

# [Column oriented database essay](https://assignbuster.com/column-oriented-database-essay/)

This column-oriented DB’S has advantages for data warehouses, customer relationship management (CRM) systems, and library card catalogs, and other ad hoc inquiry systems where aggregates are computed over large numbers of similar data items. It is possible to achieve some of the benefits of column-oriented and row-oriented organization with any Dobbs.

By netting one as column-oriented, we are referring to both the ease of expression of a column-oriented structure and the focus on optimizations for column-oriented workloads. This approach is in contrast to row-oriented or row store databases and with correlation databases, which use a value-based storage structure. II. History of Column Oriented Database Column stores or transposed files have been implemented from the early days of DB’S development. TAXI was the first application of a column-oriented database storage system with focus on Information-retrieval In biology In 1969.

Statistics Canada Implemented the RAPID system In 1976 and used It for processing and retrieval of the Canadian Census of Population and Housing as well as several other statistical applications. RAPID was shared with other statistical organizations throughout the world and used widely in the sass. It continued to be used by Statistics Canada until the sass. KID was the first commercially available column oriented database developed In 1993 followed in 1995 by Sybase IQ. However, that has changed rapidly since about 2005 with many open source and commercial Implementations. Ill.

Working of Column Oriented Database A relational database management system provides data that represents a two- dimensional table, of columns and rows. For example, a database might have this table: Ample Lasted Firestone Salary 10 Smith Joe 12 Jones Mary 11 Johnson Cathy Bob 55000 This simple table includes an employee identifier (Ample), name fields (Lasted and Firestone) and a salary (Salary). This two-dimensional format exists only in theory, in practice, storage hardware requires the data to be serialized into one form or another. The most expensive operations involving hard drives are seeks.

In order to improve overall performance, related data should be stored in a fashion to minimize the number of seeks. This is known as locality of reference, and the basic concept appears in a number of different contexts. Hard drives are organized into a series of blocks of a fixed size, typically enough to store several rows of the table. By organizing the data so rows fit within the blocks, and related rows are grouped together, the number of blocks that need to be read or sought is minimized. A column-oriented database serializes all of the values of a column together, then the values of the next column, and so on.

For our example table, the data would be stored in this fashion; 10: LOL Jones: Johansson: 00440000: 001 In this layout, any one of the columns more closely matches the structure of an index in a row-based system. This causes confusion about how a column-oriented store “ is really Just” a row-store with an index on every column. However, it is the mapping of the data that differs dramatically. In a row-oriented indexed system, the primary key is the rowed that is mapped to indexed data. In the column-oriented system primary key is the data, mapping back to rowdies. 3] This may seem subtle, but the difference can be seen in his common modification to the same store: … ; Smith: 001 Jones: 002, Johnson: 003; As two of the records store the same value, “ Jones”, it is possible to store this only once in the column store, along with pointers to all of the rows that match it. For many common searches, like “ find all the people with the last name Jones”, the answer is retrieved in a single operation. Other operations, like counting the number of matching records or performing math over a set of data, can be greatly improved through this organization.

Whether or not a column-oriented system will be more efficient in operation depends heavily on the workload being automated. It would appear that operations that retrieve data for objects would be slower, requiring numerous disk operations to collect data from multiple columns to build up the record. However, these whole-row operations are generally rare. In the majority of cases, only a limited subset of data is retrieved. In a rolodex application, for instance, operations collecting the first and last names from many rows in order to build a list of contacts is far more common than operations reading the data for any single address.

This is even more true for writing data into the database, especially if the data tends to be “ sparse” with many optional columns. For this reason, column stores have demonstrated excellent real-world performance in spite of any theoretical disadvantages. [4]This is a simplification. Moreover, partitioning, indexing, caching, views, OLAP cubes, and transactional systems such as write ahead logging or either system. That said, online transaction processing (ALTO)-focused READS systems are more row-oriented, while online analytical processing (OLAP)-focused systems are a balance of row-oriented and column-oriented. ‘ V. Top Five Market

Selling Column Oriented Database Sybase A highly optimized analytics server designed specifically to deliver superior performance for mission-critical business intelligence, analytics and data warehousing solutions on any standard hardware and operating system. Its column oriented grid-based architecture, patented data compression, and advanced query optimizer delivers high performance, flexibility, and economy in challenging reporting and analytics environments. Fig. Sybase Essentially a data partitioned, index based storage technology, Sybase Sis’s engine offers several key features which include: Web enabled analytics

Communications & Security Fast Data Loading Query Engine supporting Full Text Search Column Indexing Sub System Column Storage Processor User Friendly CHI based Administration & Monitoring Multiplex Grid Architecture Information Live-cycle management The Sybase IQ Very Large Data Base (BLVD) option provides partitioning and placement where a table can have a specified column partition key with value ranges. This partition allows data that should be grouped together to be grouped together and separates data where they should be separated. The drawback to this methodology is that it is not always known which is which.

