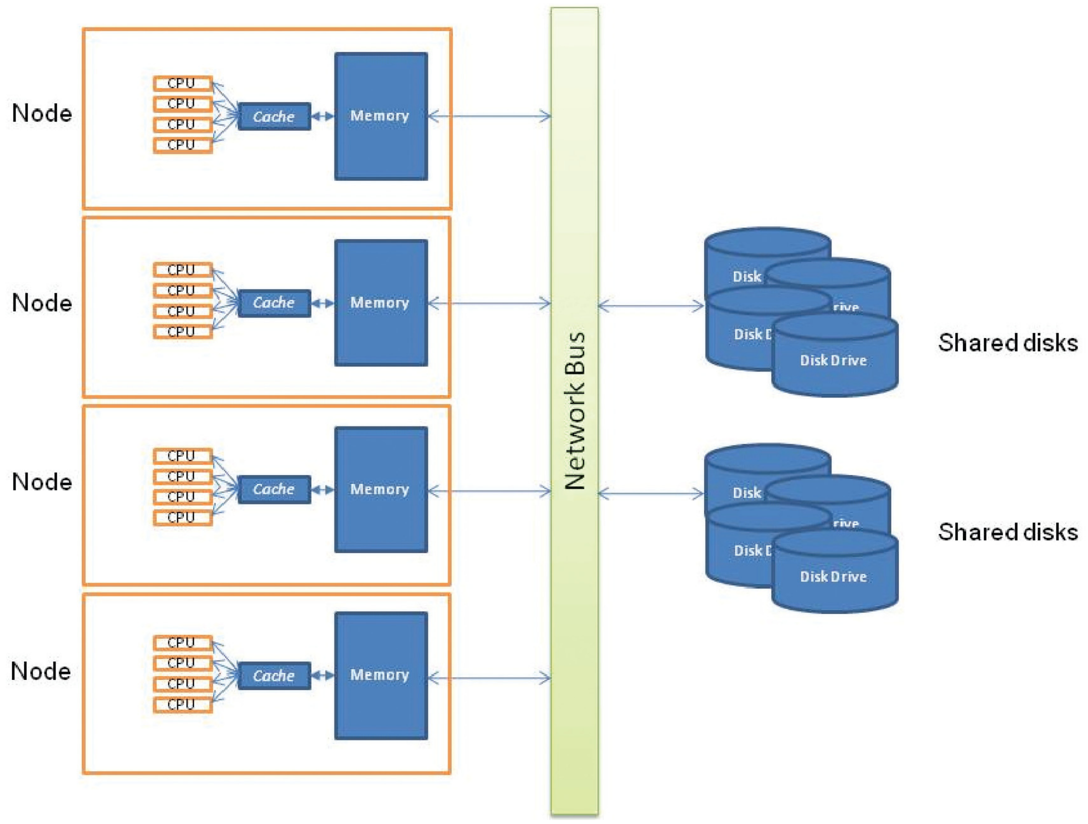
Understanding the hierarchy of computer memory management provides some insight into ways the columnar approach, when coupled with some additional system characteristics, leads to improved query performance. This overview of the computer memory hierarchy can be used to demonstrate the value of the columnar approach, especially when coupled with the right kinds of persistent storage architecture.

The memory hierarchy

In order to execute a database query, records must be streamed into the CPU to compare the query conditions, at which point the specific values are selected to generate a result set, aggregation, or other types of output to be returned to the user. However, CPU speeds are usually much faster than the underlying storage systems, especially in multiprocessor systems that share network-attached storage or SAN disk clusters. To address this, frequently accessed data values or records are stored in faster areas of memory, such as temporarily-allocated areas of memory or caches.

Realize, though, that most data values do not usually reside in the faster areas of memory—the lion's share at any time are stored in persistent memory systems such as SAN storage or even near-line storage systems, whose access times are even greater than that of main memory. These

Figure 3: The memory hierarchy of a multiple-processor configuration with shared disk



layers of the memory hierarchy add latency to every data access, and any approach to enable the streaming of data directly to the processor from the higher latency storage into faster areas of memory (as a way to track with CPU speed) will reduce query execution time.

As is shown in Figure 3, to be able to satisfy a database query, records accessed from shared disk storage must be streamed across the network into main memory (and subsequently, into the cache) so that they can be accessed by the processor in order to execute queries. It is valuable to recognize that as the access latency decreases, the memory capacity decreases, and this frames a large component of performance improvements for queries: only accessing what is needed, reducing the amount of data to be accessed, and making that access happen faster.

Performance drivers
More to the point, the characteristics of the memory hierarchy suggest a number of criteria relating to query performance addressed by some column-oriented database systems.

