FROM FLAT FILES TO DECONSTRUCTED DATABASE

The evolution and future of the Big Data ecosystem.

Julien Le Dem
@J_
julien.ledem@wework.com
April 2018

wework

Julien Le Dem
@J_

Principal Data Engineer wework

- Author of Parquet
- Apache member
- Apache PMCs: Arrow, Kudu, Heron, Incubator, Pig, Parquet, Tez
- Used Hadoop first at Yahoo in 2007
- Formerly Twitter Data platform and Dremio

Agenda

- At the beginning there was Hadoop (2005)
- Actually, SQL was invented in the 70s

“MapReduce: A major step backwards”

- The deconstructed database
- What next?

wework

At the beginning there was Hadoop

Hadoop

Storage:

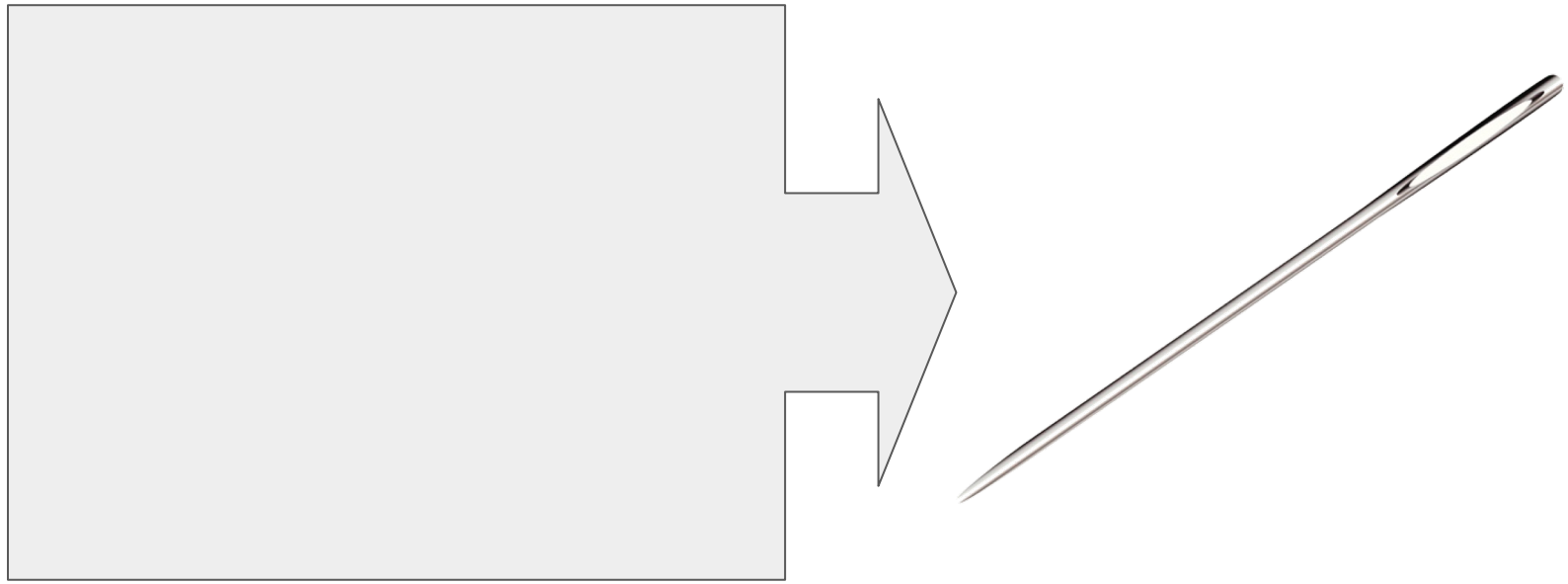
A distributed file system

Execution:

Map Reduce

Based on Google's GFS and MapReduce papers

Great at looking for a needle in a haystack



Great at looking for a needle in a haystack …

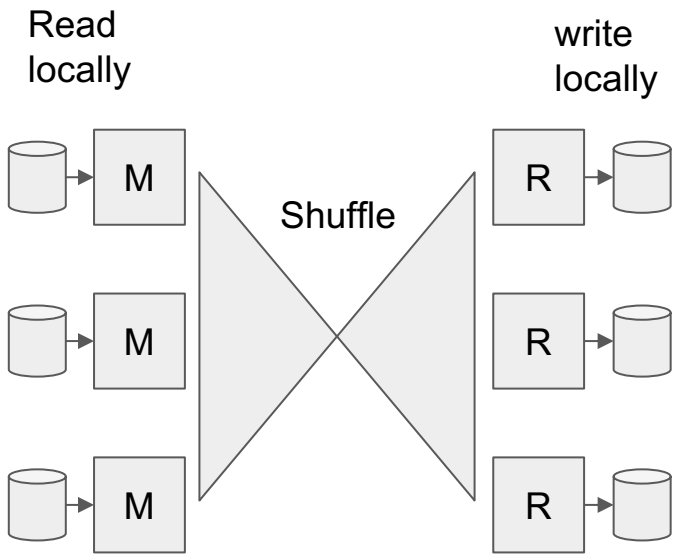
… with snowplows

Original Hadoop abstractions

Execution
Map/Reduce

Simple

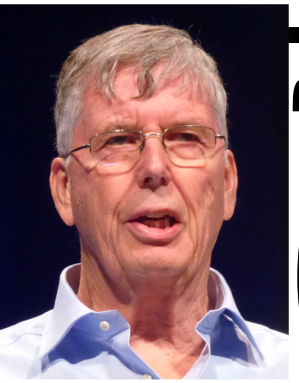
- Flexible/Composable
- Logic and optimizations tightly coupled
- Ties execution with persistence



