LanceDB - Embracing Composability in the Storage Layer

Weston Pace
LanceDB
weston@lancedb.com

Chang She
LanceDB
chang@lancedb.com

Lei Xu
LanceDB
lei@lancedb.com

Will Jones
LanceDB
will@lancedb.com

Rob Meng
LanceDB
rob@lancedb.com

Yang Cen
LanceDB
yang@lancedb.com

ABSTRACT

Apache Arrow has become the industry standard for in-memory and over-the-wire tabular representation, yielding an ecosystem of powerful components with clean interfaces, ready to be used as building blocks in composable data systems[2]. Unfortunately, the same cannot be said for the storage layer, as today's data management systems are often tightly coupled to the underlying storage format (e.g. Parquet). As new AI workloads emerge over the same cloud primitives (object storage, ephemeral compute), innovation is required to overcome these storage limitations.

Starting from the storage layer, we built a modular system tailored for a new class of multi-modal, retrieval-based workloads. By using a small set of core technologies, such as Arrow and Substrait, we allow these components to be used with limited coupling while introducing only minor trade-offs, which we will discuss. We summarize our contributions as follows:

(1) We introduce a new storage format, the Lance file format[1], focused on better random access performance and memory utilization compared to Parquet. By using Arrow for our type system and Substrait to describe filter expressions, this format is able to be readily adopted within any composable system supporting Arrow.
(2) We introduce the Lance table format, designed to support AI workloads by offering both search and scan workloads. In addition, expensive copies of multimodal data are avoided through the use of secondary indices and two dimensional storage. The format's interface is an Arrow based interface modeled after the Pyarrow datasets API. This allows reuse without requiring a hard dependency on implementation details such as the manifest design.
(3) We introduce LanceDB, a database frontend loosely modeled after the Pyarrow datasets API, but adding support for search as well as scan. The frontend is based on three key interfaces (table, database, catalog) which are defined via Arrow and Substrait. As a result, the same user API can be used for native tables, remote tables, and presumably even tables backed by radically different technologies.

The Lance format, table, and database frontend would not have been possible outside of the composable data system community. In turn, embracing this very philosophy, we are now in a position to give back to the community new modular components up and down the data stack, in the hope they will foster another iteration of innovation. This effort has also highlighted gaps in the ecosystem which could be addressed by future work. Tools aside, we believe our experience in building with and building for other systems is valuable to a broad audience of data practitioners.

VLDB Workshop Reference Format:
Weston Pace, Chang She, Lei Xu, Will Jones, Rob Meng, and Yang Cen. LanceDB - Embracing Composability in the Storage Layer. VLDB 2025 Workshop: Third International Workshop on Composable Data Management Systems.

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/lancedb/lancedb.

REFERENCES

[1] Weston Pace, Chang She, Lei Xu, Will Jones, Albert Lockett, Jun Wang, and Raunak Shah. 2025. Lance: Efficient Random Access in Columnar Storage through Adaptive Structural Encodings. arXiv:2504.15247 [cs.DB] https://arxiv.org/abs/2504.15247
[2] Jacopo Tagliabue, Ciro Greco, and Luca Bigon. 2023. Building a serverless Data Lakehouse from spare parts. In *Second International Workshop on Composable Data Management Systems.* arXiv:2308.05368 [cs.DB] https://arxiv.org/abs/2308.05368

Proceedings of the VLDB Endowment. ISSN 2150-8097.