Column-Oriented Databases, an Alternative for Analytical Environment

Gheorghe MATEI
Romanian Commercial Bank, Bucharest, ROMANIA
George.matei@bcr.ro

It is widely accepted that a data warehouse is the central place of a Business Intelligence system. It stores all data that is relevant for the company, data that is acquired both from internal and external sources. Such a repository stores data from more years than a transactional system can do, and offer valuable information to its users to make the best decisions, based on accurate and reliable data. As the volume of data stored in an enterprise data warehouse becomes larger and larger, new approaches are needed to make the analytical system more efficient. This paper presents column-oriented databases, which are considered an element of the new generation of DBMS technology. The paper emphasizes the need and the advantages of these databases for an analytical environment and make a short presentation of two of the DBMS built in a columnar approach.

Keywords: *column-oriented database, row-oriented database, data warehouse, Business Intelligence, symmetric multiprocessing, massively parallel processing.*

1 Introduction

In the evolution of computing science, three generations of database technology are identified since the 60's till nowadays. The first generation started in the 60's and its main purpose was to enable disparate but related application to share data otherwise than passing files between them.

The publishing of "*A Relational Model of Data for Large Shared Data Banks*" by E. F. Codd marked the beginning of the second generation of DBMS (*database management systems*) technology. Codd's premise was that data had to be managed in structures developed according to the mathematical set theory. He stated that data had to be organized into tuples, as attributes and relations.

A third generation began to emerge in the late 90's and now is going to replace second-generation products. Multi-core processors became common, 64-bit technology is used largely for database servers, memory is cheaper and disks are cheaper and faster than ever before.

A recent IDC study [1] examines emerging trends in DBMS technology as elements of the third generation of such technology. It considers that, at the current rate of development and adoption, the following innovations will be achieved in the next five years:

- most data warehouses will be stored in a columnar fashion;
- most OLTP (*On-Line Transaction Processing*) databases will either be augmented by an in-memory database or reside entirely in memory;
- most large-scale database servers will achieve horizontal scalability through clustering;
- many data collection and reporting problems will be solved with databases that will have no formal schema at all.

This study examines how some innovations in database technology field are implemented more and more. Most of these technologies have been developed for at least ten years, but they are only now becoming widely adopted.

As Carl Olofson, research vice president for database management and data integration software research at IDC, said, "*many of these new systems encourage you to forget disk-based partitioning schemes, indexing strategies and buffer management, and embrace a world of large-memory models, many processors with many cores,* *clustered servers, and highly compressed columnwise storage*".

From the innovations that the study considers that will be achieved in the next years, this paper presents the columnar data storage.

2. The need for column-oriented databases

The volume of data acquired into an organization is growing rapidly. So does the number of users who need to access and analyse this data. IT systems are used more and more intensive, in order to answer more numerous and complex demands needed to make critical business decisions. Data analysis and business reporting need more and more resources. Therefore, better, faster and more effective alternatives have to be found. Business Intelligence (BI) systems are proper solutions for solving the problems above. Decision-makers need a better access to information, in order to make accurate and fast decisions in a permanent changing environment. As part of a BI system, reporting has become critical for a company's business.

Years ago, reports prepared by analysts were addressed only to the company's executive management. Nowadays, reporting has become an instrument addressed to decision-makers on all organizational levels, aiming to improve the company's activity, to ensure decision quality, control costs and prevent losses.

As already mentioned, the volume of data acquired into a company is growing permanently, because business operations expand and, on the other hand, the company has to interact with more sources of data and keep more data online. More than ever before, users need a faster and more convenient access to historical data for analysing purposes. Enterprise data warehouses are a necessity for the companies that want to stay competitive and successful. More and more reports and ad-hoc queries are requested to support the decision making process. At the same time, companies have to run audit reports on their operational and historical data in order to ensure compliance [2].

These new demands add more pressures upon IT departments. More and more hardware resources are needed in order to store and manage an increasing volume of data. The increasing number of queries needs larger amounts of CPU cycles, so more processors, having a higher performance, must be added to the system

The size of the data warehouses storing this data is increasing permanently, becoming larger and larger. While five years ago the largest data warehouses were around 100 terabytes in size, now a data warehouse size at the petabyte level is no longer unusual. The challenge is to maintain the performance of these repositories, which are built, mostly, as relational structures, storing data in a row-oriented manner. The relational model is a flexible one and it has proven its capacity to support both transactional and analytical processing. But, as the size and complexity of data warehouses have increased, a new approach was proposed as an alternative on the row-oriented approach, namely storing data in a column-oriented manner. Unlike the row-oriented approach, where the data storage layer contains records (rows), in a column-oriented system it contains columns. This is a simple model, more adequate for data repositories used by analytical applications, with a wide range of users and query types.

Researches indicate that the size of the largest data warehouse doubles every three years. Growth rates of system hardware performance are being overrun by the need for analytical performance [3]. The volume of data needed to be stored is growing due to more and various requirements for reporting and analytics, from more and more business areas, increased time periods for data retention, a greater number of observations loaded in data warehouses and a greater number of attributes for each observation. This is true if taking into consideration only structured data. But nowadays, organizations collect a larger and larger volume of unstructured data, as images, audio and video files, which need a much greater storing space than structured data.