Storage
File System

Just flat files

- Any binary
- No schema
- No standard
- The job must know how to split the data



“MapReduce: A major step backwards” (2008)

Databases have been around for a long time

Relational model

- First described in 1969 by Edgar F. Codd

SQL

- Originally SEQUEL developed in the early 70s at IBM
- 1986: first SQL standard
- Updated in 1989, 1992, 1996, 1999, 2003, 2006, 2008, 2011, 2016

Global Standard

- Universal Data access language
- Widely understood

Underlying principles of relational databases

Separation of logic and optimization Separation of Schema and Application High level language focusing on logic (SQL) Indexing Optimizer	**Standard** SQL is understood by many
Integrity Transactions Integrity constraints Referential integrity	**Evolution** Views Schemas

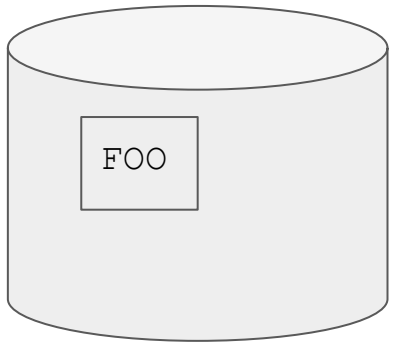
Relational Database abstractions

Execution
SQL

SQL

- Decouples logic of query from:
  - Optimizations
  - Data representation
  - Indexing

```
SELECT a, AVG(b)
FROM FOO GROUP BY a
```

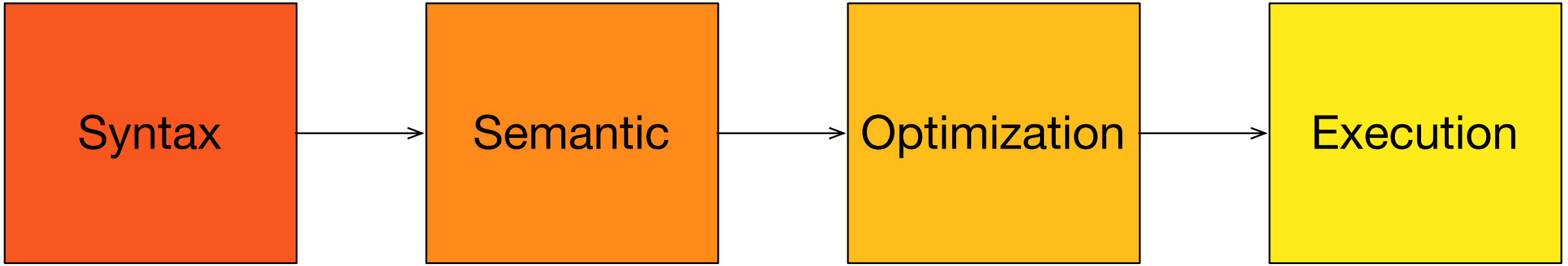


Storage
Tables

Abstracted notion of data

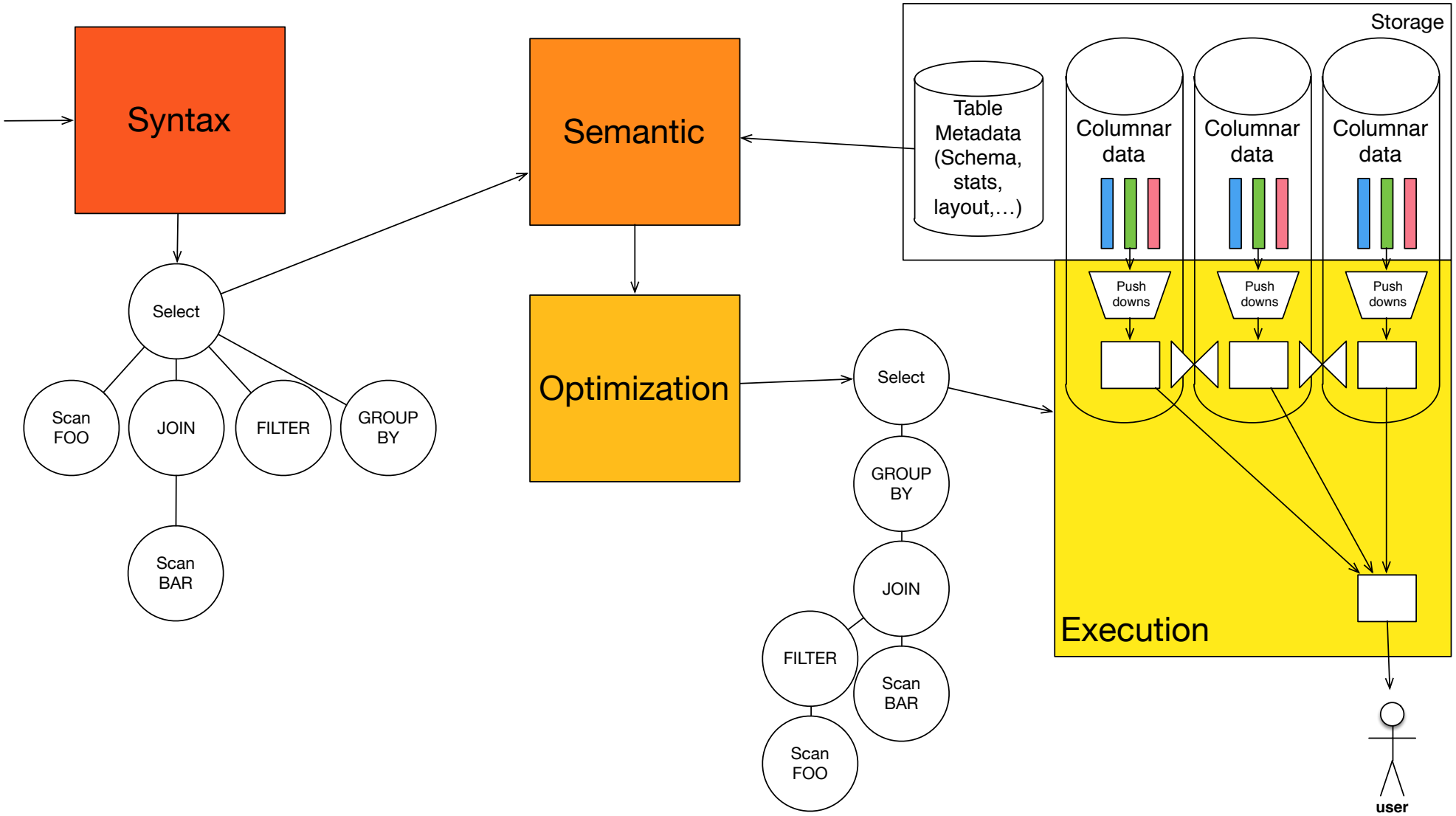
- Defines a Schema
- Format/layout decoupled from queries
- Has statistics/indexing
- Can evolve over time

