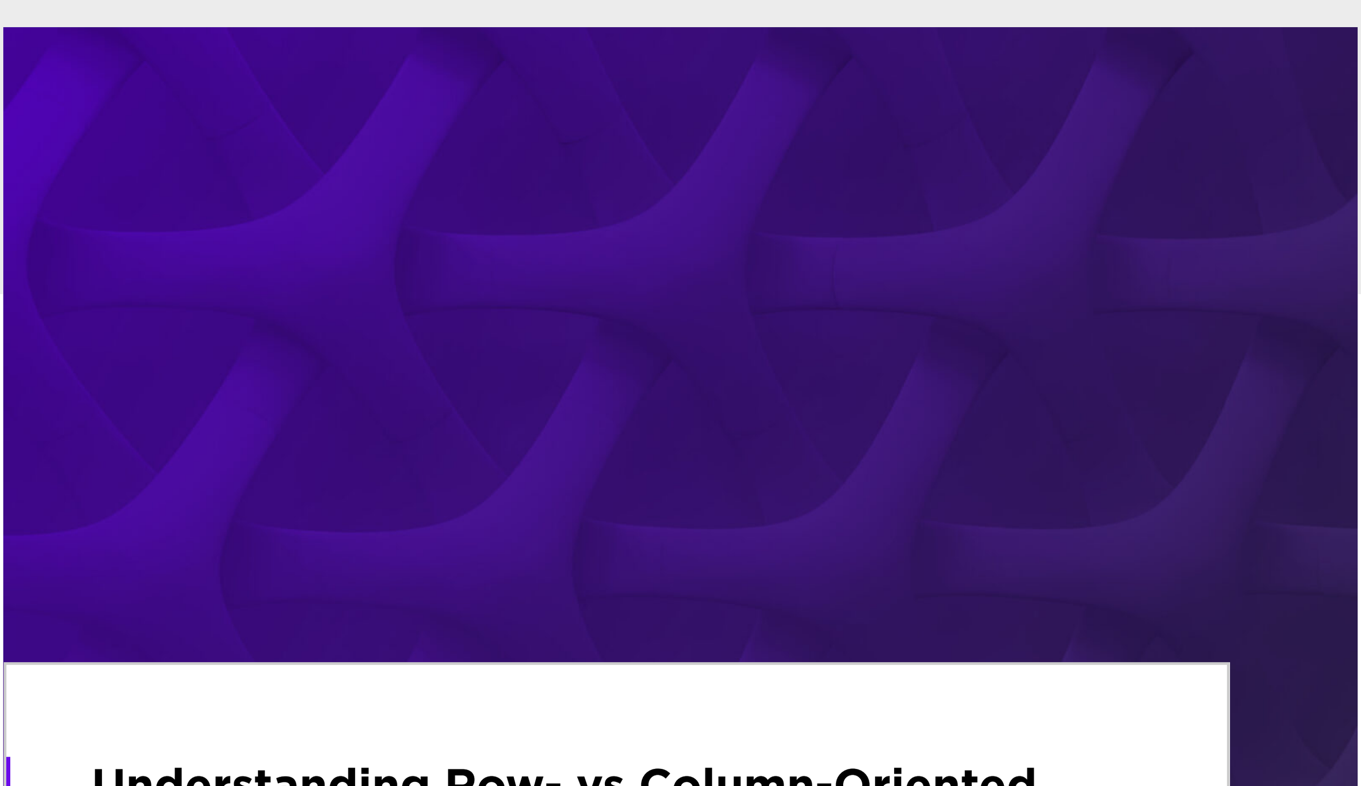

Understanding Row- vs Column-Oriented Databases

April 20, 2021
by SentinelOne

Understandably enough, we want our databases to be fast. We want to select the most appropriate database for whatever we're doing so that the queries we run will come back at speed. By changing the way we store data on computer memory, we can make certain types of queries return faster than others, which has a big impact on the database's performance. These exchanges are important because we want the types of queries we run most frequently to be as quick as possible.

Row- and column-oriented databases use different architecture methods to store data. The purpose of this post is to explain what row- and column-oriented databases are. You'll see how they work, their differences, and in which cases one is more suitable than the other.

left arrow vs down arrow signifying row vs column database

How Databases Read Data From Disk

To get a better understanding of row- and column-oriented databases, it's important to know how databases in general read data from disk. Before we mention anything else, we need to clarify that we're talking about permanent disk storage. That's data stored in a computer, even if it gets turned off at some point. In any case, it shouldn't be confused with random-access memory (RAM), which doesn't save data after you turn off your computer.

With that out of the way, let's continue.