Row-oriented databases have been designed for transactional processing. For example, in the account management system of a bank, all attributes of an account are stored in a single row. Such an approach is not optimal in an analytical system, where a lot of read operations are executed in order to access a small number of attributes from a vast volume of data. In a row-oriented architecture, system performance, users' access and data storage become major issues very quickly [4]. As they are designed to retrieve all elements from several rows, row-oriented databases are not well suited for large scale processing, as needed in an analytical environment. As opposed to transactional queries, analytical queries typically scan all the database's records, but process only a few elements of them. In a column-oriented database all instances of a single data element, such as account number, are stored together so they can be accessed as a unit. Therefore, column-oriented databases are more efficient in an analytical environment, where queries need to read all instances of a small number of data elements.

System performance enhances spectacularly in a column-oriented solution, because queries search only few attributes, and they will not scan the attributes that are irrelevant for those queries. Requested data is found faster, because less sort operations have to be performed.

A typical feature of evolved BI systems is their capability to make strategic business analyses, to process complex events and to drill deeply into data. As the volume of data becomes impressive and performance demands required by users are likely to outpace, it is obviously that row-oriented relational database management systems stopped to be the solution for implementing a BI system having powerful analytical and predictive capabilities. A new model tends to come into prominence as an alternative on developing analytical databases, namely one that manages data by columns.

A column-oriented DBMS stores data in a columnar manner and not by rows, as classic DBMS do. In the columnar approach, each attribute is stored in a separate table, so successive values of that attribute are stored consecutively. This is an important advantage for data warehouses where, generally, information is obtained by aggregating a vast volume of data. Therefore, operations as MIN, MAX, SUM, COUNT, AVG and so forth are performed very quickly [5].

When the tables of a database are designed, their columns are established. The number of rows will be determined when the tables will be populated with data. In a row-oriented database, data is stored in a tabular manner. The data items of a row are stored one after another; rows are also stored one after another, so the last item of a row is followed by the first item of the next row.

In a column-oriented database, the data items of a column are stored one after another, and also are the columns; so the last item of a column is followed by the first item of the next column.

3. Differences between the row-oriented and column-oriented approaches

In a typical relational DBMS, data is stored and managed as rows, each row containing all the attributes of an element of that entity (table). Such systems are used by transactional applications which, at a certain moment, generate or modify one or a small number of records. Unlike transactional applications, which use all, or almost all the attributes of a record, analytical and BI applications scan few attributes (columns) of a vast number of records. Most often, they have to aggregate data stored in those columns in order to meet the users' demands. Because of the row-oriented structure of the database, the entire record has to be read in order to access the required attributes. This fact causes the reading of a vast amount of unuseful additional data in order to access the requested information.

Figure 1 shows that much more data than needed is read to satisfy the request for the total volume of term deposits opened at the branches in Bucharest.

