

# Best Practices in the Use of **COLUMNAR DATABASES**

How to select the workloads for columnar databases  
based on the benefits provided

William Mcknight  
[www.mcknightcg.com](http://www.mcknightcg.com)

SPONSORED BY:

**calpont**<sup>®</sup>

ACCELERATING DATA INSIGHTS

[www.calpont.com](http://www.calpont.com)

**InfiniDB**<sup>®</sup>

## Table of Contents

Columnar Database Fundamentals .....	1
Best Practices with Columnar Databases .....	2
Use to Save Money on Storage.....	2
Use to Speed Up Important Input/Output Bound Queries (Many Are!).....	4
Use to Speed Up Scan Operations.....	4
Use to Scale Beyond Cubes .....	5
Use to Execute Real-Time Analytics of Big Data (Not Hadoop's Sweet Spot).....	6
Use to Efficiently Execute Column Selective Queries.....	6
Case Study .....	7
Conclusion.....	8

## Columnar Database Fundamentals

Columnar databases are surging with the NoSQL movement, but actually they play in both SQL and NoSQL camps. From a SQL perspective, some columnar databases have been around for two decades (e.g., Sybase IQ had product momentum in the 1980's) and the SQL language works just fine against them. However, if you define NoSQL as having a different data layer from the relational "all columns successively" row-based layout, columnar databases are NoSQL. Furthermore, if you define NoSQL as being not ACID (atomicity, consistency, isolation, and durability) compliant, some columnar databases are and some are not.<sup>1</sup>

Whether columnar databases are part of the NoSQL movement or not by your definition, they are becoming an essential component of an enterprise infrastructure for the storage of data designed to run specific workloads. When an organization embraces the value of performance, it must do everything it can to remove barriers to the delivery of the right information at the right time to the right people and systems. There is no "ERP" for post-operational data. No one-size-fits-all system. Some gave that role to the relational, row-based data warehouse, but that ship has sailed. In addition to columnar databases, very-large data stores like Hadoop™, real-time stream processing, and data virtualization are required today to bring together result sets across all data systems.

This paper focuses on conveying an understanding of columnar databases and the proper utilization of columnar databases within the enterprise. Where it makes a difference, emphasis is given to the implementation of columnar by Calpont's column analytic database, Calpont InfiniDB® (hereafter referred to as InfiniDB).

Columnar storage reduces the primary bottleneck in analytic queries: I/O. I/O is such a bottleneck that the number of I/Os a query will require is absolutely a fair relative measure of the time it will take to execute. It wasn't always this way. CPUs were the initial bottleneck and SMP, clusters and MPP came to the rescue, providing the ability to get thousands of CPUs as busy as possible. However, they continue to sit idle more often than desired. This is due to the inability of the pre-CPU layers of memory, L2 (especially) and L1 caches to throughput data rapidly. One of the primary reasons for this is that complete records are sent through the layers by row-based systems, which are designed to process rows instead of columns.

Think about it from an efficiency standpoint. When I want just a few songs from an album, it's cheaper to purchase only those songs from iTunes that I want. When I want most of the songs, I will save a couple bucks by purchasing the whole album. Over time, I may find that I like one of those songs. However, when it comes to a query, the query either wants a column or it doesn't. It will not come to like a column later that it was forced to select. This is foundational to the value proposition for columnar databases like InfiniDB.

<sup>1</sup>Calpont InfiniDB is ACID compliant

## Best Practices in the Use of Columnar Databases

The major significant difference between columnar and row-based stores is that all the columns of a table are not stored successively in storage – in the data pages. This eliminates much of the metadata that is stored on a data page, which helps the data management component of the DBMS navigate the many columns in a row-based database as quickly as it can. In a relational, row-based page, there is a map of offsets near the end of the page to where the records start on the page. This map is updated as records come and go on the page. The offset number is also an important part of how an index entry would find the rest of the record in the data page in a row-based system. The need for indexes is greatly minimized in column-based systems, to the point of not being offered in many, as is true with InfiniDB.

## Best Practices with Columnar Databases

### Use to Save Money on Storage

InfiniDB knows where all column values begin on the page with a single offset calculation from the beginning of the file. There is no value-level metadata. All column data stores keep the data in the same row order so that when the records are pieced together, the correct concatenation of columns is done to make up the row. This way “William” (from the first name column file<sup>2</sup>) is matched with “McKnight” (from the last name column file) correctly – instead of matching William with Smith, for example. Columnar systems used to store the record number alongside the value, but few do any more. InfiniDB matches values to rows according to the position of the value (i.e., 3rd value in each column belongs to the 3rd row, etc.).