Query evaluation



A well integrated system

```
SELECT f.c, AVG(b.d)
  FROM FOO f
  JOIN BAR b ON f.a = b.b
  GROUP BY f.c WHERE f.d = x
```



wework

So why? Why Hadoop? Why Snowplows?

The relational model was constrained

Constraints are good

They allow optimizations

- Statistics
- Pick the best join algorithm
- Change the data layout
- Reusable optimization logic

We need the right Constraints

Need the right abstractions

Traditional SQL implementations:

- Flat schema
- Inflexible schema evolution
- History rewrite required
- No lower level abstractions
- Not scalable

wework

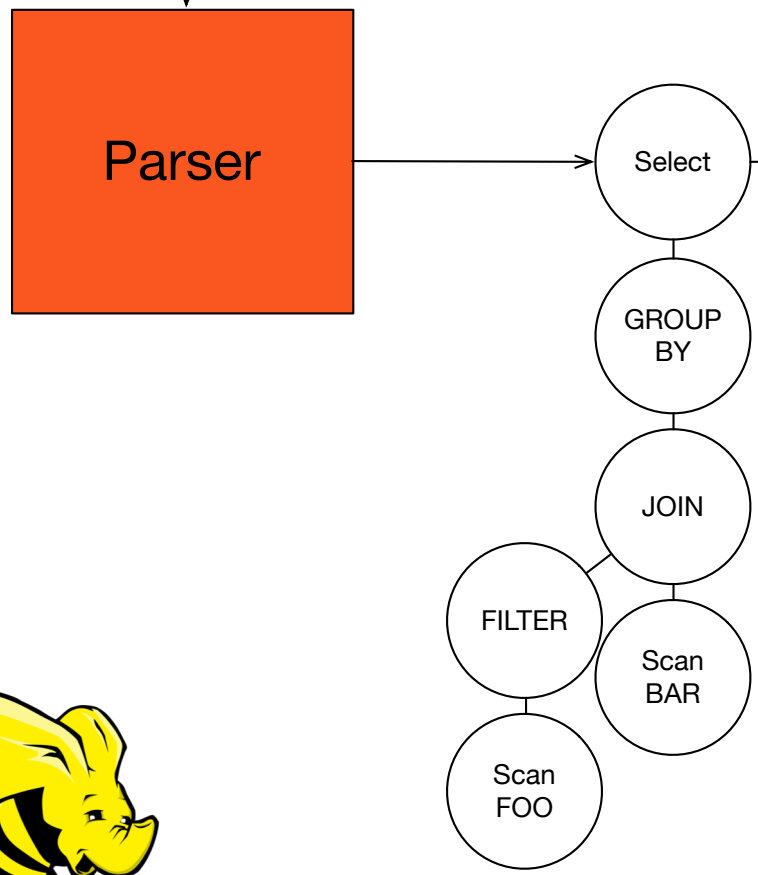
Hadoop is flexible and scalable

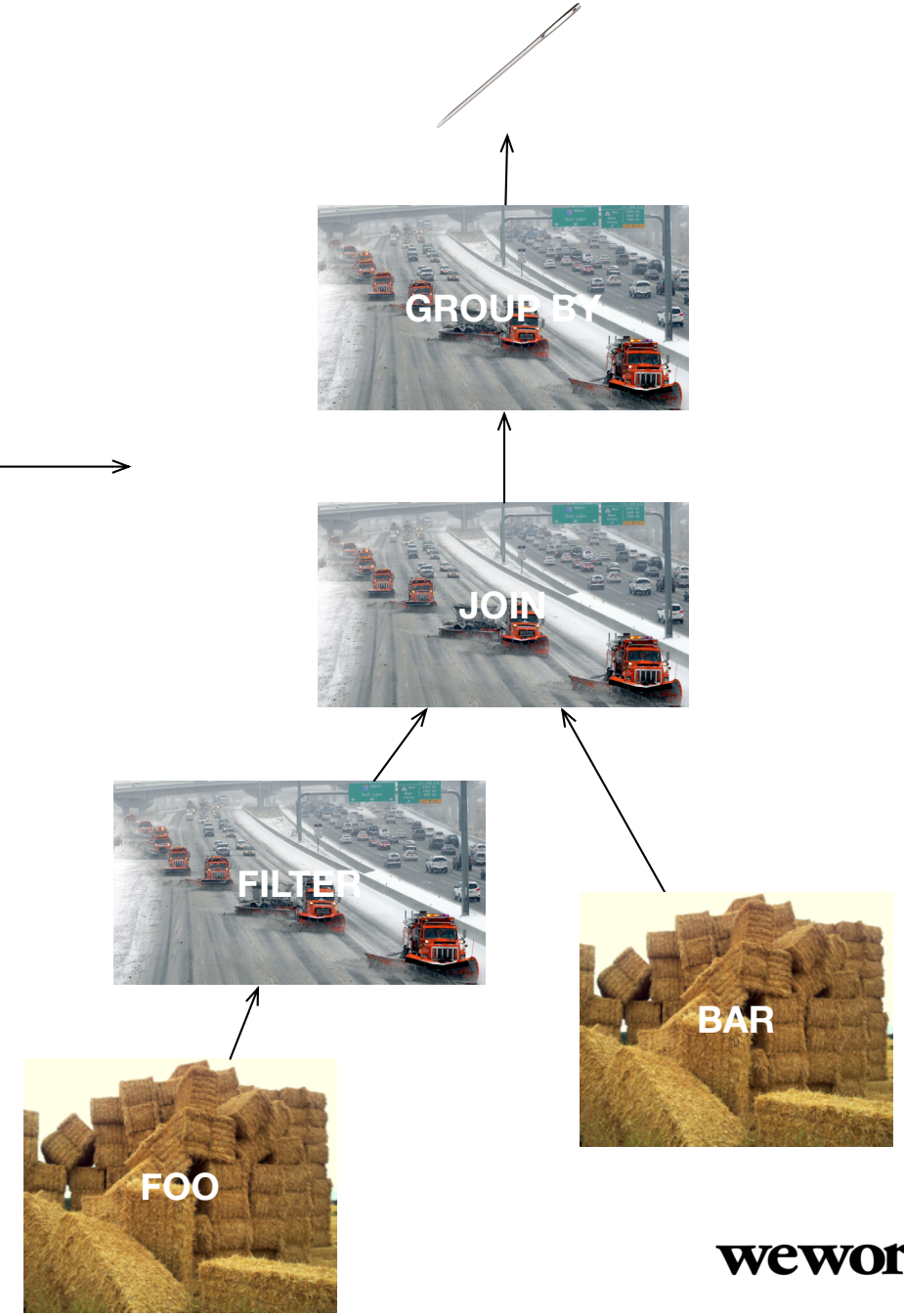
 **No Data shape constraint**	• Nested data structures • Unstructured text with semantic annotations • Graphs • Non-uniform schemas
 **It's just code**	• Room to scale algorithms that are not part of the standard • Machine learning • Your imagination is the limit
 **Open source**	• You can improve it • You can expand it • You can reinvent it

You can actually implement SQL with this

And they did...

```
SELECT f.c, AVG(b.d)
  FROM FOO f
  JOIN BAR b ON f.a = b.b
  GROUP BY f.c WHERE f.d = x
```





(open-sourced 2009)

