Improving performance of data analysis with **Vertical Clustered Index**

Improving performance of data analysis with **Vertical Clustered Index**

Organizations are finding it increasingly challenging to be responsive to business requirements with the growth in volume and variety of data they need to capture every year.

While databases have provided a number of solutions to address these challenges, they fall short in their ability to deliver satisfactory performance of both business transactions and analytical queries.

The reason is that the data architecture most used in IT systems has been built with online transaction processing (OLTP) in mind, which is not suitable for efficient online analytic processing (OLAP).

As a result, speed gains in one type of processing come at the expense of the other.

This is particularly problematic as we enter an era in which enterprise IT systems need to support hybrid transactional analytical processing (HTAP) workloads to help businesses make more timely decisions.

This white paper discusses how keeping both row data and columnar data can deliver significant performance gains by providing the means for fast aggregation of large data sets without impacting online business transactions.

Executive Summary

One of the driving forces behind business innovation is utilization of information to gain insight on your business and respond accordingly. This is made all the more challenging by the fact that we currently generate 10 zettabytes (trillion gigabytes) of data every year, with this number projected to increase to 180 zettabytes in 2025. [1] This means that IT systems will need to adapt, not only to keep pace with exponential data generation, but also to be able to quickly analyze larger and larger amounts of data.

On top of that, the decentralization of IT systems mean that more users have access to this data, and require different views of different subsets of it to perform their jobs.

In spite of that, organizations will need to be able to quickly aggregate gathered data for analysis, to make faster business decisions and quickly identify business opportunities in order to remain competitive.

Introduction

The demand to quickly perform data analysis is an important element for information utilization in recent years, and as result IT systems are under pressure to provide high-volume data manipulation. It is no longer acceptable to have to wait for several hours while data aggregation/manipulation processes run before the result can be used for business intelligence.

Being able to obtain real-time information quickly and often, and swiftly reflect that to the business often impacts the ability to quickly perform transactions, but as we will see in this paper, this need not be the case.

This creates the need for a single database system that can efficiently handle both large amounts of data transactions and data analysis of large amounts of data, bridging the gap between the solutions tailored for either operation.

In contrast to row-oriented data, which is best suited for OLTP, column-oriented data is best suited for data analysis. But the problem is that this architecture slows performance of data insertion/update/deletion.

Row-oriented architecture

Talking about database systems is almost synonymous with talking about row-oriented data structures, given their prevalence in database architectures. Historically, this has always been the case, and the reason for their dominance is how well they address the needs of most organizations when it comes to their bottom line - doing business.

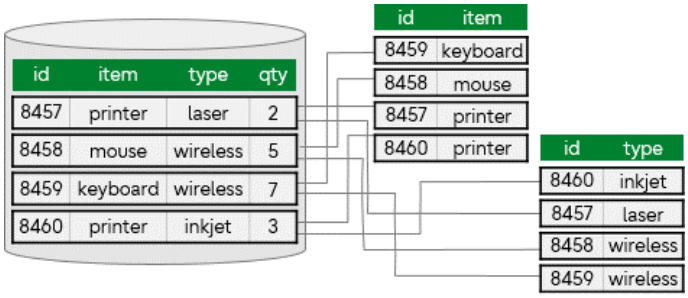
Strengths

Whether for customer data, sales history, or payment details, horizontal orientation is the most efficient way to store data on disk and to retrieve it. The reason is that the attributes (columns) will be laid out adjacent to each other, thus minimizing the amount of disk seeks for read/write, which are significant bottlenecks in time-critical and data-intensive applications. To store or retrieve all attributes of a certain customer, for example, a single disk seek will be required.

