International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

A Study on Columnar Databases

Adithya Viswanathan[1], Devanshu Mishra[1], Srividya MS[3]

[1]B.E. Department of Computer Science and Engineering, R.V. College of Engineering
[1]B.E. Department of Computer Science and Engineering, R.V. College of Engineering
[2] Assistant Professor, Department of Computer Science and Engineering, R.V. College of Engineering

---

Abstract - *Over the last few decades, the amount of data in an enterprise of an organization is growing exponentially. Along with that, there is an increase in the number of people who are trying to access and analyze such data and interpret results. More tools that offer quick and efficient methods for collection, storage, and querying for analysis of such data are the need of the hour. Column-oriented storage structures have a growing market share in the last few years. This architecture is preferred when there are usually copious dynamic queries and in high frequency. This survey paper details the architecture of column-store, the advantages, and limitations of what column-store can offer us, and how it fares when looked up against the traditional row-store.*

Key Words: **Column-Store, Databases, Row-Store, Relational database management, ERP, Querying**

1. INTRODUCTION

One of the largest hurdles faced by enterprises today is to provide an amalgamation of analysis features and essential reporting within services and applications that they provide on their platform. Adding to that, users need to adequately operate on the ever-increasing velocity of data without delays caused by re-optimization causing time delays related to evolving query patterns or data. It becomes necessary to support analytically intensive database workloads, the data mining processes, and inquisitive ad-hoc query statements, which were not expected during initial phases of planning. With the world rapidly changing, data is the purest form of currency. It is been created faster than ever, and in more volume than ever. Multiple problems develop due to this, one being storing such large volumes with variety. But the more important problem is accessing this data when the time comes efficiently in an eloquent manner for the true purpose it serves.

Databases primarily store data in two forms, row-store databases and column store. For the former, the data is stored in a sequence where each data the field values of each tuple is stored successively, while in the databases present using column-store, data exists according to the values of the column. [1]. Both are necessary for they serve different purposes. Row Store is optimized for writes and quite popular and the norm. Column Store is optimized for reading the data. In this paper, the column store database approach will be analyzed to see if it does help enterprises in their database requirements.

2. ARCHITECTURE OF COLUMNNAR DATABASES

Different workloads in both systems have explained the conventional business division into online analytical processing (OLAP) and online transaction processing (OLTP). While online transaction processing (OLTP) computations are characterized by a mix of writes and reads from a few rows at a time, online analytical processing (OLAP) applications are defined by complex read operations including joins and large longitudinal scans spanning only a few columns but several database rows. Usually, different systems handle these two workloads: data warehousing systems or business intelligence (BI) and TPS transaction processing systems [3].