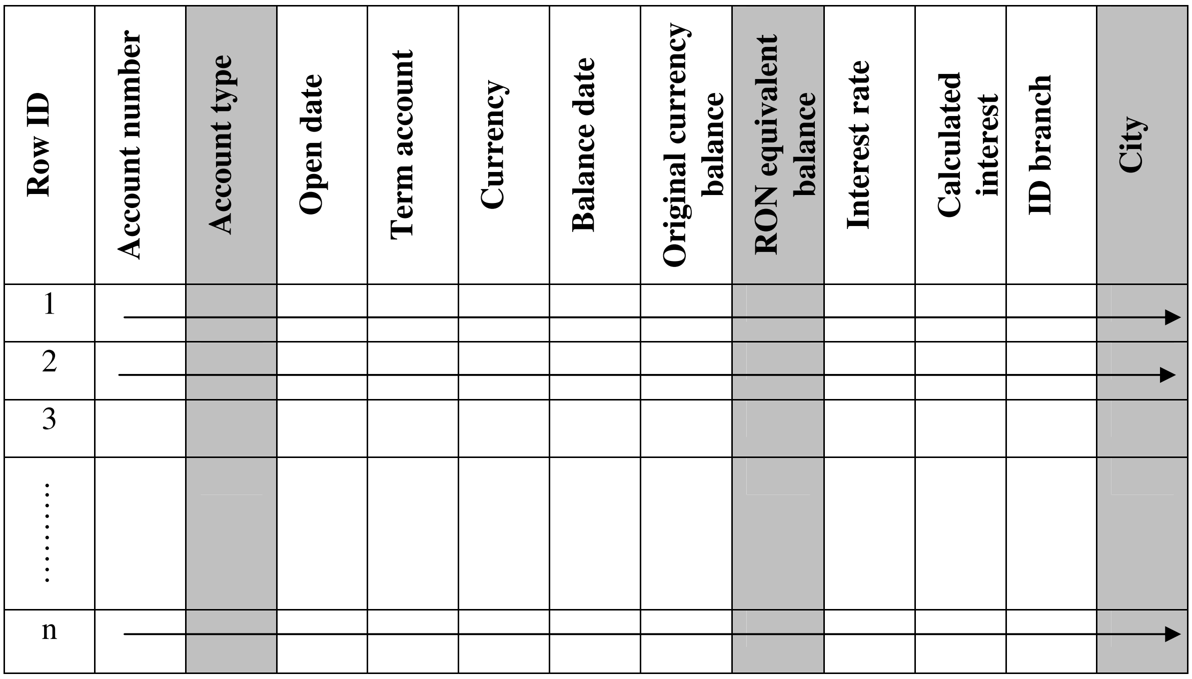


Fig. 1. Analytical request in a row-oriented database

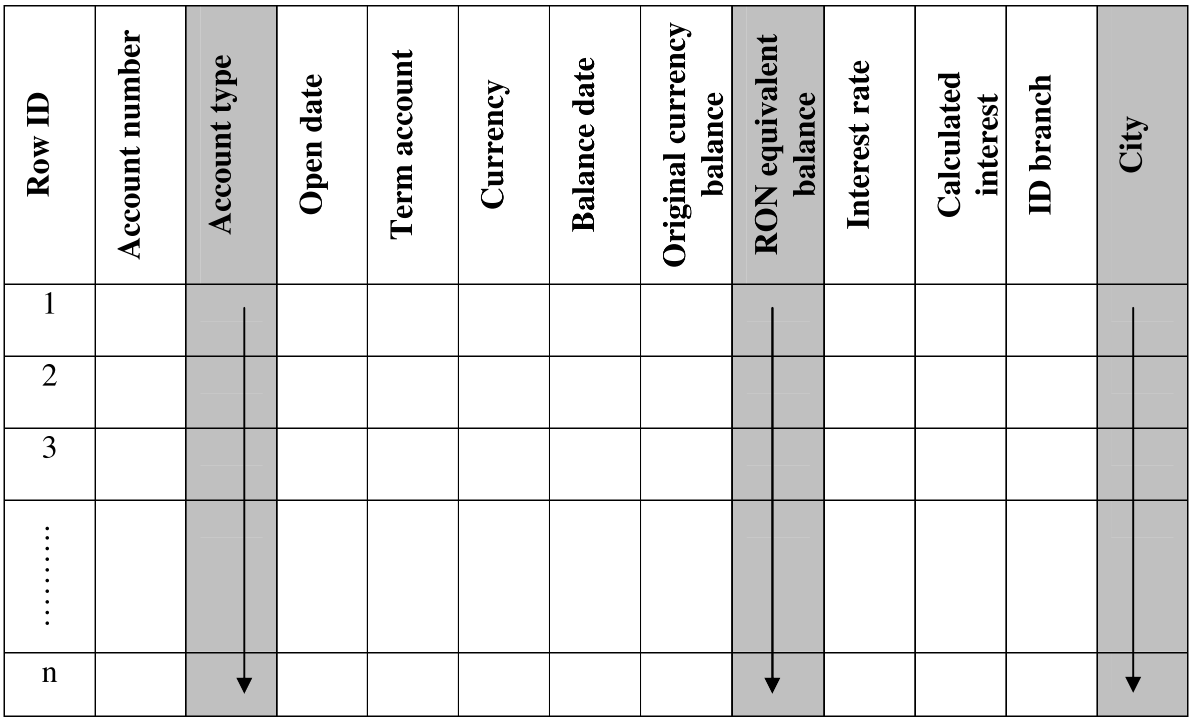
Row-oriented databases were designed for transactional applications, and they are optimized for retrieval and processing of small data sets. Seldom, to support analytical requests, it is necessary to build additional indexes, pre-aggregating data structures, or special materialized views and cubes. All these aspects require additional processing time and data storage. However, because they are built in order to provide quickly results for queries that were known at the design stage, they will not have the same performance when ad-hoc queries, that were not foreseen before, are performed.

The business demands require the storage of many data items. But any user wants to get information as soon as possible. Therefore, a proper solution for data organization has to be implemented in order to ensure a good performance of the system.

Several technical solutions can be used to improve system performance, such as partitioning, star indexes, query pre-processing, bitmap and index joins, or hashing. These solutions aim to offer support for more specific data retrieval, but they still have to examine the entire content of a row.

Taking into consideration those presented above, a new approach was proposed, to store data along columns. In such data organization, each column is stored separately and the system selects only the columns requested by users. In every column data is stored in row order, so that the 50th entry for each column belongs to the same row, namely the 50th row.

Figure 2 shows that the same query as those in figure 1 reads less data in a column-oriented system, in order to provide the same result. No additional indexes have to be built for improving query performance, because every column forms an index. This fact reduces the number of I/O operations and enables quick access to data, without the need to read the entire database. Data from each column is stored contiguously on disk. Column values are joined into rows based on their relative position in each column. As a result of the column-oriented architecture, only those columns needed for a specific query are read form disk. Because in an analytical environment most of queries need to retrieve only few columns, this vertical partitioning approach produces important I/O savings. This fact contributes to system performance improvement, as regards the query execution time.



Row ID	**Account number**	**Account type**	**Open date**	**Term account**	**Currency**	**Balance date**	**Original currency balance**	**RON equivalent balance**	**Interest rate**	**Calculated interest**	**ID branch**	**City**
1		↓						↓				↓
2												
3												
....												
n												

Fig. 2. Analytical request in a column-oriented database

<u>Note.</u> Figure 2 presents a reunion of the tables in a column-oriented database. In fact, each table has two columns: one containing the row ID, and the other, the values of the appropriate attribute. Because of the limited space on the page, the row ID column is not multiplied for every table, and the attribute columns are close together.

Comparing the two figures above, it's easy to observe that the same request has to read more data in a row-oriented structure than in a column-oriented one. In order to read a certain attribute in a row-oriented structure, all the adjacent attributes have to be read, even if they are not interesting for the requester. In a column-oriented structure, since all values of an attribute are stored together, consecutively, this problem doesn't exist [6].