Caveats

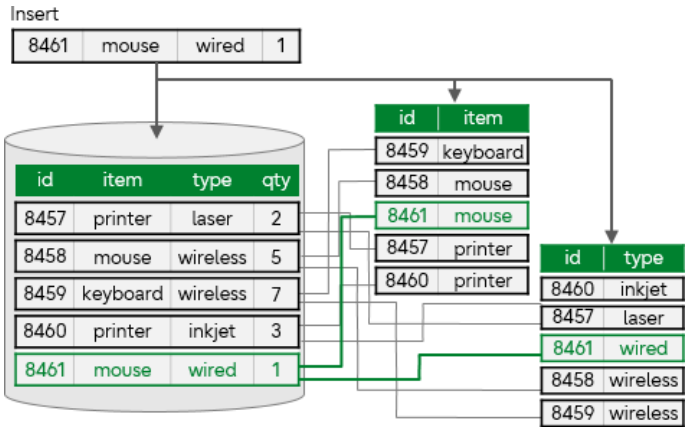
This data layout creates a problem when an application needs to access only one attribute or group of attributes of hundreds or thousands of customers. In theory, the size of the pages read would grow linearly in tandem with the number of data points accessed.

In reality, this situation is circumvented by creating an index on those attributes, which provides an alternative, smaller data set to read from.



Indexes on row-oriented data (reading data)

To ensure high performance, an index must be created for each attribute or group of attributes that is the target of aggregation. This greatly deteriorates performance, as database modifications require the ensuing update of several indexes until the modification is deemed complete.



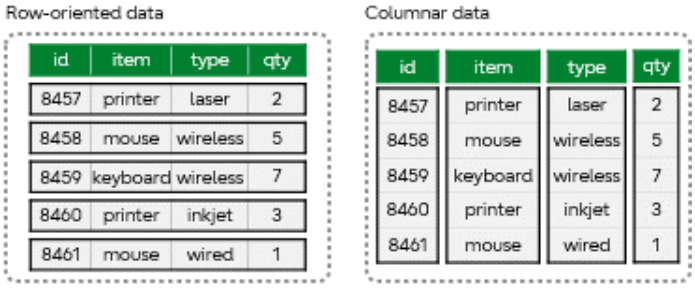
Indexes on row-oriented data (writing data)

As applications are created and modified to meet new requirements, DBAs must reassess whether indexes must be created or modified, and which ones are no longer necessary and must therefore be deleted.

Introducing column-oriented architecture

Another approach to storing data is to use vertical orientation, storing tuple attributes adjacently.

Using this architecture, scans do not read unnecessary data pertaining to attributes that are not part of the aggregation. Conversely, OLTP operations are impacted, since attributes of a tuple will be dispersed in different areas of the disk, requiring time-consuming disk seeks to access the tuple.



id	item	type	qty
8457	printer	laser	2
8458	mouse	wireless	5
8459	keyboard	wireless	7
8460	printer	inkjet	3
8461	mouse	wired	1

Column-oriented data

Having defined a columnar index for a table, it can be used in various aggregation patterns, but unlike for OLTP, it is not necessary to create an index for each aggregation pattern used - you need to specify only one indexed column in the pattern to have the index used. This reduces both the cost associated with OLTP index management and the overhead of update operations.

Another advantage of column stores is that they increase the degree of similarity between adjacent records when compared to row stores, thus allowing for better data compression. [2]

Background, outlook

Column-oriented architecture has been receiving more attention and consideration recently as an alternative or addition to row-oriented databases, due to their capacity to provide better speed for scanning and aggregating large volumes of data.

Adoption of these databases has been impacted by the fact that row-oriented data is not automatically reflected to its column-oriented counterpart and that the size of the column-oriented data is constrained by installed memory. Moreover, while it solves the issue of data analysis performance, it introduces the problem of data update overhead resulting from the resulting update of indexes.

To work around this inherent conflict, the solution is to store both row-oriented and column-oriented data, so that the system is optimized for both OLTP and OLAP workloads. [3] But most implementations of this solution do not automatically reflect changes to the row-oriented data to the column-oriented data and are affected by memory constraints.

Fujitsu Enterprise Postgres' Vertical Clustered Index

Developed by Fujitsu Laboratories Ltd, Vertical Clustered Index (VCI) is Fujitsu's proprietary implementation of In-Memory Columnar Index.