Infighter Offering both a commercial (IEEE) and a free community (ICE) edition, the combination of a column oriented database with their Knowledge Grid architecture delivers a self- managed, scalable, high performance analytics query platform. Allowing TPTB using a single server, their industry-leading data compression (10: 1 up to 40: 1) significantly reduces storage requirements and expensive hardware infrastructures. Delivered as a Myself engine, Infighter runs on multiple operating systems and processors needing only a minimum of bib of RAM (however Bibb is a recommended starting point).

Avoiding partition schemes, Infighter data is stored in data packs, each node containing pre-aggregated statistics about the data stored within them. The Knowledge Grid above provides related metadata providing a high level view of the entire content of the database. Indexes, projections, partitioning or aggregated tables are not needed as these metadata statistics are managed automatically. The granular computing engine processes queries using the Knowledge Grid information to optimize query processing eliminating or significantly reducing the amount of data required for decompressing and access to answer a query.

Some queries may not need to access the data at all, finding instead the answer in the Knowledge Grid Fig. Infighter The Infighter Data Loader is highly efficient so data inserts are very fast. This performance gain does come at a price so avoid updates unless absolutely necessary, design De-normalized tables, and don’t plan on any deletes. New features to the data loader include a reject option which allows valid rows to commit while invalid rows are logged. Fig. Query Time of Myself and Infighter This is highly useful when loading millions of rows and only having a few rows with bad data.

Without this feature the entire data load would be rolled back. Vertical (HP) Recently acquired by Hewlett Packard, this platform was purpose built from the ground up to enable data values having high performance real-time analytics needs. With extensive data loading, queries, columnar storage, MAP architecture, and data compression features, diverse communities can develop and scale with a seamless integration ecosystem. Fig. Vertical Claiming elasticity, scale, performance, and simplicity the Vertical analytics platform uses transformation partitioning to specify which rows belong together and realism’s for speed.

Several key features include: Columnar Storage & Execution Real-Time Query & Loading Scale-out MAP Architecture Automatic High Availability Aggressive Data Compression Extensible In-Database Analytics Framework In-Database Analytics Library Database Designer & Administration Tools Native Bal & TTL support for Unprepared & Hoodoo The Vertical Optimizer is the brains of the analytics platform producing optimal query execution plans where several choices exist.

It does this through traditional considerations like disk 1/0 and further incorporates CAP], memory, network, nonoccurrence, parallelism factors and the unique details of the columnar operator and runtime environment. Parcel Analytic-driven companies need a platform, not Just a database where speed, agility, and complexity drive the data ecosystem. The Parcel Analytic Platform streamlines the delivery of complex business decisions through its high performance analytic database. Designed for speed, its extensible framework supports on-demand integration and embedded functions.

The Parcel Database (PAD) present four main components: the ‘ Leader’ node, the ‘ Compute’ node, the Parallel Communications Fabric, and an optional Storage Area Network (SAN). The ‘ Leader’ controls the execution of the ‘ Compute’ nodes and all nodes communicate with each other via the ‘ Fabric’ running on standard ex. Linux servers. Each ‘ Compute’ node is subdivided into a set of parallel processes called ‘ slices’ that include a CPU core, and provides a low-level MAP network protocol for increased performance. Fig.

Parcel Key PAD features include: High Performance & Scalability Columnar Orientation Extensible Analytics Query Compilation High Availability Solution Simplicity Fig. Parcel Requirements Parcel Integrated Analytics Library and Extensibility Framework incorporates advanced functions along with an API to add own functions to help address complex business problems right in the core database enabling customers to focus upon their specific data complexities. Microsoft SQL Server 2012 Microsoft has now embraced the columnar database idea.

The latest SQL Server release 2012 includes explicitly, a column-store index feature that stores data similar to a column-oriented DB’S. While not a true column oriented database, this technique allows for the creation of a memory optimized index that groups and tortes data for each column then and Joins them together to complete the index. For certain types of queries, like aggregations, the query processor can take advantage of the column-store index to significantly improve execution times.

Column store indexes can be used with partitioned tables providing a new way to think about how to design and process large datasets. The column-store index can be very useful on large fact tables in a Star scheme improving overall performance, however the cost model approach utilized may choose the column-store index for a table when a row based index would have been better. Fig. Microsoft SQL Server Architecture Using the query hint will work around this if it occurs.

When data is stored with a column-store index, data can often be compressed more effectively over a row based index. This is accomplished as typically there is more redundancy within a column than within a row. Higher compression means less 10 is required to retrieve data into memory which can significantly reduce response times. V. Other Implementations of Column Oriented Database Greenroom Database Callout Infinite IBM DUB Accumulator Dratted Sensate Sybase IQ Mounted Recipe Swirl KID VI. Advantages of Column Oriented Database

High performance on aggregation queries (like COUNT, SUM, AVGAS, MIN, MAX) Highly efficient data compression and/or partitioning True scalability and fast data loading for Big Data Accessible by many 3rd party Bal analytic tools Fairly simple systems administration Due to their aggregation capabilities which compute large numbers of similar data items, column oriented databases offer key advantages for certain types of systems, including: Data Warehouses and Business Intelligence Customer Relationship Management (CRM) Library Card Catalogs Ad hoc query systems VI’.