A column-oriented database is faster than a row-oriented one, because its processing is not affected by unnecessary content of rows. As long as many database tables can have dozens of columns and most business requests need only a few of them, the columnar approach is a proper solution for analytical systems.

Talking about the efficiency of a column-oriented system, some remarks are to be made concerning processing time. Thus, such a system is more efficient when it's necessary to aggregate a large number of rows, but only a small number of columns are requested. If many columns of a small number of rows have to be processed, a row-oriented system is more efficient than a column-oriented one. The efficiency is even greater when row size is relatively small, because the entire row can be retrieve with a single disk seek.

Updating an entire column at once is faster in a column-oriented database. All the data of that column is modified through only one updating command, without the need to read all columns of each row. But writing or updating a single row is more efficient in a row-oriented database if all attributes are supplied at the same time, because the entire row can be written with a single disk access, whereas writing to multiple columns requires multiple writes.

SQL queries in a column-oriented database are identical with those in a row-oriented database, without any modification. What is different, is the way that the database administrator has to think about data. While in a row-oriented database he thinks in terms of individual transactions, in a column-oriented database he has to think in terms of similar items derived from sets of transactions. From the indexing point of view, he has to pay more attention to the cardinality of the data, because an index is related with a subject, such as the balance account, and not with an entire transaction with all its fields.

4. Advantages of the column-oriented approach

Column-oriented databases provide important advantages towards the row-oriented ones, some of them being presented below.

Column-oriented databases provide a **better performance** for analytical requests. In the row-oriented approach, the system performance decreases significantly as the number of simultaneous queries increases. Building additional indexes in order to accelerate queries becomes uneffective with a large number of diverse queries, because more storage and CPU time are required to load and maintain those indexes. In a column-oriented system indexes are built to store data, while in a row-oriented system they represent the way to point to the storage area that contains the row data. As a result, a column-oriented system will read only the columns required in a certain query.

On the other hand, as they store data as blocks by columns rather than by rows, the actions performed on a column can be completed with less I/O operations. Only those attributes requested by users are read from disk. Although a row-oriented table can be partitioned vertically, or an index can be created for every column so it could be accessed independently, the performance is significantly lower than in a column-oriented structure [7]. And taking into consideration that I/O operations are the bottleneck of a database application, the column-oriented approach proves its superiority against the row-oriented one.

Unlike the row-oriented approach, the column-oriented approach allows **rapid joins and aggregations**. Tables are already sorted, so there is no need to sort them before merge or join them. In addition, accessing data along columns allows incremental data aggregation, which is very important for BI applications. In addition, this approach allows parallel data access, improving the system performance. Thereby, complex aggregations can be fulfiled by the parallel processing of columns and then joining data in order to achieve the final result.

Column-oriented databases need a **smaller disk space** to store data than row-oriented databases. To accommodate the sustained increase of volume of data, additional structures – as indexes, tables for pre-aggregation and materialized views, are built in row-oriented systems. Column-oriented databases are more efficient structures. They don't need additional storage for indexes, because data is stored within the indexes themselves. Bitmap indexes are used to optimize data store and its fast retrieval. That's why in a column-oriented database queries are more efficient than in a row-oriented one.

Moreover, a higher data compression rate can be achieved in a column-oriented database than in a row-oriented one. It is well known that compression is more effective when repeated values are presented, and values within a column are quite similar to each other. A column-oriented approach allows the ability to highly compress the data due to the high potential for the existence of similar values in adjacent rows of a certain column. In a row-oriented database, values in a row of a table are not likely to be very similar; therefore, they cannot be compressed as efficient as in a column-oriented database.

Concerning the repository presented in figures 1 and 2, there is no doubt that many repeated values will be found within the CITY column, but no repetition will be found between CITY and another attribute in a row.

Data loading is a **faster process** if it's executed in a column-oriented database than in a row-oriented one. As known, to load data in a data warehouse involves to perform more activities. Data is extracted from source systems and loaded into a staging area. This is the place where data transformations and joins are performed in order to denormalize data and load it into the data warehouse as fact and dimension tables. Then the needed indexes and views are created. In a row-oriented structure, all data in a row (record) is stored together, and indexes are built taking into consideration all the rows. In a column-oriented structure, data of each column is stored together and the system allows the parallel loading of the columns, ensuring a shorter time for data loading.

Taking into consideration the features presented above, it can be stated that a column-oriented database is a scalable environment that keeps providing fast queries when the volume of data, the number of users and the number of simultaneous queries are increasing.

But this thing doesn't mean that all repositories have to be built in a columnar manner. A column-oriented architecture is more suitable for data warehousing, with selective access to a small number of columns, while a row-oriented one is a better solution for OLTP systems. For an OLTP system, which is heavily loaded with interactive transactions, a row-oriented architecture is well-suited. All data for a certain row is stored as a block. In such an architecture, all the attributes of a record are written on disk with a single command, this thing ensuring a high performance for writing operations. Usually, an operation in such a system creates, queries or changes an entry in one or more tables. For an OLAP (*On-Line Analytical Processing*) system, designed for analytical purposes, which involve processing of a large number of values of few columns, a column-oriented architecture is a better solution. A data warehouse, which is the central place of an analytical system, must be optimized for reading data. In such an architecture, data for a certain column is stored as a block, so the analytical queries, which usually aggregate data along columns, are performed faster. A column-oriented system reads only the columns required for processing a certain query, without bringing into memory irrelevant attributes. Such an approach provides important advantages concerning the system performance, because typical queries involve aggregation of large volumes of data [8].