1) Reducing the physical amount of data to be streamed from the disks to the processor will reduce query time by reducing data access latency across the hierarchy and increases efficient use of the limited capacities.
2) Use of intelligent parallelization when placing information onto storage will reduce contention for disk resources, thereby reducing thrashing and increasing throughput.
3) Reducing the physical storage space required for the data will cut the cost of both the storage itself as well as the storage required for some types of backup.
4) Reducing the network bandwidth needed.
5) Reducing the need for replication.
6) Increasing scalability of data loads for queries.

The architecture of the column-oriented database system will dictate the degree to which these optimizations can be used to provide faster query responses and linear scalability in proportion to the number of concurrent queries. Because many analytic queries focus on specific columns, only those columns requested need to be loaded across the memory hierarchy, and because of the sequential layout of each column, each load is more efficient, streaming blocks of data directly through main memory and into the cache. The data compression techniques, coupled with any special indexing schemes and the use of parallelism, address the reduction in storage requirements, and as a byproduct also reduces required network bandwidth. Not only that, intelligent management of the memory hierarchy enables additional performance opportunities, such as optimizing system throughput as more simultaneous users participate executing different kinds of queries. The amount of memory and CPU resources allocated to individual queries and data loads can be scaled to ensure a smooth total system throughput, instead of allowing any one specific query acquiring and holding on to its maximum required number of resources.

Parallelism and exploiting the shared-disk framework
Parallel architectures span many different configurations. Massively Parallel Processing (MPP) systems typically have many processors, each of which is connected to its own persistent storage layout. There are some Symmetric Multi-Processing (SMP) configurations whose network storage architectures do not allow for concurrent access to storage. Both of these "shared-nothing" approaches can be contrasted with other types of SMP certain configurations ("shared-disk") in which the clusters share storage and network bandwidth, thereby sharing resources such as main memory and disk systems. For analytic data warehousing, there are benefits to both configurations. However, when considering our drivers for optimization, it becomes clear that the shared-disk approach provides some benefits not provided by the shared-nothing approach, such as:

- Lowered I/O bandwidth demands on the network interconnect
- Reduced I/O bandwidth requirement through reduction duplication of storage

- Reduction of disk I/O latency through flexibility that permits intelligent allocation of storage space (which eliminates contention for specific RAID groupings thereby significantly reduces or even eliminates thrashing on the disk arrays in contention)
- Through the reduction of I/O, increasing the number of users and simultaneous queries that can be supported before SAN controller architectural limits are reached
- Reduction in storage requirements through compression enables faster, more reliable, less costly backup procedures
- Reduction in storage requirements preserves the feasibility of snapshot duplication as a backup technology into far larger data warehouses
- Faster and more scalable storage and disk I/O reduces the risk of exhaustion of time available for loading the data into the analytic store
- Scalable performance that can be achieved without depending on manual partitioning schemes
- No restriction on adding capacity—new processing units and disk drives can be added independently without a need for pairing
- Linear scalability for concurrent query data loads can be achieved—the partitioning schemes necessary for MPP systems will drag down concurrent query performance as contention for CPU cycles, disk drives and network bandwidth increase
- Bulk loads are more efficient by avoiding unnecessary duplication

The configuration shown in Figure 3 captures these aspects—a multiprocessor environment with shared disk storage enables optimized queries with linear scalability in performance

Factors for evaluation

Understanding the benefits of the columnar approach and the issues that drive architectural decisions helps in guiding the evaluation process when considering a columnar system. If the primary drivers are high performance for analytic data environments, then reviewing the desired characteristics of candidate systems will help establish the differentiating factors that can be applied within any organization's to best meet business objectives. In other words, keeping these criteria in the forefront when evaluating candidate database architectures will not only provide greater clarity when considering a decision, but may even obviate the need for costly proofs-of-concept prior to down-selection and decision-making during the procurement process.

- **Data loading and integration**
  In many cases, the columnar database is used almost exclusively for analysis, which means that data from operational and transactional systems will need to initially be imported into the environment, followed by either periodic full loads or incremental loads. We have seen four aspects to be considered.
- **Incremental loading:** Once data has been integrated into the columnar system, it is likely that periodic loads will be performed to incrementally update the analytic database system. Some issues to consider include whether updates require full loads or if incremental loads are supported, the speed of incremental updates, and if incremental updates need to be performed with the system offline or if they can be done while the system is live, and if so, the degree to which performance is degraded during the incremental load.
- **Compression**: One of the key factors in reducing the storage footprint is the ability to compress data along each column. Different compression techniques can be used, each of which providing alternate benefits that can be compared, such as speed of compression, the speed of decompression, and the reduction in required space for storage.

- **Indexing**: Augmenting the natural index of the columnar orientation with additional indexes can improve performance, but that also means there is a need to index data as it is loaded into the database, providing yet another aspect for evaluation.
- **Schema flexibility**: Most analytic and BI applications, when run on traditional databases require the use of denormalized schema, plus pre-aggregations, materialized views and additional indexes, in order to achieve reasonable query performance. These techniques, while mandatory on row-based stores are optional in column-oriented systems. This is because the inherent performance advantages of column-oriented platforms over traditional ones can often provide sufficient "headroom" to permit simplification of ETL processes through the eliminating certain complex transformations to achieve faster loading, while still achieving net improvement in query performance.