Disadvantages of Column Oriented Database Transactions are to be avoided or Just not supported Queries with table Joins can reduce high performance Record updates and deletes reduce storage efficiency Effective partitioning/indexing schemes can be difficult to design VIII. Row vs. Column Oriented Database Comparisons between row-oriented and column-oriented data layouts are typically concerned with the efficiency of hard-disk access for a given workload, as seek time is incredibly long compared to the other delays in computers.

Sometimes, reading a megabyte of sequentially stored data takes no more time than one random access. 5] Further, because seek time is improving much more slowly than CPU power, this focus will likely continue on systems that rely on hard disks for storage. Following is a set of oversimplified observations which attempt to paint a picture of the trade-offs between column- and row-oriented organizations. Unless, of course, the application can be reasonably assured to fit most/all data into memory, in which case huge optimizations are available from in-memory database systems. . Column-oriented organizations are more efficient when an aggregate needs to be computed over many sows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can be faster than reading all data. 2. Column-oriented organizations are more efficient when new values of a column are supplied for all rows at once, because that column data can be written efficiently and replace old column data without touching any other columns for the rows. 3.

Row-oriented organizations are more efficient when many columns of a single row are required at the same time, and when row-size is relatively small, as the entire row can be retrieved with a single disk seek. . Row-oriented organizations are more efficient when writing a new row if all of the row data is supplied at the same time, as the entire row can be written with a single disk seek. In practice, row-oriented storage interactive transactions. Fig. Row vs. Column Oriented Database Column-oriented storage layouts are well-suited for OLAP-like workloads (e. . , data warehouses) which typically involve a smaller number of highly complex queries over all data (possibly terabytes). ‘ X. Column Oriented Database in Big Data Project Columnar databases can be very helpful in big data project. Relational databases are row oriented, as the data in each row of a table is stored together. In a columnar, or column-oriented database, the data is stored across rows. Although this may seem like a trivial distinction, it is the most important underlying characteristic of columnar databases.

It is very easy to add columns, and they may be added row by row, offering great flexibility, performance, and scalability. When we have volume and variety of data, we might want to use a columnar database. It is very adaptable; we simply continue to add columns. One of the most popular columnar databases is Habeas. It, too, is a project in the Apache Software Foundation distributed under the Apache Software License v. O. Habeas uses the Hoodoo file system and Unprepared engine for its core data storage needs.

The design of Habeas is modeled on Google’s Vegetable. Therefore, implementations of Habeas are highly scalable, sparse, distributed, persistent multidimensional sorted maps. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterrupted array of bytes. When big data implementation requires random, real-time read/write data access, Habeas is a very good solution. It is often used to store results for later analytical processing.

Important characteristics of Habeas include the following: Consistency: Although not an “ ACID” implementation, Habeas offers strongly consistent reads and writes and is not based on an eventually consistent model. This means we can use it for high-speed requirements as long as we do not need the “ extra features” offered by READS like full transaction support or typed columns. Sharing: Because the data is distributed by the supporting file system, Habeas offers transparent, automatic splitting and redistribution of its content.

High availability: Through the implementation of region servers, Habeas supports LANA and WAN failover and recovery. At the core, there is a master server responsible for monitoring the region servers and all metadata for the cluster. Client API: Habeas offers programmatic access through a Java API. Support for IT operations: Implementers can expose performance and other metrics through a set of built-in web pages. Habeas implementations are best suited for High-volume, incremental data gathering and processing Real-time information exchange (for example, messaging)

Frequently changing content serving X. Industries to benefit from Column Oriented Database Telecommunications: While a consumer is waiting on the phone with a customer service agent, the agent must search the Consumer’s Call Detail Records (CDR), which may have over one hundred columns. However to get answers to the caller’s questions, a typical query needs data from only a few of the columns. Columnar database is able to decrease response time to the customer by reducing input and output.

Financial Services: When a bank marketing manager uses CRM data to more effectively personalize each ales contract he needs less than 10 attributes about the customer, such as customer number, demographic characteristics, last product purchased, and last channel used. However, CRM system captures hundreds of customer attributes. With columnar database, the amount of data read from the customer record is reduced by 90 percent, because only the 10 required attributes will be read, not the entire row of hundred attributes.

This capability supports high performance, millisecond response time to queries required for inbound market. Retail: A purchasing agent is ordering products for a chain of stores and is only interested n ordering a selected number of items, based on specific SKU numbers. However, retail demand chain management systems store a broad set of attributes for each item, including the date, store locations and SKIS to meet a variety of reporting requirements. The business user only wants data relevant to the question and does not require all the metrics.

The columnar database only reads the data referenced in the agent’s question, driving higher performance and lower processing costs compared to reading all of the columns in the table. X’. Conclusion Thus, columnar database is a key technology that delivers high business value by eloping enterprises adapt their information infrastructure to the evolving demands for timely, reliable intelligence to run the business. In addition, it will have far- reaching implications for the design of systems, and offer major cost savings affecting higher power and cooling requirements.