VCI uses a parallel-processing engine that does not depend on memory capacity, so it instantly updates column-oriented data in response to changes in row-oriented data, and processes column-oriented data quickly. Developers can write their applications without giving special consideration to whether the storage method is row-oriented or column-oriented.

The engine is designed for column-oriented data, and as a consequence, our tests show that for a 280 GB dataset on a 56-core Linux node, this results in almost 5 times the throughput of analytical queries while maintaining equivalent transaction volumes. Even on smaller computer systems with little memory, this technology enables real-time data analysis reflecting the latest data.

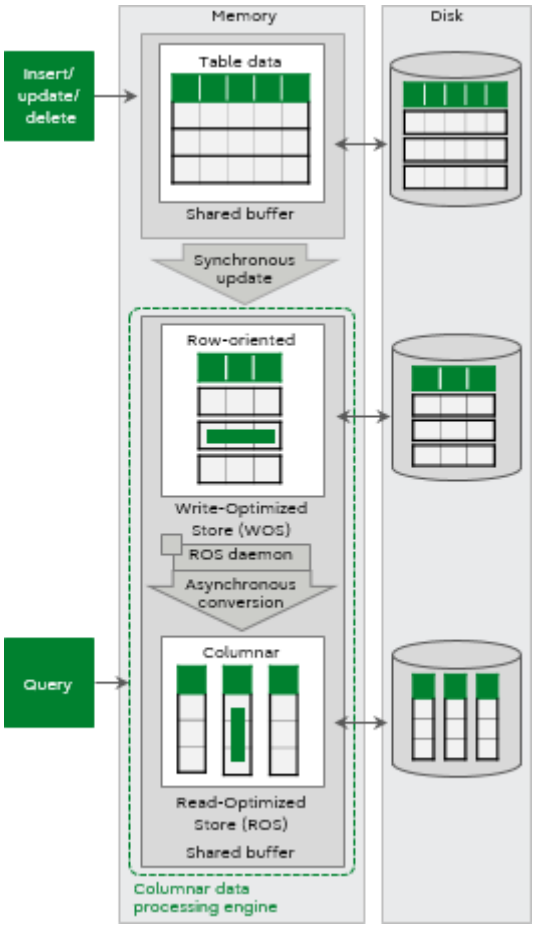
VCI data is stored in its own dedicated portion in the shared buffer value, containing row data for OLTP, and the engine ensures that consistency is maintained between them via asynchronous updates.

Maintaining data integrity

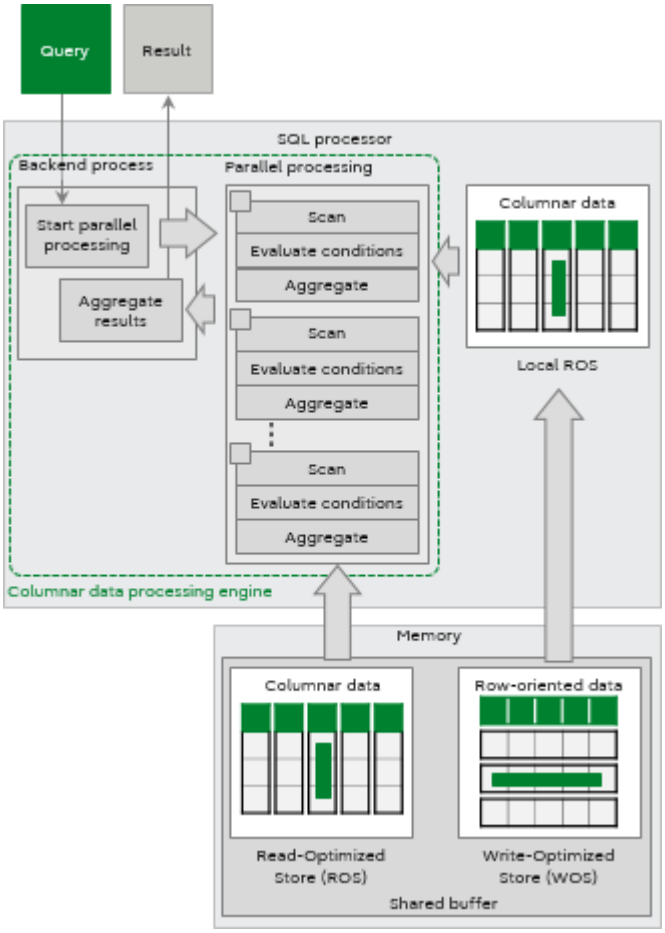
In a hybrid architecture consisting of both data orientations, row data must be converted to columnar data, which affects OLTP performance. To solve this issue, Fujitsu's Vertical Cluster Index provides two storage areas: the Write Optimized Store (WOS) for row data and the Read Optimized Store (ROS) for columnar data.