wework

10 years later

wework

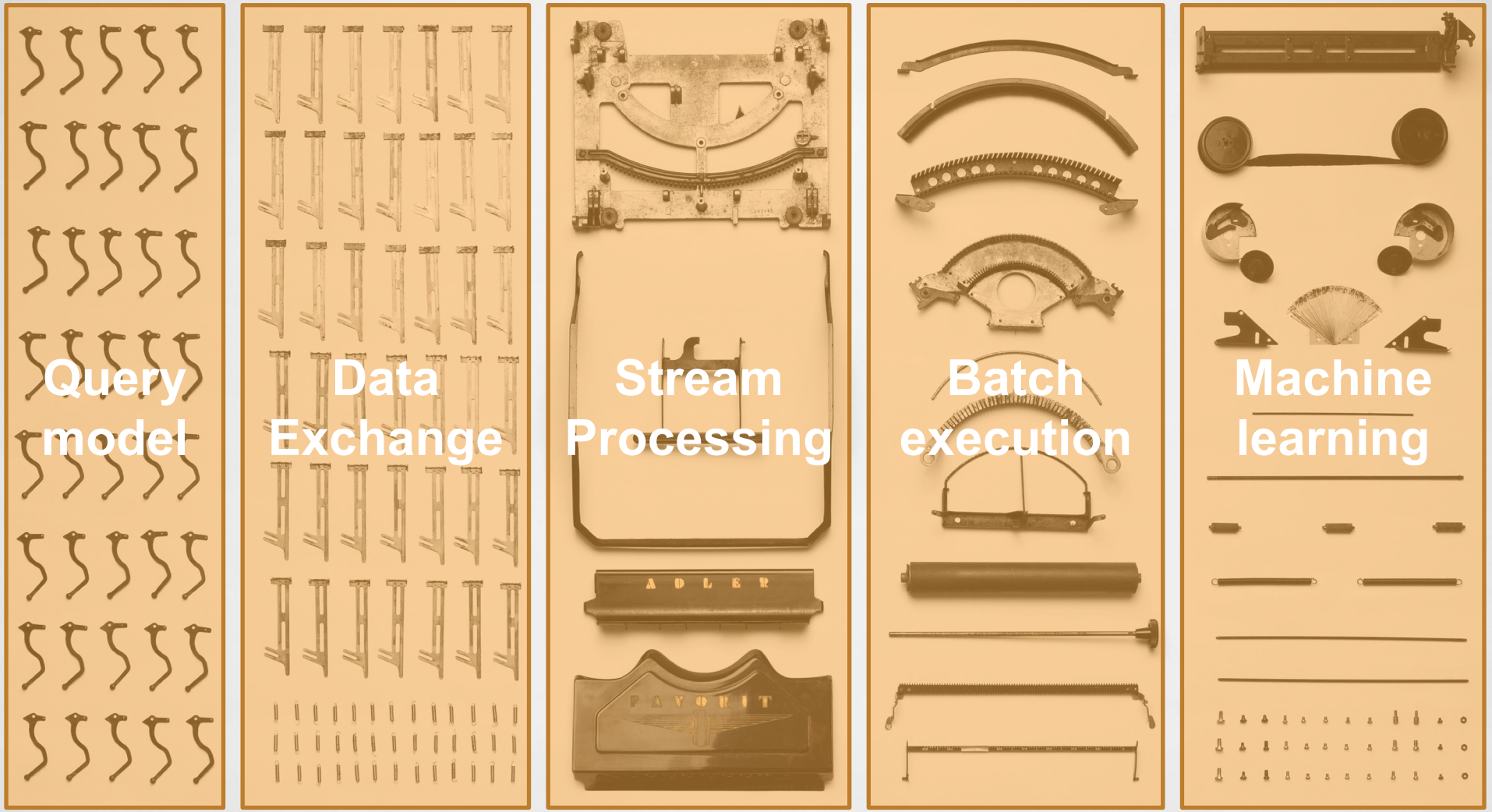
The deconstructed database

Author: gamene https://www.flickr.com/photos/gamene/4007091102

wework

The deconstructed database

The deconstructed database



We can mix and match individual components

Stream processing

Spark Streaming, HERON, APACHE STORM™, Apache Flink, APACHE kafka® Streams, APEX, samza

Stream persistance

APACHE kafka®, PULSAR

Storage

cassandra, APACHE KUDU, APACHE HBASE, hadoop HDFS, Parquet, AVRO

Execution

APACHE Spark™, hadoop

SQL

HIVE, APACHE DRILL, APACHE PHOENIX, APACHE KYLIN, Spark SQL, presto

Resource management

kubernetes, hadoop YARN, Apache MESOS™

Machine learning

TensorFlow, mahout, Spark MLlib, PYTORCH

Specialized Components

Apache Calcite,  , beam

*not exhaustive!

wework

We can mix and match individual components

Storage

Row oriented or columnar
Immutable or mutable
Stream storage vs analytics optimized

Query model

SQL
Functional
…

Machine Learning

Training models

Data Exchange

Row oriented
Columnar

Streaming Execution

Optimized for High Throughput and Low latency processing

Batch Execution

Optimized for high throughput and historical analysis

Emergence of standard components

wework

Emergence of standard components

Columnar Storage

Apache Parquet as columnar representation at rest.

SQL parsing and optimization

Apache Calcite as a versatile query optimizer framework

Schema model

Apache Avro as pipeline friendly schema for the analytics world.