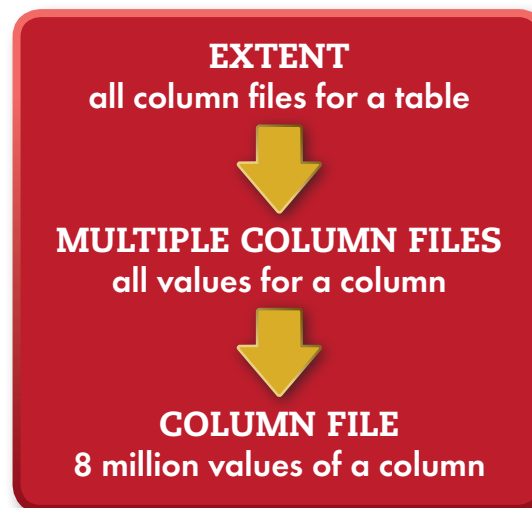


Figure 1. Storage relationships in InfiniDB

<sup>2</sup>See Figure 1 for an understanding of “column file”

InfiniDB stores fields of 8 bytes or less (which is most fields) with a field length of the smallest possible in the sequence 1, 2, 4, 8<sup>3</sup>. For example, a 2-byte character is stored as 2 bytes and a 3-byte character is stored as 4 bytes. For fields above 8 bytes (characters and variable-length characters), a separate dictionary structure is used to store the actual values along with tokens. These 8-byte tokens are used in place of the value in the data page and allow the data page to continue to operate without value-level metadata. The higher the repeat level of the values<sup>4</sup>, the more space will be saved.

For example, 1=State Grid Corporation of China, 2=Nippon Telegraph and Telephone and 3=Federal Home Loan Mortgage Corporation could be in the dictionary and when those are the column values, the 1, 2 and 3 are used in lieu of the actual values. If there are 1,000,000 customers with only 50 possible values, the entire column could be stored with 8 megabytes (8 bytes per value). The separate dictionary structure, containing each unique value and its associated token, would have more page-level metadata. Since each value can have a different length, a map to where the values start on the page would be stored, managed and utilized in page navigation.

**The dictionary arrangement allows InfiniDB to trim insignificant trailing nulls from character fields, furthering the space savings. Effectively, characters over 8 bytes are treated as variable length characters.**

**DICTIONARY:**  
1, State Grid Corporation of China,  
2, Nippon Telegraph and Telephone,  
3, Federal Home Loan Mortgage Corporation

**DATA PAGE:**  
1,3,2,3,1,3,1, ...

**Figure 2.** Dictionary and data page snippet showing company column data on a data page

In addition to dictionary compression including the “trimming” of character fields, traditional compression (with algorithm LZ0) is made available to InfiniDB data.

<sup>3</sup>Including variable-length character fields, stored as fixed-length  
<sup>4</sup>Or, put another way, the lower the cardinality



## Use to Speed Up Important Input/Output Bound Queries (Many Are!)

To optimize the I/O operation, it is essential to read as much as possible in each I/O. Many units of I/O have been dramatically expanded in columnar systems from the typical 8K/16K/32K/64K you find in row-based relational systems. InfiniDB data pages are 8K. When selection and projection

**A columnar database not only provides a greater amount of data in I/Os, but also a greater amount "relevant" data in I/Os. It knows that all the data values that it reads must be processed, and that those reads are less cluttered by page metadata for the DBMS' use.**

operations of a query are done in parallel (explained later), it is advantageous for InfiniDB to be able to I/O at a page, or a few pages, level. However, once it is possible, the I/O becomes multiple pages. Most I/Os are 512 pages, which is 4 MB of 99% relevant data.

An I/O in a columnar database will only retrieve one column – a column interesting to the particular query from either a selection or projection (WHERE

clause) capacity. The projection function starts first and gathers a list of record numbers to be returned, which is used with the selection queries (if different from projection) to materialize the result set.

In row-based databases, complete file scans mean I/O of data that is non-essential to the query. This non-essential data could comprise a very large percentage of the I/O. Much more of the data in the I/O is essential to a columnar query. Columnar databases can therefore turn to full column scans much quicker than a row-based system would turn to a full table scan. Query time spent in the optimizer is reduced as well since this decision is not nearly as critical or fatal if less than ideal, to the overall query time.

Columnar databases are one of many new approaches taking workloads off the star schema data warehouse, which is where many of the I/O bound queries are today. Heterogeneity in post-operational systems is going to be the norm for some time, and columnar databases are a major reason because they can outperform many of the queries executed in the data warehouse.

