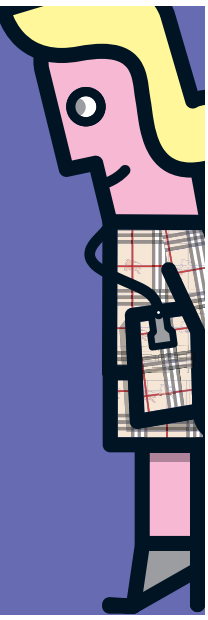


New breed has been fundamentally re-designed

Inside the New-Generation Analytic DBMS

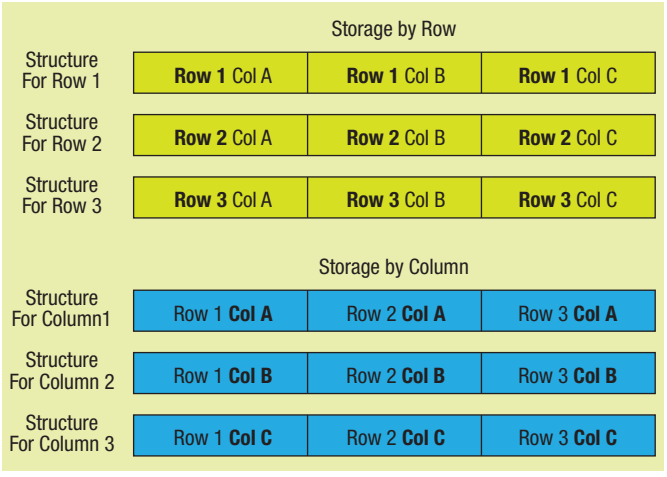
Kim Stanick

It is not surprising that the Business Analytics market continues to grow rapidly and invite new innovations to feed companies' aggressive appetites for more data more quickly. IDC's latest research revealed that this market segment outpaced previous estimates to grow more than 14% from 2006 to 2007.

There are many reasons for the phenomenon, but it is fair to summarize that the business game is changing. Businesses increasingly rely on analytics to support their decision-making. As more companies compete by using business analytics, the more business analytics become a necessity for all companies to compete. The problem is that analytic processing has traditionally been resource intensive and achieving adequate performance has been cumbersome in the long term. One thing is constant: as IT organizations seek new ways to improve analytic performance new vendors emerge to provide them.

A New Breed

The new generation of analytic DBMS providers is combining new innovations with well-known performance principles to make claims of very high performance out of the box. These claims resonate as companies seek greater analytic efficiency. There are a number of common threads among these new vendors, but the one that is getting the most attention is that they use a column-based approach. (Note: there are a number of terms used to denote this concept: column-wise, column-oriented, columnvectored, column store, columnar). In addition to being columnar, these vendors also emphasize deep compression, shared-nothing massively parallel processing, and, to some extent, a penchant for processing in-memory whenever possible. This article will describe these basic architectural principles, and why they matter, in a vendor-neutral way. Individual vendors will naturally boast innovations and specializations, but those nuances will be left to the reader to explore on a vendor-by-vendor basis.



Afbeelding 1: Data Stored by Row vs. Column.

What is Columnar?

I/O tends to be the largest resource bottleneck in processing analytic workloads on today's platforms. Therefore, I/O optimizations can pay off in big ways. In a row-wise DBMS, I/O is saved by using auxiliary structures like secondary indices, join indices and summary tables. The problem is that these structures need to be designed and managed, they take up space, and they impact load performance. Plus, it's not feasible to index everything, so valuable ad hoc discovery-type queries will likely be discouraged in such an environment.

By contrast, a columnar DBMS saves I/O by optimizing the data organization within the storage layer to be more favorable for analytic queries. There is a one dimensional nature to how disk heads write to sectors. Whereas a row-wise DBMS writes its records *by row in field order*, a columnar DBMS writes its records *by field in row order*. This technique is a foundational element in enabling good performance because the nature of business analytics is to discern facts not from a single record, but across sets of data – sometimes very large sets. Specifically, the data for each column is written in column data structures that preserve the same row order as every other column so that

a whole record can be easily assembled from the column data structures: the first record's data is in position one, the second record's data is in position two, etc. (See Figure 1).

