

Parquet Basics

03-06-2025

Welcome to the fantastic world of Parquet

What is it?

Apache Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval. It provides high performance compression and encoding schemes to handle complex data in bulk and is supported in many programming language and analytics tools¹.

The open-source project to build Apache Parquet began as a joint effort between Twitter[3] and Cloudera.[4] Parquet was designed as an improvement on the Trevni columnar storage format created by Doug Cutting, the creator of Hadoop. The first version, Apache Parquet 1.0, was released in July 2013. Since April 27, 2015, Apache Parquet has been a top-level Apache Software Foundation (ASF)-sponsored project.

Why should we care?

Apache Parquet is a file format designed to support fast data processing for complex data, with several notable characteristics:

1. **Columnar:** Unlike row-based formats such as CSV, Apache Parquet is column-oriented – meaning the values of each table column are stored next to each other, rather than those of each record:

¹<https://parquet.apache.org/docs/overview/>

ROW-BASED STORAGE	COLUMNAR STORAGE
1 MARC, JOHNSON, WASHINGTON, 27	ID: 1 2 3
2 JIM, THOMPSON, DENVER, 33	FIRST NAME: MARC, JIM, JACK
3 JACK, RILEY, SEATTLE, 51	LAST NAME: JOHNSON, THOMPSON, RILEY
	CITY: WASHINGTON, DENVER, SEATTLE
	AGE: 27 33 51

The key difference between a CSV and Parquet file format is how each one is organized. A Parquet file format is structured by row, with every separate column independently accessible from the rest. Since the data in each column is expected to be of the same type, the parquet file format makes encoding, compressing and optimizing data storage possible.

2. **Open-source:** Parquet is free to use and open source under the Apache Hadoop license.

Apache Parquet is a columnar storage format available to any project [...], regardless of the choice of data processing framework, data model or programming language².

3. **Self-describing:** In addition to data, a Parquet file contains metadata including schema and structure. Each file stores both the data and the standards used for accessing each record – making it easier to decouple services that write, store, and read Parquet files.
4. **Binary format:** Parquet file formats store data in binary format, which reduces the overhead of textual representation. It's important to note that Parquet files are not stored in plain text, thus cannot be opened in a text editor.

Advantages of Parquet Columnar Storage – Why Should You Use It?

The above characteristics of the Apache Parquet file format create several distinct benefits when it comes to storing and analysing large volumes of data.

Compression

File compression is the act of taking a file and making it smaller. In Parquet, compression is performed column by column and it is built to support flexible compression options and extendable encoding schemas per data type – e.g., different encoding can be used for compressing integer and string data.

²<https://parquet.apache.org/>

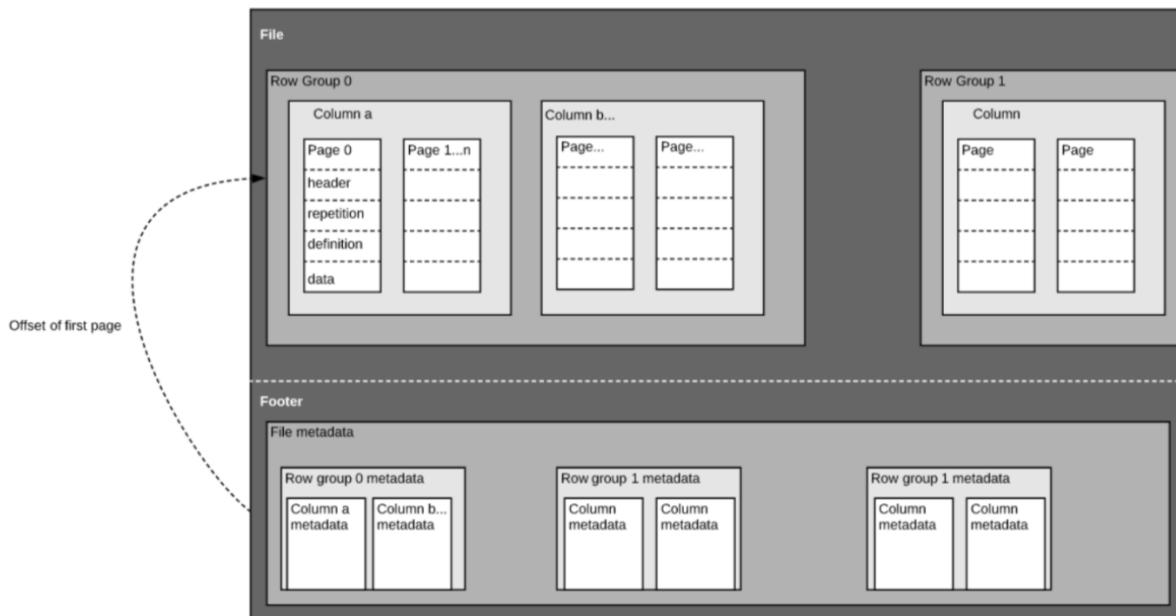
Parquet data can be compressed using these encoding methods:

- **Dictionary encoding:** this is enabled automatically and dynamically for data with a small number of unique values.
- **Bit packing:** Storage of integers is usually done with dedicated 32 or 64 bits per integer. This allows more efficient storage of small integers.
- **Run length encoding (RLE):** when the same value occurs multiple times, a single value is stored once along with the number of occurrences. Parquet implements a combined version of bit packing and RLE, in which the encoding switches based on which produces the best compression results.