5. Examples of column-oriented database systems

Besides the column-oriented approach, another important innovation applied in data warehousing consists in the way in which data is processed. Two major techniques are used to design a data warehouse architecture: *symmetric multiprocessing* (SMP) and *massively parallel processing* (MPP). It couldn't be certified the superiority of one approach against the other. Each of these solutions has its own supporters, because both of them are valid approaches and, when properly applied, lead to notable results.

Two database systems are presented in the next sections, each of them using one of the two types of architecture.

5.1.Sybase IQ

Sybase IQ is a high-performance decision support server designed specifically for data warehousing. It is a column-oriented relational database that was built, from the very beginning, for analytics and BI applications, in order to assist reporting and decision support systems. This fact offers it several advantages within a data warehousing environment, including performance, scalability and cost of ownership benefits.

A Sybase IQ database is different from a conventional relational database, because its main purpose is to allow data analysis and not its writing or updating. While in a conventional database the most important thing is to allow many users to update the database instantly and accurately, without interfering with one another, in a Sybase IQ database the most important thing is to ensure fast query response for many users.

Sybase IQ has a column-oriented structure and has its own indexing technology that ensures high performance to reporting and analytical queries, which are performed, as its developers state, up to 100 times faster than in a traditional DBMS. Sybase IQ offers enhanced features for data loading. Its flexible architecture ensures that the system will provide rapidly the requested information, within seconds or minutes at most, no matter how many queries are issued.

Sybase IQ has enhanced compression algorithms, which reduce the disk space necessary for data storing from 30 to 85 percent, depending on data's structure. A significant cost reduction results due to this fact. As already mentioned, storing data in a column-oriented manner increases the similarity of adjacent records on disk, and values within a column are often quite similar to each other. More tests confirmed that Sybase IQ requires 260 Terabytes of physical storage for storing 1000 Terabytes of row input data. A row-oriented database requires additional storage for indexes and, in the example above, this additional storage can reach up to 500 Terabytes [2]. In addition, Sybase IQ allows operating directly with compressed data [9], with a positive impact on processing speed.

As opposed to traditional databases, Sybase IQ is easier to maintain, and the tuning needed to get a higher performance requires less time and hardware resources. It doesn't need specialized schemas (dimensional modeling) to perform well. Sybase IQ is built upon open standards, so the integration and interoperability with other reporting systems and dashboards are easy to achieve. In order to enhance the performance of ad-hoc queries, it delivers more specialized indexes, such as: indexes for low cardinality data, grouped data, range data, joined columns, textual analysis, real-time comparisons for Web applications, date and time analysis [10]. Furthermore, every field of a row can be used as an index, without the need to define conventional indexes. Due to these indexes, analytical queries focus on specific columns, and only those columns are loaded into memory. This reduces very much the need for expensive, high performance disk storage and, at the same time, the number of I/O operations.

Offering an enhanced scalability, Sybase IQ can be used by a large number of users – hundreds and even thousands – which can access vast volumes of data, from a few gigabytes to several hundred terabytes.

Sybase IQ offers a fast and flexible access to information. As already mentioned, it is designed for query processing and ad-hoc analysis. As opposed to traditional data warehouses, it does not require data to be pre-aggregated in order to analyse it. Therefore, users can efficiently and quickly analyse atomic level data. With Sybase IQ users can analyse the business performance and can track the company's KPI (*key performance indicators*). Comparing with other products, it provides better solutions for measuring the business results, managing the customer relationship and ensuring financial controls.

Several intelligent features are integrated in Sybase IQ architecture. These features, such as the use of symmetric multiprocessing (SMP), enhance database's performance and reduce its maintenance overhead. A SMP architecture offers an increased performance because all processors can equally access the database's tables.

Sybase IQ architecture (figure 3) consists of multiple SMP nodes. Some of them are used only for reading data (*read-only nodes*), while other can be used both for reading and writing data (*read/write* *nodes*). However, each node can be flexibly designated as a read or write node, according to the requirements at a given moment. Thus, if an overnight batch has to be executed in order to load or update a large volume of data, it's a good idea to make all the read/write nodes to operate as write nodes, even if they run as read nodes during the day. In addition, this architecture can be scaled up incrementally, by adding new nodes as needed.

As shown in figure 3, Sybase columnar store allows storing data in more repositories, depending on its age, and it can be easily scaled out. Data can be loaded through real time loads or batch ETL (*Extract, Transform, Load*) processes. Starting with the release of version 15 of Sybase IQ, a new "*load from client*" option has been added. This option allows loading data from external sources, via ODBC, JDBC, ADO.Net, OLE DB and DBLib. Data, which can be encrypted through different methods, is rapidly accessed for analytical or reporting purposes.