During updates, data is written only to the WOS, without data compression or indexing, in order to avoid overhead caused by conversion to columnar orientation. To further increase performance, only "record ID" and VCI-indexed columns are written to the WOS. After a certain amount of data is written to the WOS, it is asynchronously converted to columnar data and written to the ROS using compression and indexing.

By separating the synchronous process of storing OLTP data this way, the overhead of conversion to columnar data is avoided, allowing the engine to strike a balance between data synchronization and OLTP and OLAP performances.



Asynchronous conversion of WOS row data to ROS columnar data



Local ROS contains WOS data converted to columnar data

When data aggregation is performed the WOS and the ROS are combined to reflect the current table data. From row-oriented data stored in the WOS, a temporary area with columnar data is created in the SQL processor: the Local ROS. This is compared against the ROS data already converted to columnar data to judge visibility/invisibility of each record. By performing aggregation using this resulting data set, the query result is the same as if OLTP data had been used.

The columnar data processing engine performs scan, aggregation, and analysis using memory buffers to achieve improved performance. This means that immediately after a system restart the columnar data will not be available, since it will have been wiped out from memory. To resume operations, columnar data will need to be loaded again. If this information were kept only in memory, table data would need to be converted to columnar data again and then loaded. To avoid performing again this time-consuming conversion, VCI keeps columnar data on the disk as well, which can then be immediately copied to memory after the system restart.

Another benefit of also keeping columnar data on disk is that if data overflows from the buffer during operation, the engine can access disk data instead, so that performance levels are maintained without obvious deterioration.

Analysis engine optimized for columnar data

Columnar data improves read performance for aggregation, but by itself it does not make the most of the benefits that this architecture can offer.

With VCI, Fujitsu developed an analysis engine that can apply the same manipulation process simultaneously to multiple columns (vector processing), thus improving performance even more. Also as a parallel-analysis mechanism, Fujitsu developed a new shared-memory structure so that multiple processes operating in parallel can hand off data with little slowdown.

Creating a vertical index

To create columnar data for one or more attributes, create a VCI index:

```
CREATE INDEX vci_item_type
ON sale_hist
USING vci (item, type)
WITH (stable_buffer=true);
```

Having created columnar data, the application can perform aggregation in the usual way, without having to specify which index or data orientation to use - this will be left to the execution plan (more on that later).

```
SELECT item, SUM(qty)
FROM sale_hist
GROUP BY item, type;
```

The advantage of this approach is that the organization is likely to already have applications that perform aggregations using the syntax above, and those will not need to be changed to use columnar data.

Using the query planner to estimate execution time

Before performing the aggregation, the query planner calculates the execution cost and chooses the most efficient plan accordingly, determining whether the VCI will be used.

The execution plan and estimated time can be displayed:

```
EXPLAIN ANALYZE SELECT item, SUM(qty) FROM sale_hist
              GROUP BY item, type;

                  QUERY PLAN
-----------------------------------------------------
Custom Scan (VCI Aggregate)
  (cost=19403.15..19403.16 rows=991261 width=47)
  (actual time=58.505..58.506 rows=100 loops=1)
   Allocated Workers: 4
-> Custom Scan (VCI Scan) using vci_item_type on
sale_hist (cost=0.00..16925.00 rows=991261 width=47)
Planning time: 0.151 ms
Execution time: 86.910 ms
```

Conclusion

By providing data in columnar form in addition to row-oriented form, performance of OLAP operations can be drastically improved with no impact to OLTP operations. But this additional data orientation by itself is not enough to provide the best possible results - additional requirements need to be catered for, as follows.