Storing data in a columnar way reduces I/O processing for analytic queries because they typically only need to analyze a subset of the fields available. To understand the potential significance of the I/O savings, consider a simple query to identify the average Age by Country in the E.U. For simplicity, let's assume there are approximately 400 million people and for each person we have collected demographic data consisting of 100 fields averaging 10 bytes each. This translates to 1,000 bytes per person and a total data volume of 400GB. To answer the query, a columnar DBMS scans only data blocks associated with "Age" and "Country" or 2% of the data (2 fields out of 100) which is 8GB. In contrast, a row-wise DBMS without an index would need to scan all of the data blocks to grab the Age and Country values from each row. (See Figure 2). That's a difference of 50X.

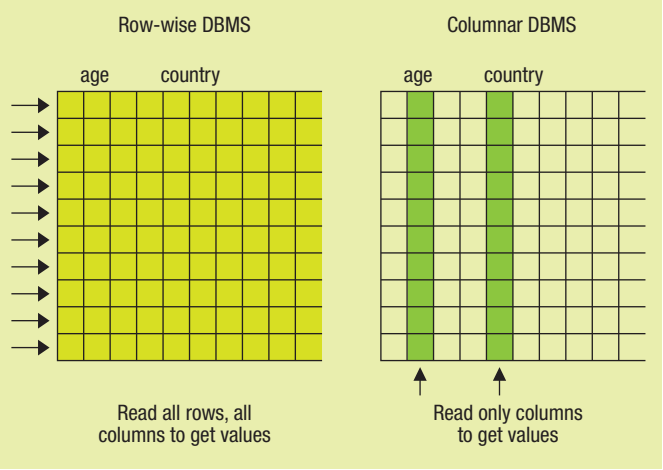
To be accurate, columnar orientation is not new – research about performance improvements using vertically partitioned database tables can be found as early as the 1970s (and perhaps earlier). However, combining columnar techniques with shared-nothing massively parallel processing is a relatively recent market development.[1]

Columnar and Compression

Data compression has a significant affect on resource efficiency across the board. After compressing there are fewer bytes to scan, fewer bytes to move, and fewer bytes to process. Compression is not unique to a columnar DBMS but, due to its nature, a columnar DBMS has a greater opportunity to take advantage of data compression because columns are domain-specific. That is, data that belongs to the same field shares the same domain type (int, char, varchar, etc.) which can optimize the selection of compression method. With this affinity of domain type, a columnar DBMS can gain greater efficiency in compressing and decompressing by applying it at the block level rather than at the individual field level. Common compression techniques like LZ, RLE, and Delta are simpler to implement and typically provide greater benefit when applied at block level versus row level. Plus, applying compression to sorted data can further amplify compression benefits. The columnar domain affinity is an enabler here, too. Furthermore, a greatly compressed data set may be more likely to fit in memory for additional performance gains (more on this in the next section).

Strengths and Weaknesses

We've already observed that a columnar DBMS saves greatly on I/O, the largest bottleneck in analytic processing, but these I/O savings have deeper implications. Greatly reducing I/O also reduces the size of intermediate result sets and the amount of data to be moved. This has positive and lasting affects down the



Afbeelding 2: Age by Country Example.