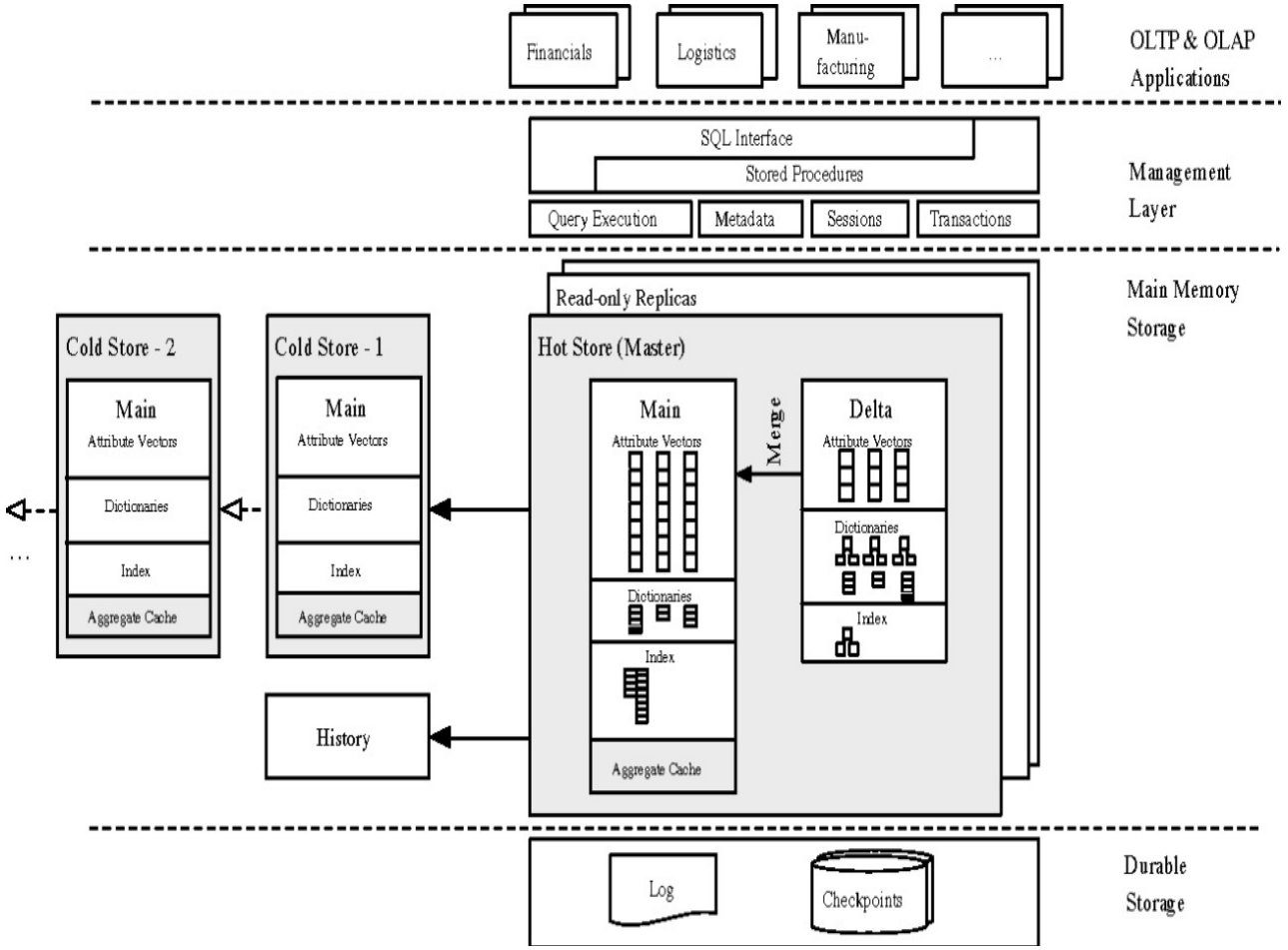

Fig -1: Architecture of In-Memory Columnar Database [3]

The above figure 1, [3] is an example of column-store database system architecture. The insert-only method involves change of code, and updates are designed as inserts and invalidate the changed line without attempting to remove it. Delete simply invalidates the removed rows, too. The tuple injection is kept in order, and the variant that was last inserted is correct. The insert-only method combined with multiple variants [10] allows the maintenance of past changes of the tables [6] and ability for retention of history and its entirety for administrative necessities. In addition, HS tables are often stored in the actual memory using an attribute along with metadata sets, with each of two divisions: the primary (main) and the delta division. The key division uses an ordered dictionary, which replaces values into an encoded format. In a write-optimized delta partition, incoming updates are collected to reduce the computation add-on of keeping the order sorted [7, 8]. Data is collected using a dictionary which is unsorted in the delta division. Additionally, a tree-based data structure is maintained per column, with all the exclusive uncondensed delta partition values [5]. Using bit-packing mechanisms the condensation of both division vectors takes places [9]. At one's discretion, the columns can be having an inverted index to allow the rapid recovery of single tuples [4].

A database table has two dimensions - rows and columns, and the intersection between them is represented by cells. The computer memory in contrast can be represented like a row with many cells, but without columns. This difference brings the question of how to store two-dimensional data in a single-dimensional form (a linear form). Figure 1 shows the answer - there are (at least) two ways to achieve this. The most common in the RDBMS world is to store table data row by row - a row-based data storage. The other way is to store data column-wise - a sequence of all cells of the first column followed by all cells of the second column etc.