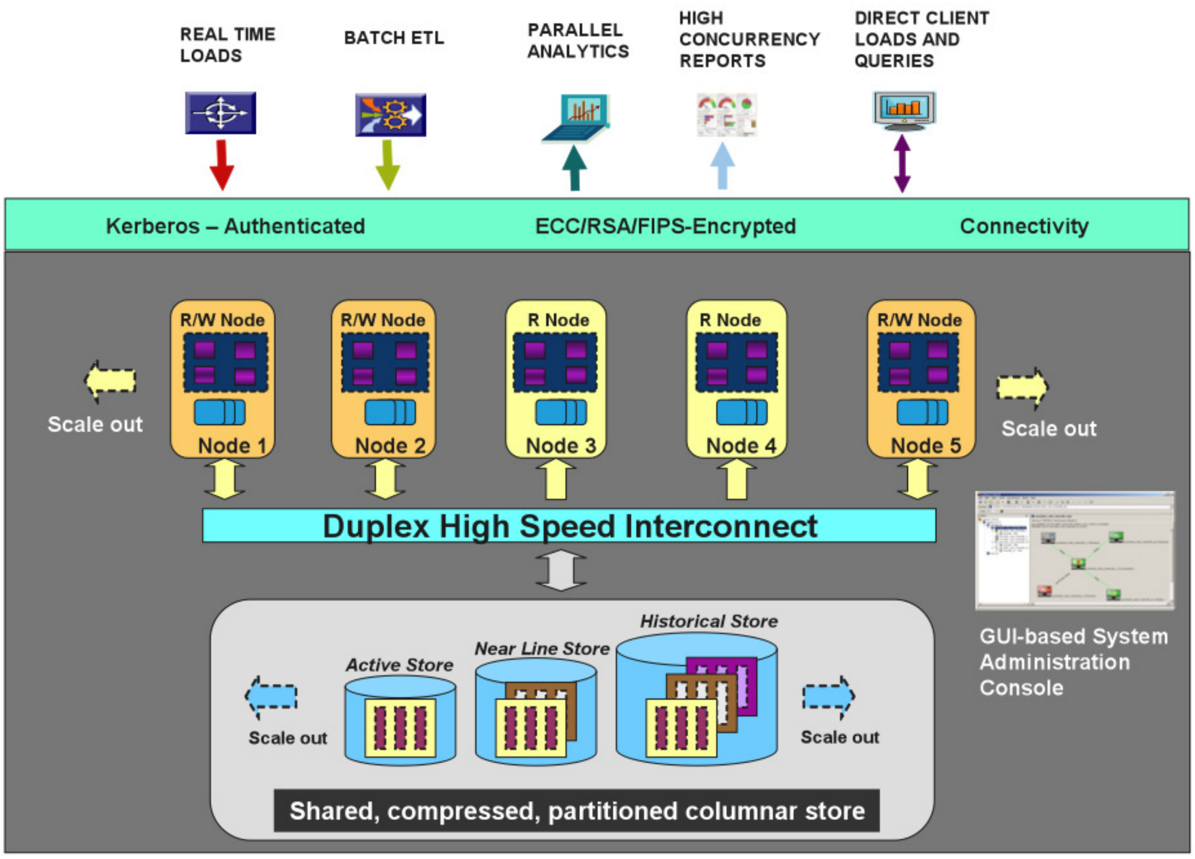


Fig. 3. Sybase IQ architecture (source: [10])

Sybase IQ enables data to be managed more efficiently than in a traditional database, built in a row-oriented approach. Complex analyses are run much faster. High data compression reduces storage costs, and vast volumes of data can be processed much more quickly.

5.2.Vertica

To gain competitive advantages and comply with new regulations, companies are obliged to develop enterprise data warehouses and powerful applications able to respond to more and more ad-hoc queries from an increasing number of users that need to analyse larger volumes of data, often in real time.

Vertica Analytic Database is a DBMS that can help in meeting these needs. It is a column-oriented database that was built in order to combine both column store and execution, as opposed to other solutions that are column-oriented only from storage point of view.

Designed by Michael Stonebraker, it incorporates a combination of architectural elements – many of them which have been used before in other contexts – to deliver a high-performance and low-cost data warehouse solution that is more than the sum of its elements.

Vertica is built in a massively parallel processing (MPP) architecture. In a MPP architecture, processors are connected with certain sets of data, data is distributed across the nodes of the network, and new processors can be added almost without limit. As data is partitioned and load into a server cluster, the data warehouse performs faster. Due to the MPP technology, the system performance and storage capacity can be enhanced simply by adding a new server to the cluster. Vertica automatically takes advantages of the new server without the need for expensive and time consuming upgrades.

While many of the new data warehouses use only MPP technology or columnar approach, Vertica is the only data warehouse that includes both innovations, as well as other features. It is designed to reduce I/O disk operations and is written natively to support grid computing.

Because of the columnar approach, which reduces the expensive I/O operations, queries are 50 to 200 times faster than a row-oriented database. Its MPP architecture offers a better scalability, that can be achieved by adding new servers in the grid architecture.

In order to minimize the disk space needed to store a database, Vertica uses more compression algorithms, depending on data type, cardinality and sort order. For each column, the proper algorithm is automatically chosen, based on a sample of the data [11]. As data into a column is homogenous, having the same type, Vertica provides a compression ratio from 8 to 13 time relative to the size of the original input data.

Vertica decomposes the logical tables and physically stores them as groups of columns named "*projections*". According to this concept, data is stored in different ways, similar to materialized views. Each projection contains a subset of the columns of one or more tables, and is sorted on a different attribute. Vertica automatically selects the proper projection in order to optimize query performance. Due to the effective compression algorithms used by Vertica, multiple projections can be maintained, which concurs to the performance improvement of a large range of queries, including ad-hoc queries needed for exploratory analysis.