## Use to Speed Up Scan Operations

Some columnar databases use row-based optimizers, which negates many of the benefits of column orientation. They materialize "rows" (comprising of only the columns of the query – selection and projection) early in the query execution and process it like a row-based row from there. Column-based optimizers are able to divide the selection and projection functions into separate operations, and don't I/O entire column files. They limit I/O from the selection column files pages interesting to the selection criteria. InfiniDB uses a column-based optimizer to further reduce I/O in this manner.

InfiniDB has MapReduce-like<sup>5</sup> functionality that divides the query into its selection and projection operations. These operations are further parallelized (up to 16 threads.) Projection must start first in order to determine which values need selection, but the idea is to ramp up selection as soon as possible (as well as 512 page reads in the I/O as soon as possible.) Therefore, InfiniDB utilizes smaller I/Os at first, with the selection function coming close on the heels of projection beginning. This is a clever maximization of the I/O from all perspectives – divide and conquer task execution, parallel reads and size of read.

**Select firstname, lastname from customer  
Where city = 'San Jose' and state = 'CA'**

**Selection column files = firstname, lastname  
Projection column files = city, state**

**Figure 3.** Selection and Projection column files

Scan queries are not mutually exclusive with I/O bound queries. However, we could say the same thing about scan operations regarding their applicability in siphoning workload off the star schema data warehouse. Whereas scans were once the domain of the data warehouse, many are moving these workloads, when few columns are involved, to columnar databases for improved performance.

## Use to Scale Beyond Cubes

Multidimensional databases (MDBs), or cubes, are separate physical structures that support very fast access to selective data. Frequently entering organizations in the Finance department, these structures create storage and processing overhead for the organization.

When a query asks for most columns of the MDB, the MDB will perform quite well relatively speaking. The physical storage of these MDBs is a denormalized dimensional model, which eliminates joins. However, MDBs get large and grow faster than expected as columns are added. They can also grow in numbers across the organization, becoming an unintentional impediment to information access.

It is difficult to develop the necessary discipline to use MDBs with its best-fit workloads. MDB abuse is a major cause of the complete overhaul of the information management environment. Many are looking for scalable alternatives and the analytic workloads used with MDBs tend to have a lot in common with the more manageable columnar databases.

<sup>5</sup>MapReduce is a method of parallel reduction of tasks; a 25 year old idea that came out of the Lisp programming language. There are popular implementations of the framework introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers.

## Use to Execute Real-Time Analytics of Big Data (Not Hadoop's Sweet Spot)

Hadoop is a parallel programming framework for large-scale data. As much as it will be confused with other information stores and replacements attempted, Hadoop is not best suited as a direct competitor to databases, including columnar stores. The tasks you can do with Hadoop are a small subset of the tasks you might do with a database like InfiniDB. The Hadoop data is, however, a different profile of data than would be stored in a database.

The ideal workload for Hadoop is data that is massive not only from the standpoint of collecting history over time, but also from the standpoint of high volume in a single day. From a processing perspective, you would put data into Hadoop for which the functionality required is limited to batch processing with a limited set of query capabilities. Since most Hadoop systems are flat file based with no relational database for performance, nearly all queries run a file scan for every task, even if the answer is found in the first block of disk data. Hadoop systems are best suited for unstructured data, for that is the data that amasses large very quickly, needing only batch processing and a basic set of query capabilities.

Therefore, Hadoop is about batch processing of a large amount of specific data types. It is not for data warehousing nor the analytic, warehousing-like workloads that columnar databases are designed for. Often, summary data will be sent from Hadoop to the columnar database. Data analysts and business consumers access the columnar database 7 X 24 for ad-hoc reporting and analysis, whereas Hadoop access is scheduled and more restricted. Calpont has a bi-directional data sharing connectivity to Hadoop for the data that interfaces between InfiniDB and Hadoop.

## Use to Efficiently Execute Column Selective Queries

"Analytic queries" is a highly malleable term that causes more confusion than it solves. I prefer "column selective queries." Now that you have seen the physical differences of columnar databases, it should be clear that the workloads that will find their best platform in columnar are queries that access less than all columns of the tables it touches. In this case, less is more. The smaller the percentage of the row's bytes needed, the better the performance difference with columnar.

Columnar databases allow you to implement a data model free of the tuning and massaging that must occur to designs in row-based databases, such as making the tables unnaturally small to simulate columnar efficiencies.