Columnar Exchange

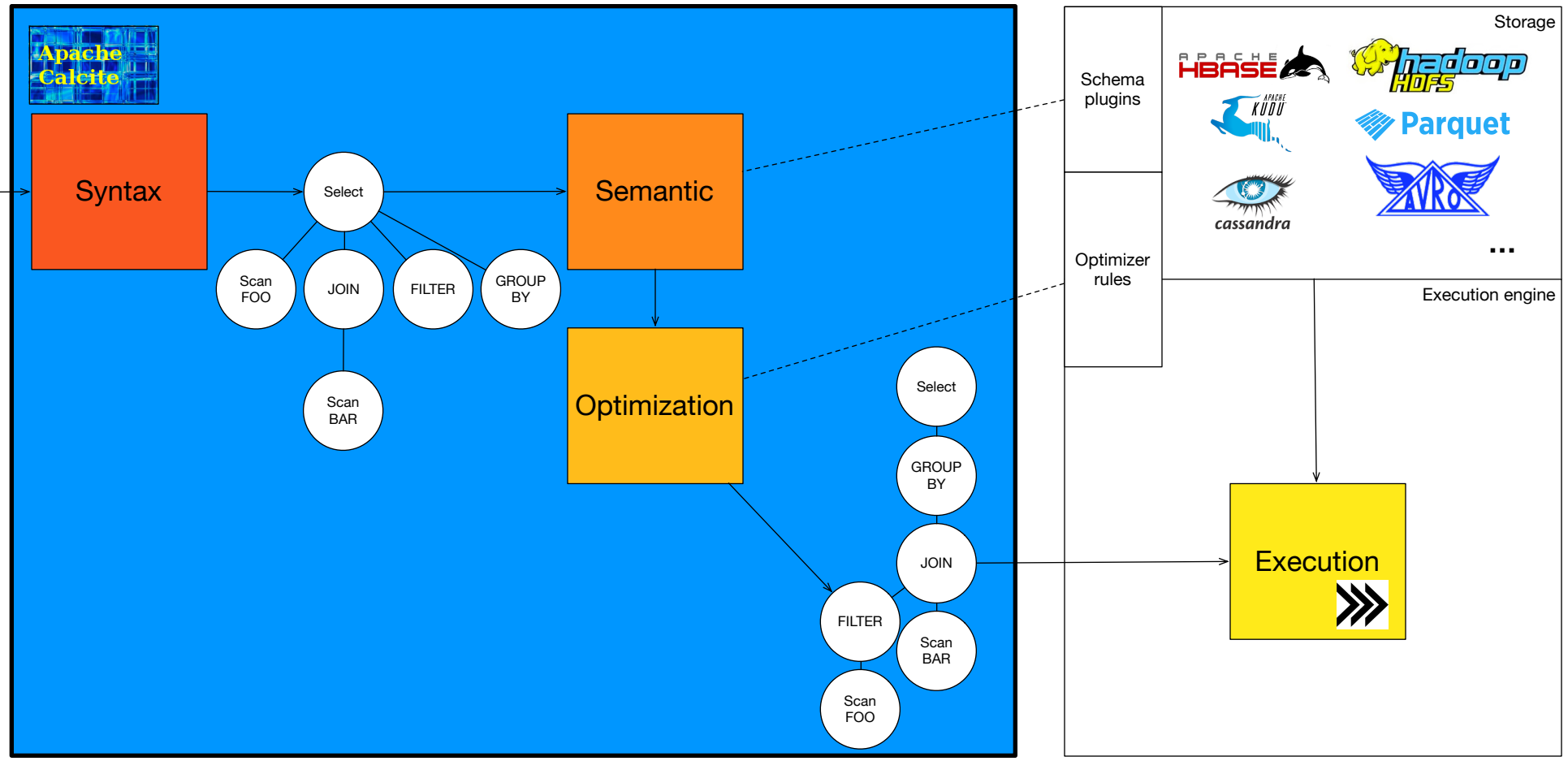
Apache Arrow as the next generation in-memory representation and no-overhead data exchange

Table abstraction

Netflix's Iceberg has a great potential to provide Snapshot isolation and layout abstraction on top of distributed file systems.

The deconstructed database's optimizer: Calcite

```
SELECT f.c, AVG(b.d)
  FROM FOO f
  JOIN BAR b ON f.a = b.b
  GROUP BY f.c WHERE f.d = x
```



wework

Apache Calcite is used in:

Batch SQL

- Apache Hive
- Apache Drill
- Apache Phoenix
- Apache Kilin

…

Streaming SQL

- Apache Apex
- Apache Flink
- Apache SamzaSQL
- Apache StormSQL

…

The deconstructed database's storage

Immutable	 Columnar  Row oriented 
Mutable	Optimized for analytics  Optimized for serving 
Query integration	To be performant a query engine requires deep integration with the storage layer. Implementing push down and a vectorized reader producing data in an efficient representation (for example Apache Arrow).

Storage: Push downs

PROJECTION
Read only what you need

Read only the columns that are needed:

- Columnar storage makes this efficient.

PREDICATE
Filter

Evaluate filters during scan to:

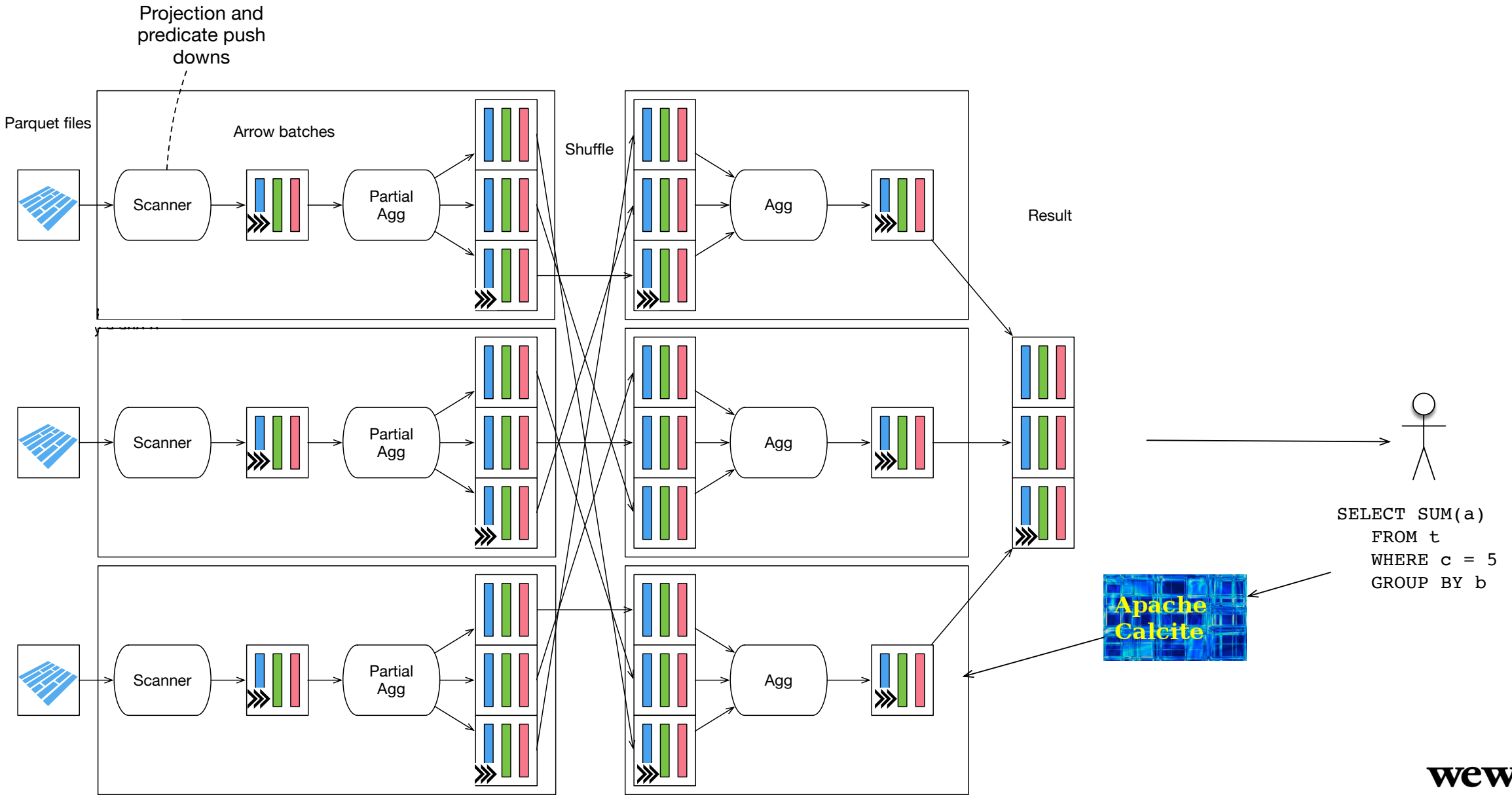
- Leverage storage properties (min/max stats, partitioning, sort, etc)
- Avoid decoding skipped data.
- Reduce IO.

AGGREGATION
Avoid materializing intermediary data

To reduce IO, aggregation can also be implemented during the scan to:

- minimize materialization of intermediary data

The deconstructed database interchange: Apache Arrow



wework

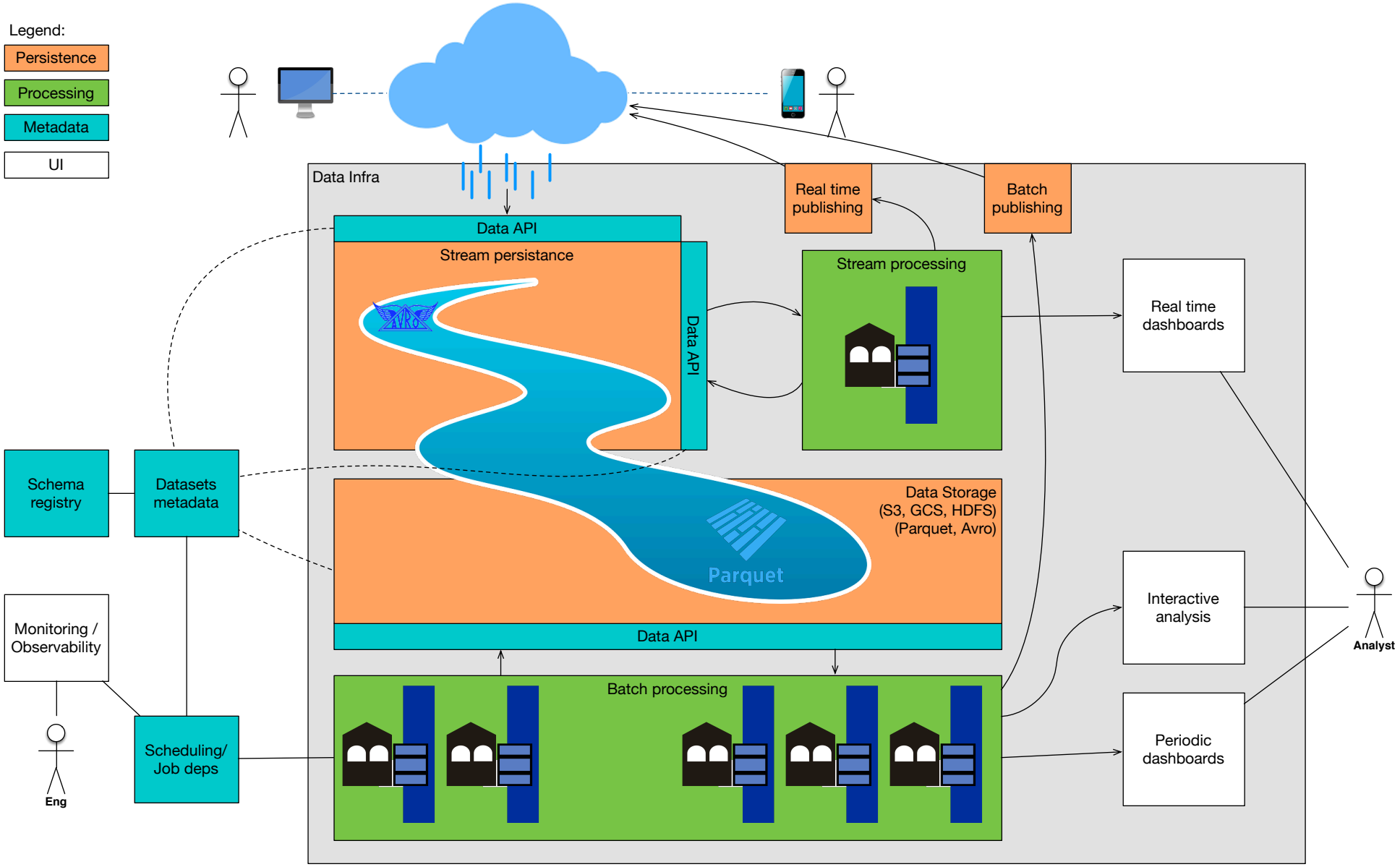
Storage: Stream persistence

Open source projects	Incubent  Interesting 
Features	• State of consumer: how do we recover from failure • Snapshot • Decoupling reads from writes • Parallel reads • Replication • Data isolation

Big Data infrastructure blueprint

Big Data infrastructure blueprint

Big Data infrastructure blueprint



wework

The Future

wework

Still Improving

A better data abstraction

