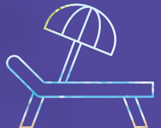
Substrait

Rethinking DBMS Composability

substrait.io



Who?

OSS

- Substrait: Co-creator
- Apache Arrow: Co-creator, Founding PMC Chair
- Apache Calcite: Founding PMC member
- Apache Drill: Co-creator, Founding PMC Chair

Commercial

- Sundeck: CEO & Co-founder
- Dremio: CTO & Co-founder

The world has changed

Data lock-in is disappearing

Cloud Storage as System-of-Record

amazon S3

Microsoft Azure Blob Storage

Google Cloud Storage

Rise of Table Formats

ICEBERG

Apache hudi

DELTA LAKE

The coupling of API and compute is breaking down

Generic Compute Kernels

databricks Photon

Velox

VOLTRON DATA

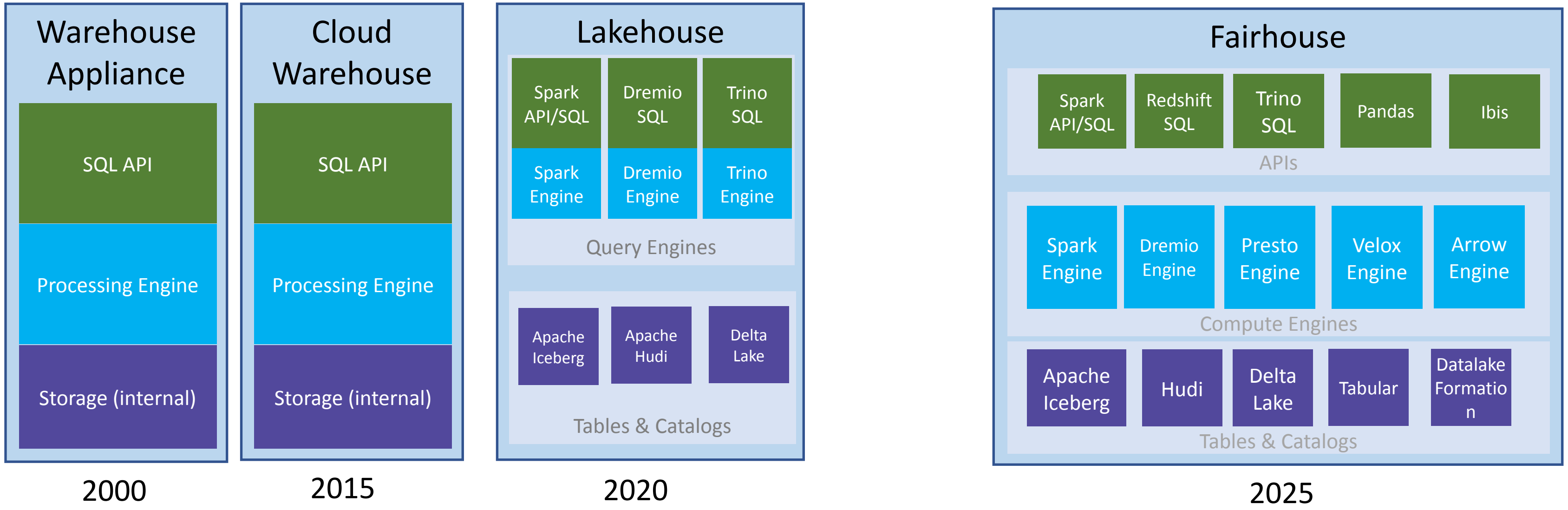
Compute Engine Specialization

pinot

druid

ClickHouse

Warehouse, Lakehouse, soon we'll see the Fairhouse



- Specialized hardware
- Designed specifically for analytical use

Teradata, Netezza

- Built on cloud storage
- Unlimited Scale
-

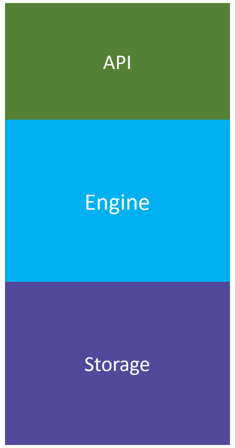
Snowflake, Redshift

- Shared data ownership
- API choice independent of storage format
- Reduction in data gravity