- An efficient and reliable asynchronous synchronization method must be put in place to quickly convert row-oriented data to columnar data without affecting business transactions.
- Columnar data must be stored on disk as well as in memory, to address buffer overflows and system restarts.

Fujitsu's Vertical Clustered Index is a robust implementation of In-Memory Columnar Index that fully addresses these requirements and offers a world-class solution for data analysis of large data sets to support business intelligence.

References

1. Kanellos, Michael. (March 2016). "152,000 smart devices every minute in 2025: IDC outlines the future of smart things" (Forbes)
2. Abadi, Daniel, Samuel Madden, and Miguel Ferreira. (June 2006) "Integrating compression and execution in column-oriented database systems"
3. Arulraj, Joy, Andrew Pavlo, and Prashanth Menon. (June 2016) "Bridging the archipelago between row-stores and column-stores for hybrid workloads."

Vertical Clustered Index is an efficient and reliable asynchronous method designed to quickly convert row-oriented data to columnar data without affecting business transactions.

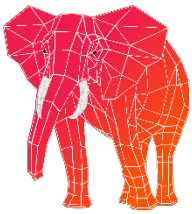
By separating the synchronous process of storing OLTP data this way, the overhead of conversion to columnar data is avoided, allowing the engine to balance data synchronisation and OLTP and OLAP performances.

As a result, what was once a primarily OLTP database can now service both operational and analytical transactions, allowing you to run HTAP workloads.

Fujitsu Enterprise Postgres can help your journey

Fujitsu Enterprise Postgres is the enhanced version of PostgreSQL, for enterprises seeking a more robust, secure, and fully supported edition for business-critical applications.

It is fully compatible with PostgreSQL and shares the same operation method, interface for application development, and inherent functionality. Designed to deliver the Quality of Service (QoS) that enterprises demand of their databases in the digital world, while supporting the openness and extensibility expected of open source platforms, all at a lower cost than traditional enterprise databases.



Fujitsu Enterprise Postgres

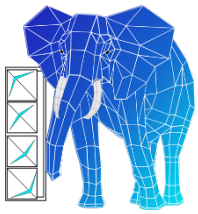
Combine the strengths of open-source PostgreSQL with the enterprise features developed by Fujitsu.

Enhanced speed, security, and support — without the costs associated with most proprietary systems.

Fujitsu Enterprise Postgres for Kubernetes

Utilize operator capabilities for provisioning and managing operations on the OpenShift Container Platform.

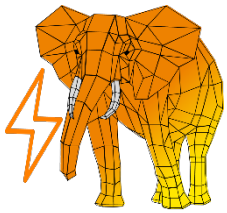
Business-ready database that integrates container operation technology for rapid development-to-production deployments.



Fujitsu Enterprise Postgres on IBM LinuxONE©

World-class platform that embraces open source and improves data security, performance, and business continuity.

The best of open source flexibility with the peace of mind that comes from knowing it is backed by Fujitsu and IBM.



Fujitsu Enterprise Postgres on IBM Power®

Experience frictionless hybrid cloud that can help you modernize to respond faster to business demands.

Fujitsu database designed for security, performance, and reliability, combined with IBM server built for agility in the hybrid cloud.

Discover how Fujitsu Enterprise Postgres' unique and enhanced features take PostgreSQL to the next level to provide enterprise-grade security, scalability, security, and performance.

Visit fast.fujitsu.com/key-features-in-fujitsu-enterprise-postgres

FUJITSU

Contact
Fujitsu Limited
Email: enterprisepostgresql@fujitsu.com
Website: fast.fujitsu.com

2025-03-26 WW EN