Adoption of Modular Data Tooling Brings Flexibility

Standardisation around Apache Arrow, Substrait and Ibis are potentially transformative.

Version: 0.2 Dated: 2024-03-08 Authored by: CM

colum.mccoole@analect.com
colum-mccoole-746b946a
analect/technical-docs-hierarchy

Docs: > Level 2 > AI/ML Value-chain

Topic Navigation

How does this topic relate to other relevant content in this space.

What Problem Are We Solving For?

Being able to quickly prototype with data (either from the core business or that of portfolio companies), build proof-of-concepts and then shift those solutions to production, without the need to re-write code brings with it great agility. With greater standardisation around protocols like Arrow, it becomes easier to assemble a modular data ecosystem for analytics.

Data Serialization Bottleneck

While compute and networking speeds have soared over the past decade, data-analytics has lagged badly - the bottleneck often being CPU-bound, given a typical, up to now, need to serialize and de-serialize data moving between different ecosystem layers.

Without a standardized solution for data interchange and in-memory computation, systems pay a steep penalty both in computational cost and development time to interoperate with each other.

A Better Way

Voltron Data (the driving force behind the open-source Ibis project) have assembled a well-articulated technical series called The Composable Codex which brings a useful historical context to what has needed to happen to unleash better productivity around the data ecosystem, all enabled by open standards for exchanging and operating on data, allowing for more composable data systems that are **modular**, **interoperable**, **customizable**, and **extensible** (MICE).

1: AI is Eating The World - https://txt.cohere.com/ai-is-eating-the-world/ 2: The Composable Codex - Voltron Data - https://voltrondata.com/codex

Standards-based Data Tooling

Together these projects are working to enable modularity in data interchange, query execution, and programming interfaces.

1. **Apache Arrow** (2015): is a cross-language development platform for in-memory data. It specifies a standardized language-independent columnar memory format for flat and hierarchical data, organized for efficient analytic operations on modern hardware.
2. **Arrow Flight**: provides a high-performance wire protocol for large-volume data transfer for analytics.
3. **Substrait** (2021): a language-independent intermediate representation (IR) middleware for analytical computing to assist in decoupling user interfaces from compute engines.
4. **Ibis** (2014) and **dplyr** (2012): backend-agnostic data frame interfaces in Python and R respectively - to enable systems to engage with each other independent of their own native SQL dialect or other query languages.
5. **RAPIDS** (2018): GPU-accelerated libraries for data analytics and machine learning. Built on NVIDIA CUDA and Apache Arrow, it unlocks the speed of GPUs with code you already know.
6. **DuckDB** (2018) and **Velox** (2021): embeddable systems providing fast columnar query processing. DuckDB is an on-disk database designed to support analytical query workloads and has no external dependencies.

Data Can be the Differentiator

Cohere, one of the pre-eminent players in LLMs, made some recent interesting observations [1] including the fact that:

- **Models derive their value from the data they're trained on**. So, there's a need to factor in data and machine learning operations (MLOps) as a layer supporting the models.
- One of the central developments in AI is that we now have pre-trained foundation models that are great at a large number of tasks (say, language tasks), which can then be trained a little bit more (a process known as **fine-tuning**) on a much smaller dataset to become excellent at one task.
- Fine-tuning matters to the economic value map because it allows businesses to build **proprietary custom models**, even if the original model was publicly accessible or even open source.
- Generation, Usage, and Feedback Data are Valuable for Future Versions of the Model - enabling RLHF (**Reinforcement Learning from Human Feedback**).

These all point to the importance of handling data (and its quality) in equal standing with model-training and ultimately that the proprietary nature of the data can usurp the importance of best-in-class model-training capabilities.

Composable Data-Tooling Ecosystem

Referencing Figure 1, The Composable Codex [2] has a concept of *Layers* (black boxes below) and *Standards*.

User Interface (1) - Users interact with this in order to initiate operations on data. This is typically exposed as a language frontend or API. **Execution Engine (2)** - this performs operations on the data, as specified by users. **Data Storage (3)** - the layer that stores data that is available to users.

Substrait (A) - is a format standard for describing compute operations on structured data. **ADBC (Arrow Database Connectivity) (B)** - applications code to this API standard (like JDBC or ODBC), but fetch result sets in Arrow format. **Arrow (C)** - is focused on a standardized memory representation of columnar data.

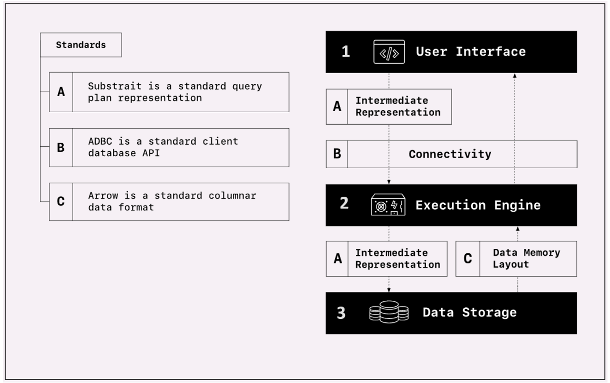
Standards
A
Substrait is a standard query plan representation
B
ADBC is a standard client database API
C
Arrow is a standard columnar data format
1
User Interface
A
Intermediate Representation
B
Connectivity
2
Execution Engine
A
Intermediate Representation
C
Data Memory Layout
3
Data Storage

Figure 1. Codex Composable Standards: Substrait, ADBC and Arrow - https://voltrondata.com/codex/standards-over-silos

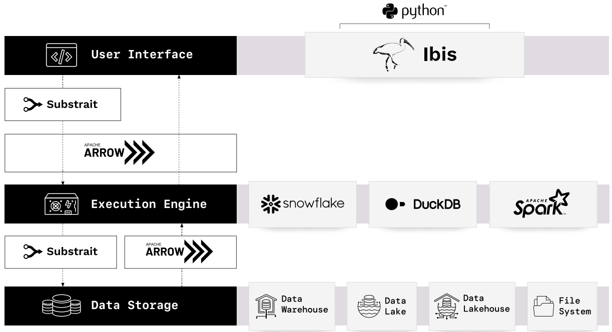

Figure 2. Example Data Tooling mapping to these standards