Performance

As opposed to row-based file formats like CSV, Parquet is optimized for performance. When running queries on your Parquet-based file-system, you can focus only on the relevant data very quickly. Moreover, the amount of data scanned will be way smaller and will result in less I/O usage. To understand this, let's look a bit deeper into how Parquet files are structured.

As we mentioned above, Parquet is a self-described format, so each file contains both data and metadata. Parquet files are composed of row groups, header and footer. Each row group contains data from the same columns. The same columns are stored together in each row group:



This structure is well-optimized both for fast query performance, as well as low I/O (minimizing the amount of data scanned). For example, if you have a table with 1000 columns, which you will usually only query using a small subset of columns. Using Parquet files will enable

you to fetch only the required columns and their values, load those in memory and answer the query. If a row-based file format like CSV was used, the entire table would have to have been loaded in memory, resulting in increased I/O and worse performance.

Schema evolution

When using columnar file formats like Parquet, users can start with a simple schema, and gradually add more columns to the schema as needed. In this way, users may end up with multiple Parquet files with different but mutually compatible schemas. In these cases, Parquet supports automatic schema merging among these files.

Column-Oriented vs Row-Based Storage for Analytic Querying

Data is often generated and more easily conceptualized in rows. We are used to thinking in terms of Excel spreadsheets, where we can see all the data relevant to a specific record in one neat and organized row. However, for large-scale analytical querying, columnar storage comes with significant advantages with regards to cost and performance.

Complex data such as logs and event streams would need to be represented as a table with hundreds or thousands of columns, and many millions of rows. Storing this table in a row based format such as CSV would mean:

- Queries will take longer to run since more data needs to be scanned, rather than only querying the subset of columns we need to answer a query (which typically requires aggregating based on dimension or category)
- Storage will be more costly since CSVs are not compressed as efficiently as Parquet

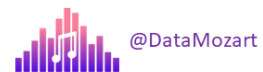
Columnar formats provide better compression and improved performance out-of-the-box, and enable you to query data vertically – column by column.

	Product	Customer	Country	Date	Sales Amount
Row 1	Ball	John Doe	USA	2023-01-01	100
Row 2	T-Shirt	John Doe	USA	2023-01-02	200
Row 3	Socks	Maria Adams	UK	2023-01-01	300
Row 4	Socks	Antonio Grant	USA	2023-01-03	100
Row 5	T-Shirt	Maria Adams	UK	2023-01-02	500
Row 6	Socks	John Doe	USA	2023-01-05	200



images columns row³

	Product	Customer	Country	Date	Sales Amount
Row 1	Ball	John Doe	USA	2023-01-01	100
Row 2	T-Shirt	John Doe	USA	2023-01-02	200
Row 3	Socks	Maria Adams	UK	2023-01-01	300
Row 4	Socks	Antonio Grant	USA	2023-01-03	100
Row 5	T-Shirt	Maria Adams	UK	2023-01-02	500
Row 6	Socks	John Doe	USA	2023-01-05	200



³<https://data-mozart.com/parquet-file-format-everything-you-need-to-know/>

Column 1	Column 2	Column 3	Column 4	Column 5
Product	Customer	Country	Date	Sales Amount
Ball	John Doe	USA	2023-01-01	100
T-Shirt	John Doe	USA	2023-01-02	200
Socks	Maria Adams	UK	2023-01-01	300
Socks	Antonio Grant	USA	2023-01-03	100
T-Shirt	Maria Adams	UK	2023-01-02	500
Socks	John Doe	USA	2023-01-05	200

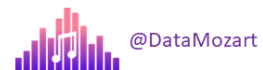


	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	Country	Date	Sales Amount
Row Group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row Group 2	Socks	Maria Adams	UK	2023-01-01	300
	Socks	Antonio Grant	USA	2023-01-03	100
Row Group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200



	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	Country	Date	Sales Amount
Row Group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row Group 2	Socks	Maria Adams	UK	2023-01-01	300
	Socks	Maria Adams	UK	2023-01-01	100
Row Group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200

The engine will not scan these records



Let's quickly stop here, as I want you to realize the difference between various types of storage in terms of the work that needs to be performed by the engine:

Row store - the engine needs to scan all 5 columns and all 6 rows

Column store - the engine needs to scan 2 columns and all 6 rows

Column store with row groups - the engine needs to scan 2 columns and 4 rows

I'm tired of reading non-sense

Ok, then let me introduce you to the R packages I've been exploring and how they made my life easier.

arrow

<https://arrow.apache.org/docs/r/>

example from real life with ebird pts?

geoparquet - sfarrow

example from real life with rangemaps?

tidyverse - parquet

<https://www.tidyverse.org/blog/2025/01/nanoparquet-0-4-0/>