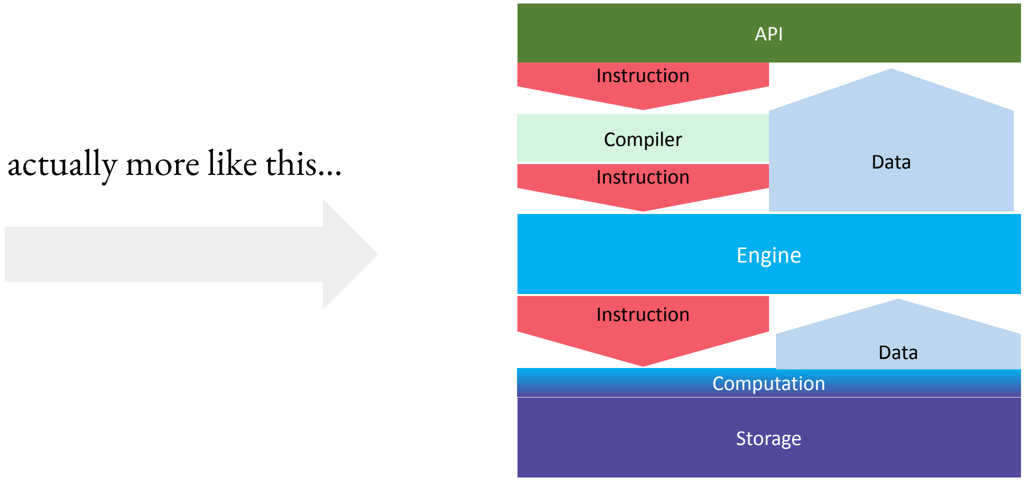
Databricks, Dremio, Starburst

- Rise of the generic engine
- Engine options independent of api choice

Best-of-breed Decomposition Requires Components



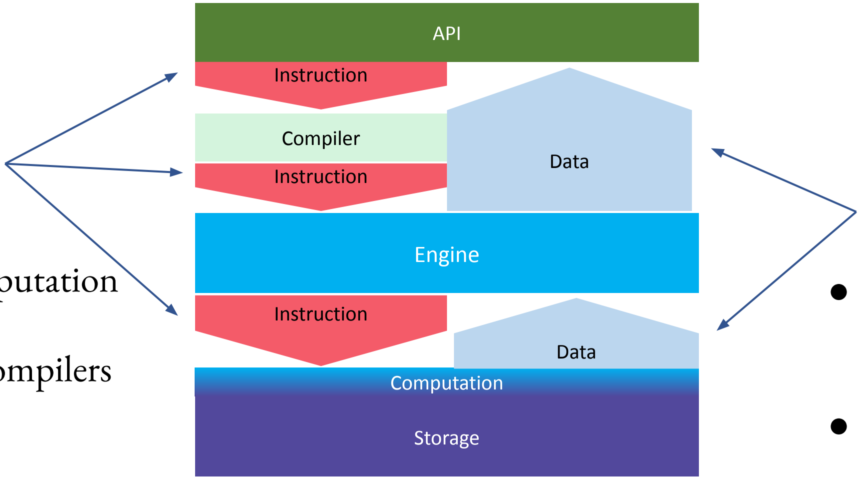
actually more like this…



How to collaborate on these layers?

Substrait

- API independent computation definition
- Engine independent compilers
- Engine independent computational storage



APACHE ARROW

- High performance independent in-memory format
- In-engine optimized wire-friendly representation

Inspiration

Abstract Need: Drive Innovation

	JVM Bytecode	LLVM IR
FE Innovations	Scala, Clojure, Kotlin	Rust, Swift
BE Innovations	Dalvik, Graal	WASM

Concrete Need: Solve Real Problems

- Iceberg: Need for a common representation of views
- Arrow: Common representation of compute plans across engines
- Calcite: expose functionality to non-jvm environments

Substrait: Cross-Language Serialization for Relational Algebra

Status

- Formed September 2021
- Several integrations ongoing, 30+ contributors from multiple companies