On the other hand, these projections serve as redundant copies of the data. Because data is compressed so efficiently, Vertica can use the disk space to store these copies to ensure fault tolerance and to improve concurrent and ad-hoc query performance. Partitioning data across the cluster, Vertica ensures that each data element is stored on two or more nodes. Thus, an intelligent data mirroring is implemented, named K-Safety, where *k* is the number of node failures that a given set of projections will not affect the system availability. In order to guarantee K-Safety, k+1 replicas of all projections are built. Each replica has the same columns, but they may have different sort order. K-Safety allows requests for data stored into failed nodes to be satisfied by corresponding projections on other nodes. Once a failed node is restored, projections on the other nodes are automatically used to repopulate its data.

The value of *k* has to be configured so that a proper trade-off between hardware costs and availability guarantees to be met. If necessary, a new node can be added in the grid, and Vertica will automatically allocate a set of objects to that node and it can begin processing queries, increasing database performance [12]. Conversely, a node can be removed and the database will continue to work, but at a lower rate.

As shown in figure 4, a single Vertica node is organized into a *hybrid store* consisting of two distinct storage structures: the **Write-Optimized Store** (WOS) and the **Read-Optimized Store** (ROS). Generally, the WOS fits into main memory and is designed to efficiently support insert and update operations. Data is stored as collections of uncompressed and unsorted columns, maintained in update order.

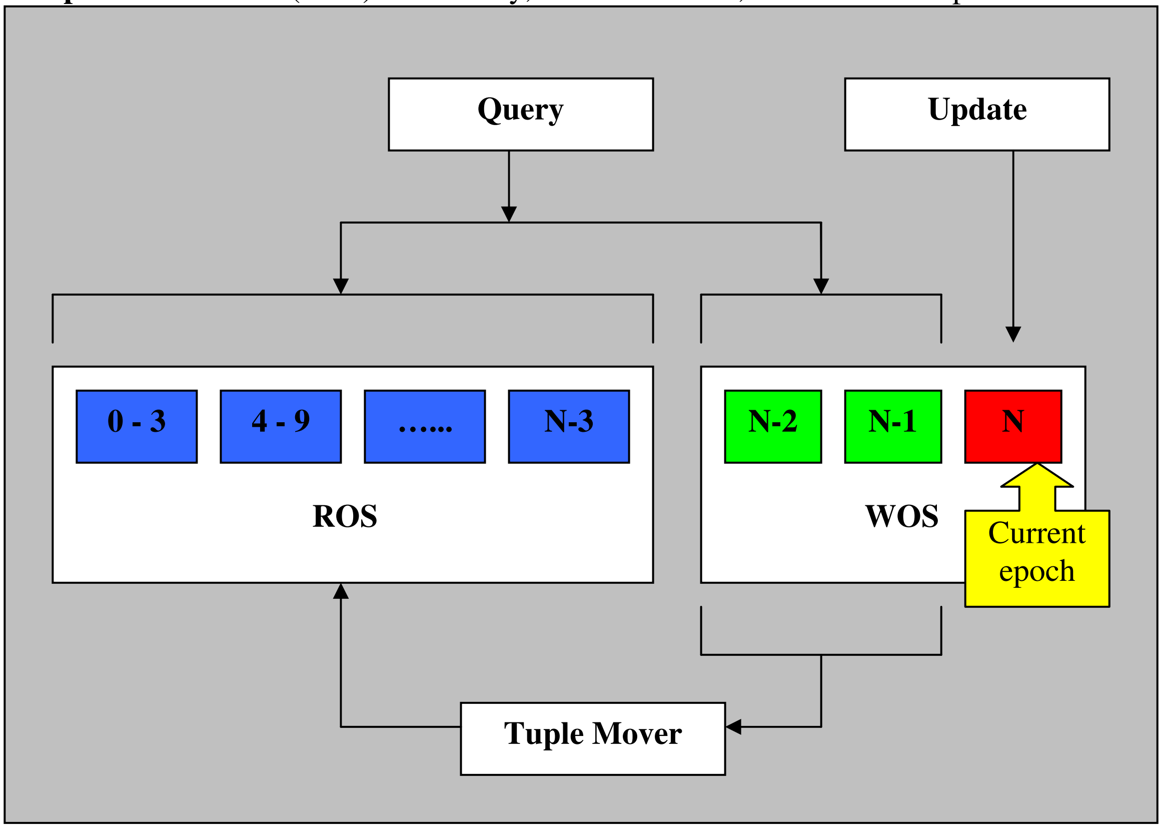


Fig. 4. Vertica storage model
(source: [11])

An asynchronous background process called the **Tuple Mover** moves data from WOS into the permanent disk storage in the ROS. The Tuple Mover operates on the entire WOS, sorting many records at a time and writing them to the ROS as a batch. Data in the ROS is sorted and compressed, so it can be efficiently read and queried.

Queries and updates do not interfere with one another. Updates are collected in time-based buckets called epochs. New updates are grouped in the current epoch until the transaction is committed. Data in older epochs is available for querying and moving into the ROS.

Because of the grid computing architecture, a query can by initiated on any node of the network. Vertica query planner decomposes the query according to the data stored into the involved nodes, and sends them the appropriate subqueries. Then it collects each node's partial result and composes them in order to offer to the requester the final answer.