## Case Study

ASI Group is a 38-year old private company that specializes in customized applications software for the aviation and postal industries. They were pioneers in the real-time data capture of bar code scan data. Their clients include Continental Airlines, Delta Airlines, SAS, LAX, and several diplomatic fleets. For most of these customers, ASI receives data from them in real-time, stores the information in its data center, and makes the applications available over the web. The applications mine all of a company's data for the many decisions each customer needs to make in the course of a day.

Considering the continuous in-flight nature of cargo, baggage and mail, decisions are crucial to airlines. Determining when to release a flight full of mail means taking into account atmospheric conditions, arrival time, the volume and pick-up location of that mail, contracts and service levels, reputation, tariffs, penalties and costs.

Gorden Rosen, CEO of ASI, spoke to me about their InfiniDB implementation, which supports many of these applications, and continually expressed the need for real-time performance. "Analysts are interrogating the database to understand the condition of several variables and add their judgment to make the right decision for the airline. It is imperative that they be able to get their questions answered very, very quickly so they can make the quick and accurate decisions that support their companies."

ASI started a large logistics project with another open source database and found that it would not scale. They then trialed three other solutions before landing on InfiniDB, which has been their analytic database of choice for over a year now. They load data into InfiniDB throughout the day. Rosen said that while data loads take a long time because of the size of the data, it is immaterial to their customers, who place the highest value on data retrieval times.

The larger customer databases exceed one billion rows and their queries tend to scan the entire database since the predicates are not very restrictive. The scan queries also tend to focus on just a few of the thousand-plus columns in the tables. For example, they continually need to see the status of all mail coming in and out of a city. This makes a good situation for a columnar database and, indeed, a row-based database could not handle the requirements. "We've tried row-oriented databases and the scans would run far too slowly, compromising the decisioning process of our clients."

ASI's philosophy is, "If the customer can think it, we'll find a way to build and support it." Rosen acknowledges the need for variety in an analytical architecture. They don't want to run into problems "brute forcing solutions" into ill-fitting architectures when alternatives are available. As for Calpont and InfiniDB, Rosen said "the Calpont team is fabulous. They've come on board like it's 'our' application. And the product simply gets the job done".

## Conclusion

Columnar databases provide a range of benefits to an environment needing to expand the envelope to improve performance of the overall analytic workload. It is usually not difficult to find important workloads that are column selective, and therefore benefit tremendously from a columnar orientation, such as the ones ASI Group uses with InfiniDB. Columnar database benefits are enhanced with larger amounts of data, large scans and I/O bound queries. While providing performance benefits, they also have unique abilities to compress their data. Like cubes, data warehouses and Hadoop, they are an important component of a modern, heterogeneous environment. By following these guidelines and moving the best workloads to columnar databases like InfiniDB, an organization is best enabled to pursue the full utilization of one of its most important assets - information.

## ABOUT THE AUTHOR

William McKnight is president of McKnight Consulting Group, a firm focused on delivering business value and solving business challenges utilizing proven, streamlined approaches in data warehousing, master data management and business intelligence, all with a focus on data quality and scalable architectures. Mr. McKnight brings process, organizational and architectural expertise to data strategies, data management implementation programs, and technology selection for clients, including several global corporate giants. McKnight, an award-winning consultant and author, can be reached at McKnight Consulting Group at [www.mcknightcg.com](http://www.mcknightcg.com).

## ABOUT THE SPONSOR

Calpont Corporation is a leading provider of scalable, high-performance analytic database software that enables fast, deep analyses of massive data sets. Calpont InfiniDB® Enterprise is the emerging choice for demanding data warehouse, Business Intelligence, and analytics environments. Known for its rapid implementation time, operational simplicity and extraordinary value, InfiniDB is a proven solution for data-intensive businesses that must minimize decision latency to be market competitive, including those in software, online media and commerce, social media, telecommunications and mobile applications.



ACCELERATING DATA INSIGHTS

**CALPONT CORPORATION**

2801 Network Blvd, Suite 220

Frisco, Texas 75034

[www.calpont.com](http://www.calpont.com)

[info@calpont.com](mailto:info@calpont.com)

Copyright 2011 Calpont Corporation. All rights reserved. "Calpont InfiniDB," the Calpont and Calpont InfiniDB logo and Calpont's product names are trademarks of Calpont. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only. No portion hereof may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the recipient's personal use, without the express written permission of Calpont. The information contained herein is subject to change without notice. Calpont shall not be liable for errors contained herein or consequential damages in connection with furnishing, performance, or use hereof.