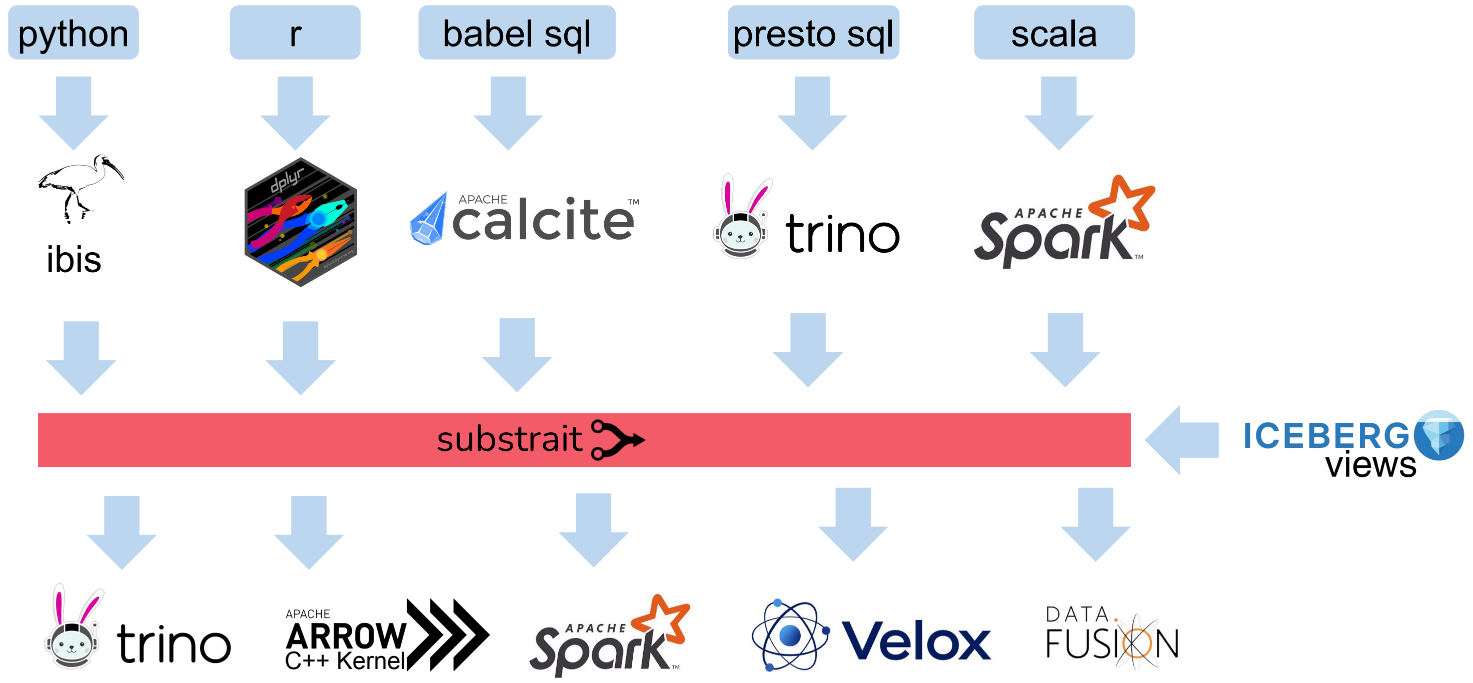
Purpose

- Create a well-defined, cross-language specification for data compute operations

Why

- New kernels/engines should work with existing analysis experiences
- It should be easy to create new computation design languages/platforms
- Innovation is stifled when each new data system needs to solve all FE and BE problems

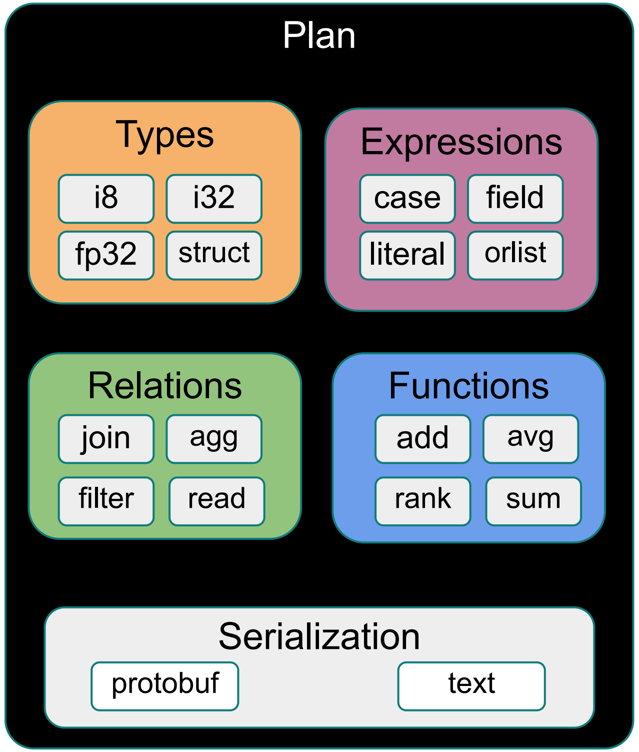
Theoretical Integrations



Core Principles

- Specification-first
- Language independent & serializable
- Plans are self-contained, have clear intention
    - Allow for dumb consumers
- Structured hierarchy of primitives
- Common primitive definitions within project
    - Common types, functions, relational operators
- Extensibility with discipline