In [13] a comparison is made between a 1.5 terabytes row-oriented data warehouse and a column-oriented database containing the same data and managed by Vertica Analytic Database. The results are presented in the table 1 bellow.

Table 1. Advantages of Vertica Analytic Database (source: [13])

	Row-oriented data warehouse	**Vertica Analytic Database**	**Vertica advantages**
Avg query response time	37 minutes	9 seconds	270x faster answers
Reports per day	30	1,000	33x more reports
Data availability	Next day	1 minute	Real-time views
Hardware cost	\$1.4M (2*6 servers + SAN)	\$50,000 (6 HP ProLiant servers)	1/28th of the hardware, built-in redundancy

All those presented above enable Vertica to manage larger volumes of historical data, analyse data at any level of detail, perform real-time analyses, conduct ad-hoc and short-lived business analytical projects, and build new analytic Software as a Service (SaaS) business.

6. Conclusions

For applications that write and update many data (OLTP systems), a row-oriented approach is a proper solution. In such an architecture, all the attributes of a record are placed contiguously in storage and are pushed out to disk through a single write operation. An OLTP system is a write-optimized one, having a high writing performance.

In contrast, an OLAP system, mainly based on ad-hoc queries performed against large volumes of data, has to be read-optimized. The repository of such a system is a data warehouse. Periodically (daily, weekly, or monthly, depending upon how current data must be), the data warehouse is load massively. Ad-hoc queries are then performed in order to analyse data and discover the right information for the decision making process. And for analytical applications, that read much more than they write, a column-oriented approach is a better solution.

Nowadays, data warehouses have to answer more and more ad-hoc queries, from a greater number of users which need to analyse quickly larger volumes of data.

Columnar database technology inverts the database's structure and stores each attribute separately, fact that eliminates the wasteful retrieval as queries are performed. On the other hand, much more data can be loaded in memory, and processing data into memory is much faster.

Column-oriented databases provide faster answers, because they read only the columns requested by users' queries, since row-oriented databases must read all rows and columns in a table. Data in a column-oriented database can be better compressed than those in a row-oriented database, because values in a column are much more homogenous than in a row. The compression of a column-oriented database may reduce its size up to 20 times, this thing providing a higher performance and reduced storage costs. Because of a greater compression rate, a column-oriented implementation stores more data into a block and therefore more data into a read operation. Since locating the right block to read and reading it are two of the most expensive computer operations, it's obviously that a column-oriented approach is the best solution for a data warehouse used by a Business Intelligence system developed for analytical purposes.

References

[1] Carl Olofson, The Third Generation of Database Technology: Vendors and Products That Are Shaking Up the Market, 2010, www.idc.com

[2] Sybase, Sybase IQ: The Economics of Business Reporting, White paper, 2010, www.sybase.com/files/White_Papers/Sybase-IQ-Business-Reporting-051109-WP.pdf

[3] David Loshin, Gaining the Performance Edge Using a Column-Oriented Database Management System, Sybase white paper, 2009, www.sybase.com

[4] Sybase, A Definitive Guide to Choosing a Column-based Architecture, White paper, 2008, www.information-management.com/white_papers/10002398-1.html

[5] William McKnight, Evolution of Analytical Platforms, Information Management Magazine, May 2009, www.information-management.com/issues/2007_58/analytics_business_intelligence_bi-10015353-1.html

[6] Daniel Abadi, Column-Stores For Wide and Sparse Data, 3rd Biennial Conference on Innovative Data Systems Research, January 7 – 10, 2007, Asilomar, California, USA, http://db.csail.mit.edu/projects/cstore/abadicidr07.pdf

[7] Daniel Abadi, Samuel Madden, Nabil Hachem, Column-Stores vs. Row-Stores: How Different Are They Really?, Proceedings of the 2008, ACM SIGMOD International Conference on Management of Data, Vancouver, Canada, http://portal.acm.org

[8] Mike Stonebraker, Daniel Abadi et al., C-Store: A column-oriented DBMS, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005, http://db.csail.mit.edu/projects/cstore/vldb.pdf

[9] Daniel Tkach, When the information is the business, Sybase white paper, 2010, www.sybase.com/files/white_papers

[10] Philip Howard, Sybase IQ 15.1, A Bloor InDetail Paper, 2009, www.it-director.com/business/innovation

[11] ***, Revolutionizing data warehousing in telecom with the Vertica Analytic Database, 2010, www.vertica.com/white-papers

[12] ***, The Vertica Analytic Database technical overview, 2010, www.vertica.com/white-papers

[13] ***, Increasing Enterprise Data Warehouse Performance, Longevity and ROI with the Vertica Analytic Database, 2010, www.vertica.com/white-papers

Gheorghe MATEI has graduated the Faculty of Planning and Economic Cybernetics in 1978. He achieved the PhD in Economic Cybernetics and Statistics in 2009, with a thesis on Business Intelligence systems in the banking industry. After a long career in the IT department, now he is working in the accounting and reporting department in Romanian Commercial Bank. His fields of interest include Business Intelligence systems, data warehousing, decision support systems, collaborative systems. He is a co-author of the book "*Business Intelligence Technology*" (2010), as well as author and co-author of several articles in journals, international databases and proceedings of national and international conferences in the mentioned domains.