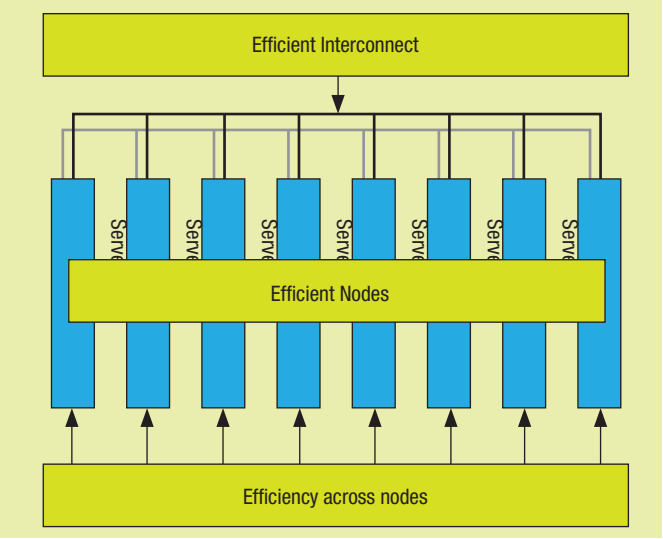
line by reducing the CPU cycles necessary to oversee I/O and Interconnect processing. The CPU is now more available for CPU-intensive work such as calculations, sorts, aggregations, etc. In short, by minimizing earlier the set of data to be processed there is less processing to do. In other words, the CPU doesn't need to prune what has already been pruned.

The column data structures are comparable to single column indexes in row-wise data bases but without the data duplication and associated overhead. Therefore, since each column acts as an index, fewer indexes and other auxiliary tuning structures (indices, data redundancy, summaries, join indices) are typically required for a given performance level. According to Forrester Research, in analytic environments data can inflate as much as 16-fold over it raw size due to various forms of data redundancy.[2] Being conservative with Forrester's estimate, consider that a 2TB data warehouse may actually only contain 500GB of raw user data (4:1). With a conservative 4:1 compression ratio, that 2TB becomes 250GB and can easily fit in memory on a moderately-sized MPP cluster (more on MPP below). Applying this inflated data concept, if a DBMS can significantly reduce the size of data by avoiding data inflation, it can potentially enable in-memory processing for even greater speed.

Finally, it is possible that by increasing performance without increasing the effort to achieve that level of performance a columnar DBMS may potentially enable ad hoc query by reducing the fear of performance degradation from unanticipated queries.

To summarize why columnar is inherently good for analytic processing, it:

- minimizes I/O;
- conserves space due to compression affinity;
- minimizes the size of intermediate results without the expense of pruning;
- reduces dependency on performance-related indices and data redundancy;
- conserves memory, CPU and interconnect bandwidth.



Afbeelding 3: Keys to a Good MPP Implementation.

It is important to note that the very thing that makes a columnar DBMS more efficient for analytic processing makes it less efficient for transaction processing or applications that require highly concurrent row-at-a time operations (e.g. you wouldn't choose a columnar DBMS for an airline reservation system).

Misconceptions

One common misconception is that a columnar DBMS is an "alternate DBMS" or is not relational. However, the columnar orientation occurs at the storage layer – the logical database design is *relational* and the underlying mapping between logical and physical representation is managed by the DBMS. We'll discuss DBA and design implications later, but realize that a columnar DBMS is a relational DBMS that contains tuples and speaks SQL.

Another misconception is that a columnar DBMS is slower than a row-wise DBMS for an all-record, full table scan that touches all columns. The rate of I/O at the HDA (head disk assembly) is a function of storage technology, not the DBMS, and would be exactly the same for each type of DBMS – the same number of bytes will take the same amount of time to read off of disk.

For example, in a full table select (Select * from table X), the performance of a columnar DBMS would be *no less* than that of a row-wise DBMS.

Yet another misconception is that a columnar DBMS has poor load speed. Because a columnar DBMS must parse the data into columns, it is thought that loading data must take longer. If you're loading a full table, a columnar DBMS requires the same number of I/Os as row-wise DBMS – it's the same amount of data. Column stores are less efficient with row-at-a-time inserts, but batch loading (even very frequent batches) has no negative performance characteristics and should be utilized whenever possible. Also, since a columnar DBMS tends to compress well, if I/O is the bottleneck then a given batch of data will load faster into a column store than a row store.