Substrait Primitives



- Types
  - Simple (e.g. i32, fp32, string)
  - Compound (e.g. varchar<N>, fixedbinary<N>)
  - Complex (e.g. List<E>, Map<K,V>, Struct<T,U,...Z>)
- Expressions
  - Switch statements
  - Field selection (simple and complex)
  - Literals
- Functions
  - Scalar
  - Aggregate
  - Window
  - Table
- Relations
  - Production
  - Consumption
  - Distribution
  - Transformation
- Plans
  - Splittable
  - Normalized for space efficiency
- Serialization
  - Binary (currently protobuf)
  - Text (tbd, likely yaml)

Extensibility with Discipline

- Project inclusive of patterns that show up in most projects
  - int32, decimal, add, subtract, aggregate, join, hash join, etc.
- Specification defined extensibility
  - Separation between optimization and semantic differences
  - Well-defined ways to sync independent systems around extensions
- Types
  - Support for physical variations of existing types (row-wise vs columnar, rle or not, etc)
  - Declare custom types in YAML and use in functions, expressions, etc.
- Functions
  - Declare custom functions via YAML, standard referencing scheme
  - User-defined functions (write once, use many times)
  - Embedded (business logic closure in scala, python, llvm, webassembly)
- Relations
  - Extend existing relations for execution optimization information
  - Declare new relations via serialization extensions (such as protobuf
  - User-defined and Embedded patterns

Project Governance

Guiding Principles

- Consensus-driven project
- All collaboration and decision-making is done in the public
- Avoid control by any particular organization or person
- Users of Substrait should be confident that the project won't one day "change its stripes" like Redis, Confluent or various other projects did.

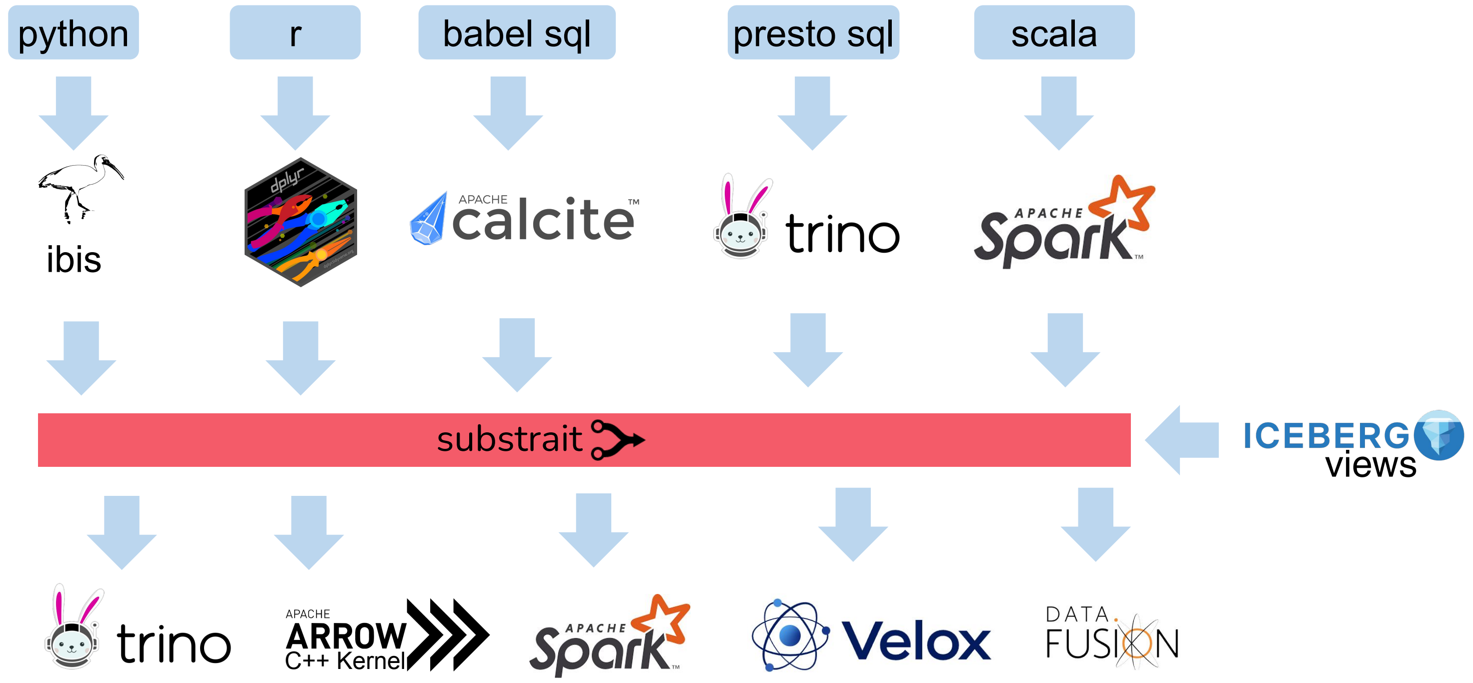
Details

- Apache 2.0 Licensed
- Github Project
- Active contributions from several companies
- Move to a foundation if contributors so prefer