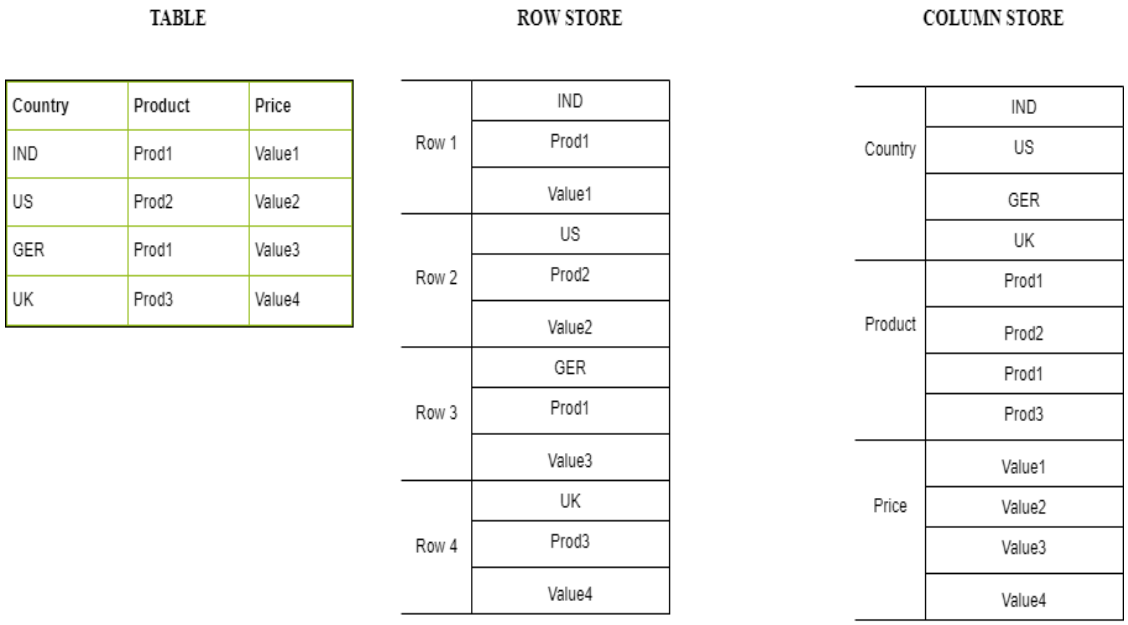

Fig -2: Example for Row Store vs Column Store

Overall performance is affected undoubtedly by type of the datastore. Depending on the type, some database operations' performance can improve, and others can decrease.

The quick sequential memory search speed of modern systems facilitates a modern enterprise system database architecture that incorporates different database techniques such as dictionary compression, columnar table layouts, and insert-only approach and multi-version competition control. Furthermore, the designed architecture of the database enables the reconstruction of applications, as easy and fast aggregations are possible and remove the overhead of maintenance of complex aggregates at the middle layer.

3. ADVANTAGES

The main advantage of column-store is ability for queries to become agile and rapid. If we need the experience average of all of our end-users, we should easily leap into the place where all "experience" values are saved and procure only values needed over actually checking the experience field for each sequentially. Instead of reading the entire rows like in traditional databases when only some fields are queried, through the columnar database, we can read only those queried columns skipping the columns we don't need, thus increasing the speed of operation and retrieving faster results when the databases are large (billions of records). The columnar database provides faster data access as only selected columns must be read. There's a higher rated condensation whilst performing algorithms for compression on columns, which will speed up the operations in the columnar databases as most hold merely a lean set of exclusive terms. In columnar databases, by default, the table is vertically portioned, which means simultaneously various columns can be operated on by assigning a processor core, thus providing better parallel processing. The columnar store table is more optimized to read the data than a row store table. Columnar databases are more suitable for analytical applications like SAP HANA. For Online analytical processing for example data warehousing which mostly involves highly complex queries overall data, columnar databases are a better option. However, data must be written in a columnar database which requires some pre-processing. On aggressive operations like AVG, COUNT, MAX, and MIN, the columnar databases provide high performance. It offers exponentially scalable features, rapid loading for variety, velocity and volumes of data along with high efficiency on condensation and partitioning. Columnar - oriented databases are efficient in accessing hard - disks. In systems like CRM that is Customer Relationship Management, Ad-hoc query systems, Business Intelligence (BI) column-oriented databases prove advantageous aiding to capabilities to process vast non-exclusive values through aggregation [1]. Business intelligence tends to concentrate on the analysis of columns – having eagle eyes on an exclusive column, operating on the values in that attribute, finding distinct values and holding an eye out for location where an update takes place. Those are some things performed better by a column-store than by a RDBMS using row-store. Some ledgers can contain more than one hundred thousand of documents, and the former database enables end-users the essential stored values, or abstract by only providing the needful data points noted for an operation. Columnar databases get results quicker because of the way the data is structured and allow for more effective data analysis.