A further area of misconception is that columnar databases are only useful for dimensional (star schema) applications. This is a vendor-specific issue relating to the fact that some implementations rely heavily on join indexing and not to the nature of their being columnar. Vendor-specific reliance on Star Schema may also negatively impact load performance which may further explain the misconception of poor load performance.

A final misconception to note is the belief that a vertically partitioned row-wise DBMS is equivalent to a columnar DBMS. But to vertically partition, a row-wise DBMS must add row overhead for each partition. This overhead becomes significant when a query spans multiple vertical partitions and must involve a join. To overcome the join penalty, vertical partitions can be grouped into sets but this adds further overhead and places a burden on the designer to decide in advance which groupings to define. This overhead also impacts load performance.

Shared-Nothing, Massively Parallel Processing (MPP)

MPP has been around since the late 1970s and is a proven technique for improving analytic performance. What is new is that it's now being applied to the columnar DBMS. MPP is a divide-and-conquer approach that greatly enhances performance and scalability for large data volumes. The concept of parallelism is very simple, by splitting up data across many nodes (servers) that are each responsible for their own data and processing, but who work together, you can process more data in a given time window. Each node in an MPP system is a self-contained DBMS instance.

MPP is a key enabler for scaling out to potentially thousands of standard servers to increase analytic capacity and performance. The keys to a good MPP implementation are efficient individual compute resources (nodes), even distribution of the workload (because a parallel system is only as fast as its *slowest* unit of parallelism) and an efficient interconnect. (See Figure 3.)

Based on concepts discussed thus far, a columnar, compressed DBMS (from here on I'll just say columnar to be brief since they most often occur together) can add further value in an MPP implementation. As stated above, a columnar DBMS tremendously reduces the data size to make I/O and CPU processing more efficient within the node. However, these benefits also apply during data movement between nodes. Greater efficiency within the node combined with greater efficiency across the nodes is a powerful combination to deliver consistently good analytic performance.[3]

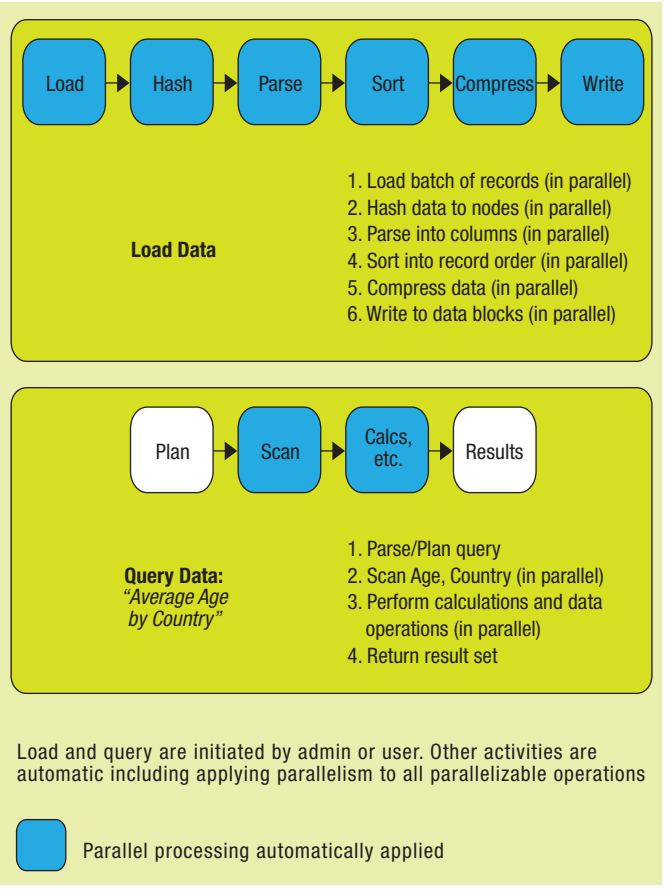
MPP also provides performance gains during data loading by enabling parallel load. Parallel loading of bulk data (including mini batches) is an important part of overall analytic processing efficiency especially as data volumes grow, load windows shrink, and applications evolve toward near-real-time. It is important to be able to scale the load with the overall size of the data and platform.