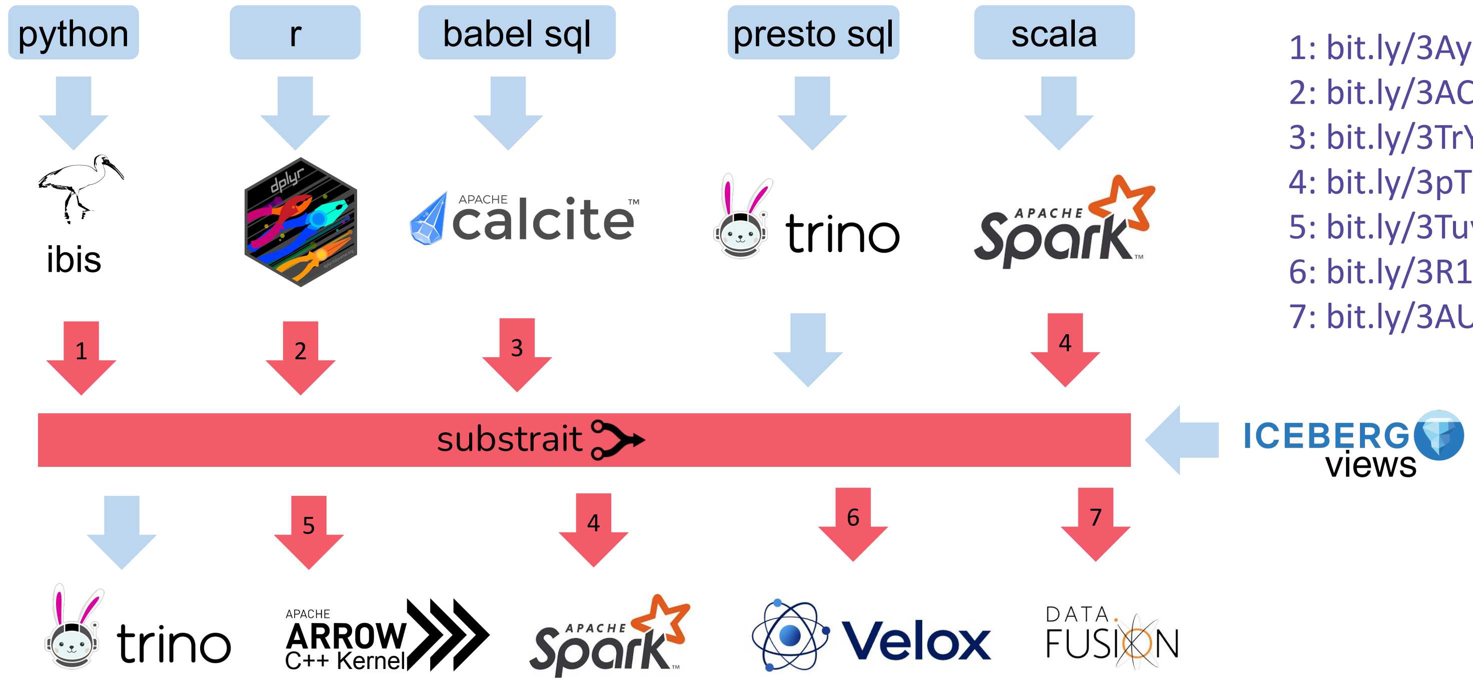
Core Components

- Specification/Format
    - github.com/substrait-io/substrait
- Language specific helper libraries (Java, C++, C#, Go, Rust)
    - github.com/substrait-io/substrait-*
- Plan Validator
    - github.com/substrait-io/substrait-validator
- Integration Tests
    - github.com/substrait-io/consumer-testing
- Network Protocol (On top of Arrow Flight SQL)
    - github.com/apache/arrow/pull/13492
- Integations
    - Next page...

Theoretical Integrations



Actual Implementations



1: bit.ly/3AyLsP9
2: bit.ly/3AC6jku
3: bit.ly/3TrY3wh
4: bit.ly/3pTi2q3
5: bit.ly/3TuySck
6: bit.ly/3R1CFwl
7: bit.ly/3AUNH0G

Join the Community

- Start using Substrait
- Join the community
- Share your feedback

- Substrait Slack
  - Subscribe
- Github
  - Start a discussion
  - Open an issue
  - Submit a PR