4. LIMITATIONS

If performing a row operation, it is very fast in a row store table than a column-store table. Row store tables are more optimized to write the data than the column store table. Performing operations like update or delete data are costlier and less efficient for columnar databases are, as multiple locations on the disk, where multiple separate columns are stored, are needed for deleting or updating a single row of data. Columnar Databases are not Suitable for typical transactional applications unlike row-oriented databases like RDBMS or NoSQL. As multiple columns are read parallelly at once, there is a disk seek time is required between each block, thus increasing the disk seek time. But this can be reduced if a large disk is pre-fetched. The column-store performance depletes and thus the cost of inserts increases due to multiple distinct locations on disk which are updated for each inserted tuple (one for each attribute), but it can be reduced if inserts are done in bulk [1]. For online transaction processing workloads columnar databases are costly since INSERT transactions must be separated into columns and compressed while storing. For online transaction processing, row-oriented databases perform better - like workloads they are more heavily loaded with interactive transactions. Writing new data in column-oriented data could make more time since each column needs to be written one by one. Just a single operation is needed to insert a record into a row-oriented database. Therefore, it could cost much time updating many values or loading new data. This is why, a row-oriented database is more suited for running the backend of the web application, etc. Columnar database is suitable to be considered like Amazon Redshift or for BI analytics queries once the app becomes huge. If only a single row needs to be processed at a time in the application, then row-based store is suitable .It is also preferred in the scenario when the table has a small number of rows and neither aggregation nor fast searching is required or the application typically needs to access complete records..

5. CONCLUSION

Dealing with the variety, velocity and volume of data has become one of the game changers for ERPs and other applications. Column-oriented databases along with in-memory features were put forth to bring out a change in the way we visualize how data must be handled. Different systems have different requirements, but none deny the plethora of advantages column store databases provide. Every coin has two sides, and so column-store with its limitations. Column-oriented databases have quicker responses since they interpret only the columns required by users' questions, while row-oriented databases will interpret all rows and columns in the list. For applications that write and update a lot of data (OLTP systems), a row-oriented approach is the best solution. On the other hand, OLAP systems primarily deal with extensive dynamic querying on large volumes of data and hence find it feasible to implement Columnar databases. The market is moving towards these new innovations, in order to sustain longevity game. Only time will tell if column-store will one day, completely dominate with technological advancement over the traditional architectures.

ACKNOWLEDGEMENT

We would like to extend our gratitude towards my guide Prof. Srividya M S. for her continued support and guidance on our work in this project and paper. We would also like to show our gratitude to Department of Computer Science and Engineering, RV College of Engineering for giving us the opportunity to write this paper with complete support.

REFERENCES

[1] Vibha Shukla, Dr. Rajdev Tiwari, “Column Oriented Database: Implementation and Performance analysis” International Journal of Science and Research (IJSR), Volume 4 Issue 9, September 2015
[2] Jhonatan W. Durán-Cazar, Eduardo J. Tandazo-Gaona, Mario R. Morales-Morales, Santiago Morales Cardoso, “Performance of Columnar Database”, INGENIUS N.◦ 22, July- December 2019
[3] Hasso Plattner, Martin Faust, Stephan Müller, David Schwalb, Matthias, Uflacker, Johannes Wust, “The Impact of Columnar In-Memory Databases on Enterprise Systems”, Proceedings of the VLDB Endowment, 2014.
[4] M. Faust, D. Schwalb, J. Krüger, and H. Plattner. Fast lookups for in-memory column stores: Group-key indices, lookup and maintenance. In ADMS@VLDB, 2012.
[5] T. Karnagel, R. Dementiev, R. Rajwar, K. Lai, T. Legler, B. Schlegel, and W. Lehner. Improving in-memory database index performance with intel transactional synchronization extensions. HPCA, 2014.
[6] M. Kaufmann, P. Vagenas, P. M. Fischer,D. Kossmann, and F. Färber. Comprehensive and
interactive temporal query processing with SAP HANA. VLDB, 2013.
[7] J. Krüger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, P. Dubey, H. Plattner, and A. Zeier. Fast updates on read-optimized databases using multi-core cpus. VLDB, 2011.
[8] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O’Neil, P. O’Neil, A. Rasin, N. Tran, and S. Zdonik. C -store: A column-oriented dbms. VLDB, 2005.

[9] T. Willhalm, N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. SIMD-Scan: Ultra-Fast in-Memory Table Scan Using on-Chip Vector Processing Units. VLDB, 2009.
[10] P. A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency control and recovery in database", systems. Boston, MA, USA, 1986.