What DBAs and Designers Need to Know

As someone who deals with the implementation environment, you'll be pleased to know that working with a modern columnar, MPP DBMS is just like other working with any other relational DBMS. As mentioned earlier, the columnar storage orientation is managed by the DBMS – as data are loaded into the database, records are automatically converted to a field orientation and affiliated with the appropriate items in the data dictionary. You don't do anything special to facilitate this.

As a standard relational DBMS, a columnar, MPP DBMS understands SQL syntax. You will use SQL Data Manipulation Language to query data and SQL Data Definition Language syntax to define tables, views, macros – and indexes – if you choose to use them.

It's appropriate to point out that while you shouldn't automatically assume you need the types of indices you've become accustomed to when using other types of DBMS in analytic environments, they are available to you (though you will incur a load performance as with all indices). This same comment applies to physical schema design. With a columnar, MPP DBMS you should experience higher levels of performance without cluttered schemas – e.g., schemas with a lot of pre-defined structures to manage that enhance performance but do not enhance the user experience. Columnar orientation, by design, should reduce the need for traditional access improvement structures, such as indices and materialized views, that are intended to avoid the data access bottlenecks that arise when using row-wise storage for analytic processing. With columnar there should be fewer tuning structures to create and manage. Secondary indices that were formerly placed on fields in a row-wise DBMS are no longer necessary because the columnar database naturally retrieves data by field – without indices.



Afbeelding 4: Data Load and Query Processes.

Figure 4 provides a high level walk through of the data load and query processes in a columnar, MPP DBMS. As you can see, the columnar orientation and parallelism are built in.

Why Now?

It took many years, but conventional wisdom finally dictates that it is proper and efficient to use an *analytic* DBMS for analytic processing. And, as expected, the resulting increase in demand for such tools has fueled a new generation of innovation to bring forth a new breed of analytic DBMS designed for today's analytic computing environments. Hence, there are a number of new DBMSs that combine the proven concepts of columnar and MPP with a healthy dose of new ideas.

From the prior discussion, it's evident that this new breed of analytic DBMS has been fundamentally re-designed from the ground up to take best advantage of system resources in order to provide greater analytic efficiency than ever before. Byte-for-byte of raw data, a columnar, compressed, massively parallel DBMS should yield better analytic performance on a given number of CPUs than it's row-wise cousin. Per Philip Howard of Bloor Research, "Columns provide better performance at a lower cost with a smaller footprint: it is difficult to understand why any company seriously interested in query performance would not consider a column-based solution."[4]

For those of you whose companies compete by using business analytics, the game just changed in your favor.

References

1. *Interestingly, although the columnar concept has been around for decades, it is still new to some people: a 2008 IDC study of BIDW professionals showed that 25% of survey respondents weren't aware of columnar DBMSs.*
2. *Data, Data Everywhere. Boris Evelson, Forrester Research 2007.*
3. *By contrast, Symmetric Multi Processing (SMP) is the most common alternative to MPP. In an SMP environment, a system is scaled by adding CPUs (and memory and disk) to a shared memory environment. The system manages the utilization of resources across all CPUs. At some point there is overhead to managing the CPUs and contention for shared resources to the point where adding another CPU may not deliver a whole CPU's power. SMP systems are scalable only up to a relatively finite number of CPUs (e.g., 128).*
4. *What's Cool About Columns. Philip Howard, Bloor 2008.*

Kim Stanick is VP Marketing, ParAccel, Inc.

The author would like to thank her colleague Rick Glick for his contributions to this article.