Announcing Vortex Support in DuckDB

Guillermo Sanchez, SpiralDB Team
2026-01-23 · 8 min

TL;DR: Vortex is a new columnar file format with a very promising design. SpiralDB and DuckDB Labs have partnered to give you a very fast experience while reading and writing Vortex files!

I think it is worth starting this intro by talking a little bit about the established format for columnar data. Parquet has done some amazing things for analytics. If you go back to the times where CSV was the better alternative, then you know how important Parquet is. However, even if the specification has evolved over time, Parquet has some design constraints. A particular limitation is that it is block-compressed and engines need to decompress pages in order to do further operations like filtering, decoding values, etc. For a while, researchers and private companies have been working on alternatives to Parquet that could improve on some of Parquet's shortcomings. Vortex, from the SpiralDB team, is one of them.

What is Vortex?

Vortex is an extensible, open source format for columnar data. It was created to handle heterogeneous compute patterns and different data modalities. But, what does this mean?

The project was donated to the Linux Foundation by the SpiralDB team in August 2025.

Vortex provides different layouts and encodings for different data types. Some of the most notorious are ALP for floating point encoding or FSST for string encoding. This lightweight compression strategy keeps data sizes down while allowing one of Vortex's most important features: compute functions. By knowing the encoded layout of the data, Vortex is able to run arbitrary expressions on compressed data. This allows a Vortex reader to execute, for example, filter expressions within storage segments without decompressing data.

We mentioned heterogeneous compute to emphasize that Vortex was designed with the idea of having optimized layouts for different data types, including vectors, large text or even image or audio, but also to maximize CPU or GPU saturation. The idea is that decompression is deferred all the way to the GPU or CPU, enabling what Vortex calls "late materialization". The FastLanes encoding, a project originating at CWI (like DuckDB), is one of the main drivers behind this feature.

Vortex also supports dynamically loaded libraries (similar to DuckDB extensions) to provide new encodings for specific types as well as specific compute functions, e.g. for geospatial data. Another very interesting feature is encoding WebAssembly into the file, which can allow the reader to benefit from specific compute kernels applied to the file.

Besides DuckDB, other engines such as DataFusion, Spark and Arrow already offer integration with Vortex.

For more information, check out the Vortex documentation .

The DuckDB Vortex Extension

DuckDB is a database as the name says, yes, but it is also widely used as an engine to query many different data sources. Through core or community extensions, DuckDB can integrate with:

- Databases like Snowflake, BigQuery or PostgreSQL.
- Lakehouse formats like Delta, Iceberg or DuckLake.
- File formats, most notably JSON, CSV, Parquet and most recently Vortex.

> The community has gotten very creative, though, so these days you can even read YAML and Markdown with DuckDB using community extensions.

All this is possible due to the DuckDB extension system, which makes it relatively easy to implement logic to interact with different file formats or external systems.

The SpiralDB team built a DuckDB extension . Together with the DuckDB Labs team, we have made the extension available as a core DuckDB extension, so that the community can enjoy Vortex as a first-class citizen in DuckDB.

Example Usage

Installing and using the Vortex extension is very simple:

```
INSTALL vortex;
LOAD vortex;
```

Then, you can easily use it to read and write, similar to other extensions such as Parquet.

```
SELECT * FROM read_vortex('my.vortex');
```

```
COPY (SELECT * FROM generate_series(0, 3) t(i))
TO 'my.vortex' (FORMAT vortex);
```

Why Vortex and DuckDB?

Vortex claims to do well primarily at three use cases:

- Traditional SQL analytics: Through late decompression and compute expressions on compressed data, Vortex can filter down data within the storage segment, reducing IO and memory consumption.
- Machine learning pre-processing pipelines: By supporting a wide variety of encodings for different data types, Vortex claims to be effective at reading and writing data, whether it is audio, text, images or vectors.
- AI model training: Encodings such as FastLanes allow for a very efficient copy of data to the GPU. Vortex is aiming at being able to copy data directly from S3 object storage to the GPU.

The promise of more efficient IO and memory use through late decompression is a good reason to try DuckDB and Vortex for SQL analytics. On another note, if you are looking at running analytics on unified datasets that are used for multiple use cases, including pre-processing pipelines and AI training, then Vortex may be a good candidate since it is designed to fit all of these use cases well.