As part of an evaluation, it is worthwhile to test these different aspects by selecting data sets to be loaded into the database, along with additional updates.

- **Performance**: This is probably the primary reason for considering a columnar database, as well as the primary differentiating factor. Reviewing reported benchmark scores may not provide a complete comparative assessment, since every organization's analysis needs are driven by a combination of canned reports, frequently-performed queries, and the ad hoc queries performed by power users.

  In order to best compare and assess the performance of different columnar database systems, it is worthwhile to configure a number of queries that are typical of the organization—standard reports as well as power-user queries. Benchmarking the actual performance of the organization's commonly-performed queries will supersede generic results from industry benchmarks.

  - **Complex query performance**: Using complex queries with nested subqueries often confound typical database systems. Review the approaches for query optimization, and how those approaches are improved as a result of the columnar layout.
  - **Mixed-load query performance**: Today's analytic platforms are targeted at a much broader collection of users, including those reviewing canned reports, those performing analysis through iterative ad hoc queries, as well as power users invoking data mining or exploration algorithms. This implies not only a heavy user load, but also one that is quite mixed. Therefore, assess the degree to which the platform can support a mixed load of queries.
- **Scalability**: There are three aspects to evaluating database scalability.
  - **Size of the data**: Even though columnar databases are intended to handle large amounts of data, there will always be considerations as the size of the data increases, especially in relation to data access times, compression/decompression, and to a great extent, requirements for managing data or indexes in memory.
  - **Simultaneous queries**: Although column-oriented databases are well-suited for analysis and queries, even these types of systems may be subject to slowdowns as contention for data resources increases.
  - **Number of users**: The number of simultaneous queries is likely to increase in concert with an increase in the number of users, and therefore one must consider the system's ability to handle multiple users as well as number of simultaneous queries as part of its administrative capabilities to provide scalable support.
- **Access**: Proprietary access languages and tools have been rapidly eclipsed by the use of database access standards such as SQL. Yet there are some systems that do not fully support the latest SQL standards, so when evaluating column-oriented databases, determine which features of SQL are used within your organization, and verify that the database supports those features.

- **Backup/recovery**: One of the biggest pain points in very large data warehousing systems is the aspect of backup/recovery and high availability. By their sheer virtue of data compression, columnar databases require less time in backup and recovery. When adding the concept of partitioned database units into the picture, recognize that the tables can be backed up and subsequently recovered independently, thereby simplifying this relatively complex problem.

There are other factors to consider as well. Column-oriented databases are relatively simple, yet as new features and capabilities are added, there is a greater need for managed administration tools. The simplicity of the design may also lead to limitations in terms of the types of data that can be incorporated into the database—seek those systems that do not restrict the use of unstructured data or XML. Structural constraints may force the user to employ the same keys across the entire system, while others provide more flexibility, both in key use and in tabular vs. the more traditional star schemas used for data warehouses.

Importantly, it is also valuable to consider product maturity and the landscape of product installations. If the evaluation criteria are met, check to see if the product has real-world production references to back up the considerations, especially if there may be a need for experienced technologists to help in supporting the implementation and migration processes to move the analytical platform into production.

Summary

As interest in analytical application grows, so do the performance and scalability requirements for the enterprise data warehouse. And as large-scale systems continue expand at an alarming rate, alternate approaches to support standard reporting, analytical analysis, and power-user ad hoc queries will become increasingly established as the platforms of choice for very large database systems. As more "power users" adjust their approaches to analysis to iteratively incorporate more ad hoc queries, the underlying system must be able to accommodate the growing performance expectations. Columnar databases are designed to support this level of performance, providing access to any user at any time, executing any variety of types of queries.

The column-oriented approach provides a combination of architectural simplicity and the ability to configure data in a way that can reduce the physical amount of data that must be accessed. Reducing the storage footprint while optimizing column access will reduce data access latency, reduce physical storage requirements, and optimize use of network bandwidth, thereby contributing to a scalable environment that continues to provide optimized performance as data volumes, number of users, and number of queries increase.

For more information

Go to http://datagov.sybase.com

About Sybase

Sybase delivers mission-critical enterprise software to manage, analyze, and mobilize information. Sybase IQ is the #1 column-based analytics server, recognized for unmatched performance for the very largest data sets. When it comes to data challenges, Sybase IQ is a fast, reliable solution for more efficient and responsive government.

Sybase Federal

For over 20 years, Sybase has supported the diverse missions of federal government departments and agencies, helping them maximize the value of their data and increase the effectiveness of their services. Government organizations have used Sybase solutions to mobilize supply-management systems, reduce ship inspection times by half, and cut data capture time from one year to two days.