The lowest data level of a disk is organized into blocks. Blocks are the smallest size a computer is capable of reading from a disk at one time. A database loads all necessary data after it searches the blocks that store the data it's looking for. Afterward, it will read these data blocks from the disk, which are now available to work with. A database will operate faster if there are fewer blocks it needs to read. This may be because it has to read less data and therefore fewer blocks or because the necessary data is stored in fewer blocks.

If the data your query is requesting is stored in the same blocks, the database will return it much faster than if it had to search through many different blocks. That's the core concept you need to keep in mind as we discuss whether it's best to store information on disk by rows or by columns.

row vs column database

Row-Oriented Databases

In row-oriented databases, often called traditional databases, rows are stored in sequence. This means that rows immediately follow each another. All columns in a single row are stored together on the same page as long as the row size is smaller than the page size. This provides excellent performance when querying multiple columns of a single row, as is typical in online transactional processing (OLTP) applications. PostgreSQL and MySQL are some of the most common row-oriented databases.

To make it easier to find the single row you're looking for, indexes are typically created in a row-oriented database in columns with unique keys or uncommon values like email addresses or names. However, indexes are usually not useful in analytical queries that span many rows. For example, if you have a complex query, it might lead to a sequential scan, which can negatively impact the performance.

How a Row-Oriented Database Works

It's important to have some data while we demonstrate how each method works. We're going to use a very simple table with four columns.

Name	Gender	Country	Age
John	Male	USA	63
Mary	Female	Canada	29
James	Male	Australia	48

In a row-oriented database, data is written and stored on disk one row at a time. In our example, we have John's name first, then his gender, his country, and finally his age. Then the next row will follow, and so on.

John_Male_USA_63	Mary_Female_Canada_29	James_Male_Australia_48

The data is stored in this order in different blocks, and each person's information is likely to be in the same block. So all of John's information is grouped, as is Mary's, etc. Because of that, if you run a query requesting John's data, the database won't have to load much information into memory.

Writing new data into row-oriented databases is easy and fast. We simply append the person's name at the very end of the block.

John_Male_USA_63	Mary_Female_Canada_29	James_Male_Australia_48	Kate_Female_USA_52

Column-Oriented Database

Imagine you have a table with hundreds of rows and columns. If you make a complex query, the information you need will be in several blocks, so you'll have to go through the whole database to get a result. Creating a composite index won't work because it's impossible to cover all possible combinations in an analytic environment.

row vs column database

How They Work

This is where column-oriented databases like Amazon Redshift and BigQuery come in handy. Columnar databases store data from one column together on disk. This means that all names form one group, genders form another, etc. If, for example, you need to access all names, you can do so quickly and efficiently.

Name	Gender	Country	Age
John	Male	USA	63
Mary	Female	Canada	29
James	Male	Australia	48

Because each column is stored together, you only have to read the blocks you need to get your response without having to read through unnecessary data.

When Are Column-Oriented Databases Better Than Row-Oriented Databases?

Column-oriented databases allow better compression, which is really important if you have a high volume of data. This happens because compression algorithms run more efficiently on similar data. Also, it's easy to add new columns to an existing table without having to shift all the data on the page, as is the case with row-oriented databases.

Column-oriented databases are the best solution for online analytical processing (OLAP) applications. Analytical applications aggregate lots of data from many different columns but don't require searching the entire table.

row vs column database

When Aren't Column-Oriented Databases Better?

If you need to insert a single row of data into a column-oriented database, each column of the new row has to be appended to its corresponding block. Potentially, this could lead to performance issues, especially if there are many columns in the table. Furthermore, the same difficulty occurs when you need to delete a row from your database. To successfully delete a row, you have to delete the records from all columns, which might take time if you have a large file.

Pros and Cons

In the table below, you can find a quick summary of the advantages and disadvantages of row- and column-oriented databases that we covered.

Row-oriented	Column-oriented
Inserting and deleting data is fast and easy	Inserting and deleting data could lead to a negative performance impact, especially if the table has many columns
The best solution for transactional processing (OLTP) applications	The best solution for analytical processing (OLAP) applications
Aggregating data is slow and inefficient	The best solution for aggregating data
Insufficient compression	High compression because of data similarity
Requires more space to store data	Requires less space to store data

Conclusion

Without a doubt, both row- and column-oriented databases have advantages and disadvantages. Choosing one over the other depends on each use case.

Scalyr takes a columnar approach to storing event data, which is what gives it such an edge on query and lookup without lost context. If interested, you can learn more about how Scalyr works under the hood.

Alex Doukas is the author of this post. Alex's main area of expertise is web development and everything that comes along with it. He also has extensive knowledge of topics such as UX design, big data, social media marketing, and SEO techniques.

Like this article? Follow us on LinkedIn, Twitter, YouTube or Facebook to see the content we post.

Read more about Cyber Security

- Grep an IP Address From a Log File: A Detailed How-To
- Scalyr and the Log4j Vulnerability