Performance Experiment

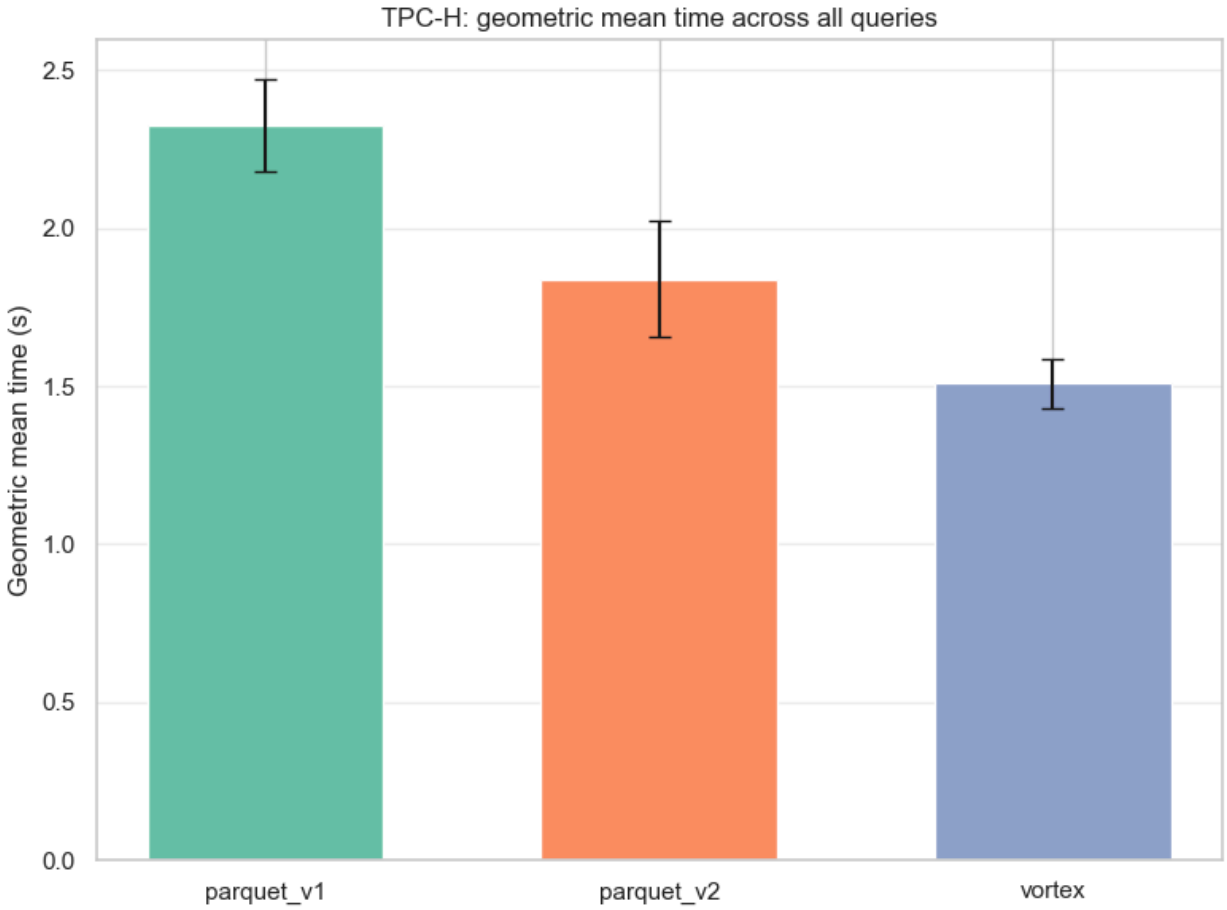
For those who are number hungry, we decided to run a TPC-H benchmark scale factor 100 with DuckDB to understand how Vortex can perform as a storage format compared to Parquet. We tried to make the benchmark as fair as possible. These are the parameters:

- Run on Mac M1 with 10 cores & 32 GB of memory.

- The benchmark runs each query 5 times and the average is used for the final report.
- The DuckDB connection is closed after each query to try to make runs “colder” and avoid DuckDB's caching (particularly with Parquet) from influencing the results. OS page caching does have an influence in subsequent runs but we decided to acknowledge this factor and still keep the first run.
- Each TPC-H table is a single file, which means that lineitem files for Parquet and Vortex are quite large (both around 20 GB). This allows us to ignore the effect of globbing and having many small files.
- Data files used for the benchmark are generated with tpchgen-rs and are copied out using DuckDB’s Parquet and Vortex extensions.
- We compared Vortex against Parquet v1 and v2. The v2 specification allows for considerably faster reading than the v1 specification but many writers do not support this, so we thought it was worth including both.

The results are very good. The TPC-H benchmark runs 18% faster with respect to Parquet V2 and 35% faster than Parquet V1 (using the geometric means, which is the recommended approach).

Another interesting result is the standard deviation across runs. There was a considerable difference between the first (and coldest) run of each query and subsequent runs in Parquet, while Vortex performed very well across all runs with a much smaller standard deviation.



Format	Geometric Mean (s)	Arithmetic Mean (s)	Avg Std Dev (s)	Total Time (s)
parquet_v1	2.324712	2.875722	0.145914	63.265881
parquet_v2	1.839171	2.288013	0.182962	50.336281
vortex	1.507675	1.991289	0.078893	43.808349

The times did vary across different runs of the same benchmark, and subsequent runs have yielded similar results but with slight variations. The differences between Parquet v2 and Vortex have always been around 12-18% in geometric means and around 8-14% in total times. Benchmarking is very hard!

Click here to see a more detailed breakdown of the benchmark results.

Conclusion

Vortex is a very interesting alternative to established columnar formats like Parquet. Its focus on lightweight compression encodings, late decompression and being able to run compute expressions on compressed data makes it very interesting for a wide range of use cases. With regard to DuckDB, we see that Vortex is already very performant for analytical queries, where it is on par or better than Parquet v2 on the TPC-H benchmark queries.

> Vortex has been backwards compatible since version 0.36.0, which was released more than 6 months ago. Vortex is now at version 0.56.0.