- A better metadata repository
- A better table abstraction: Netflix/Iceberg
- Common Push down implementations (filter, projection, aggregation)

Better interoperability

- More efficient interoperability:

Continued Apache Arrow adoption

Better data governance

- Global Data Lineage
- Access control
- Protect privacy
- Record User consent

wework

Some predictions

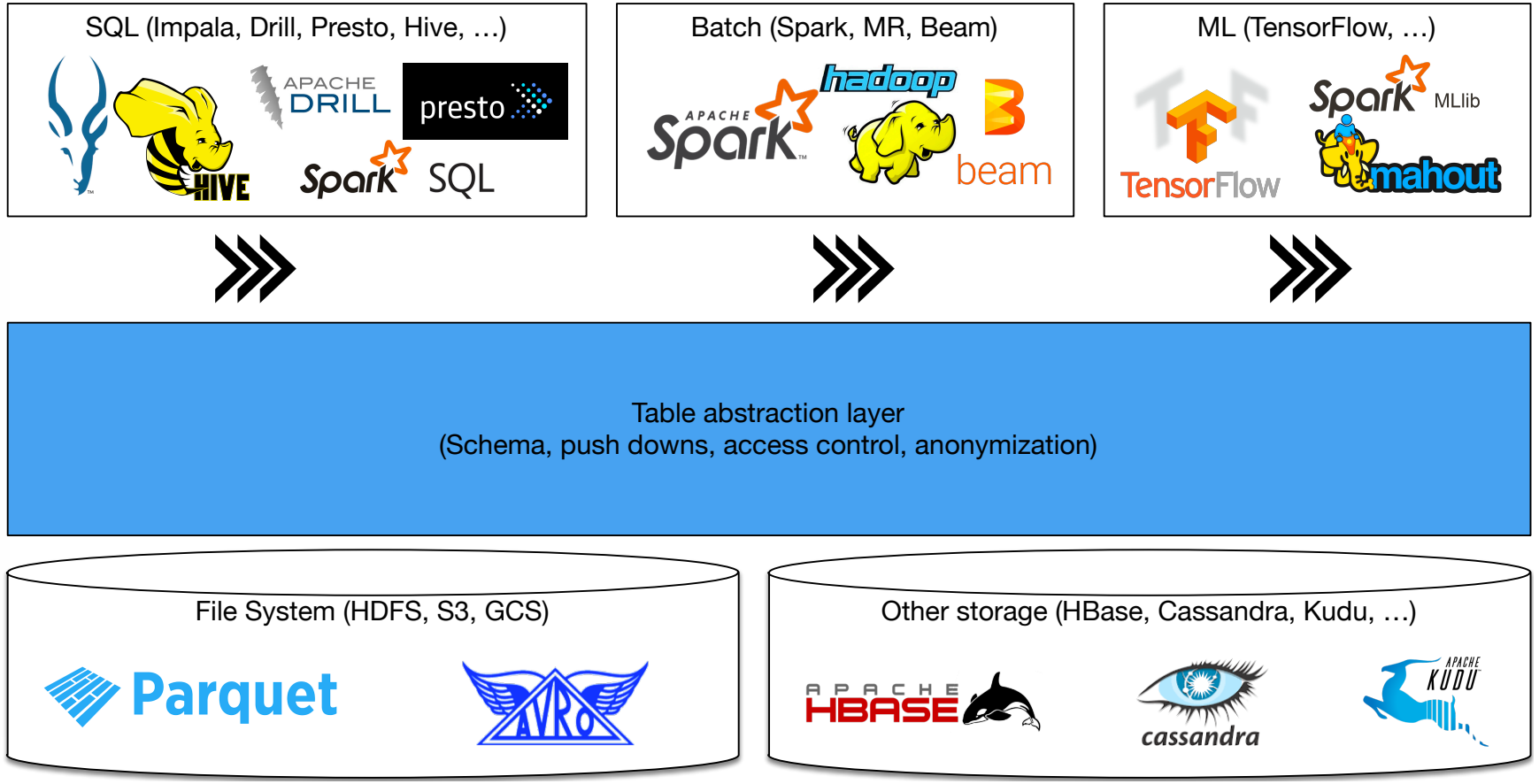
wework

A common access layer

Distributed access service

Centralizes:

- Schema evolution
- Access control/anonymization
- Efficient push downs
- Efficient interoperability

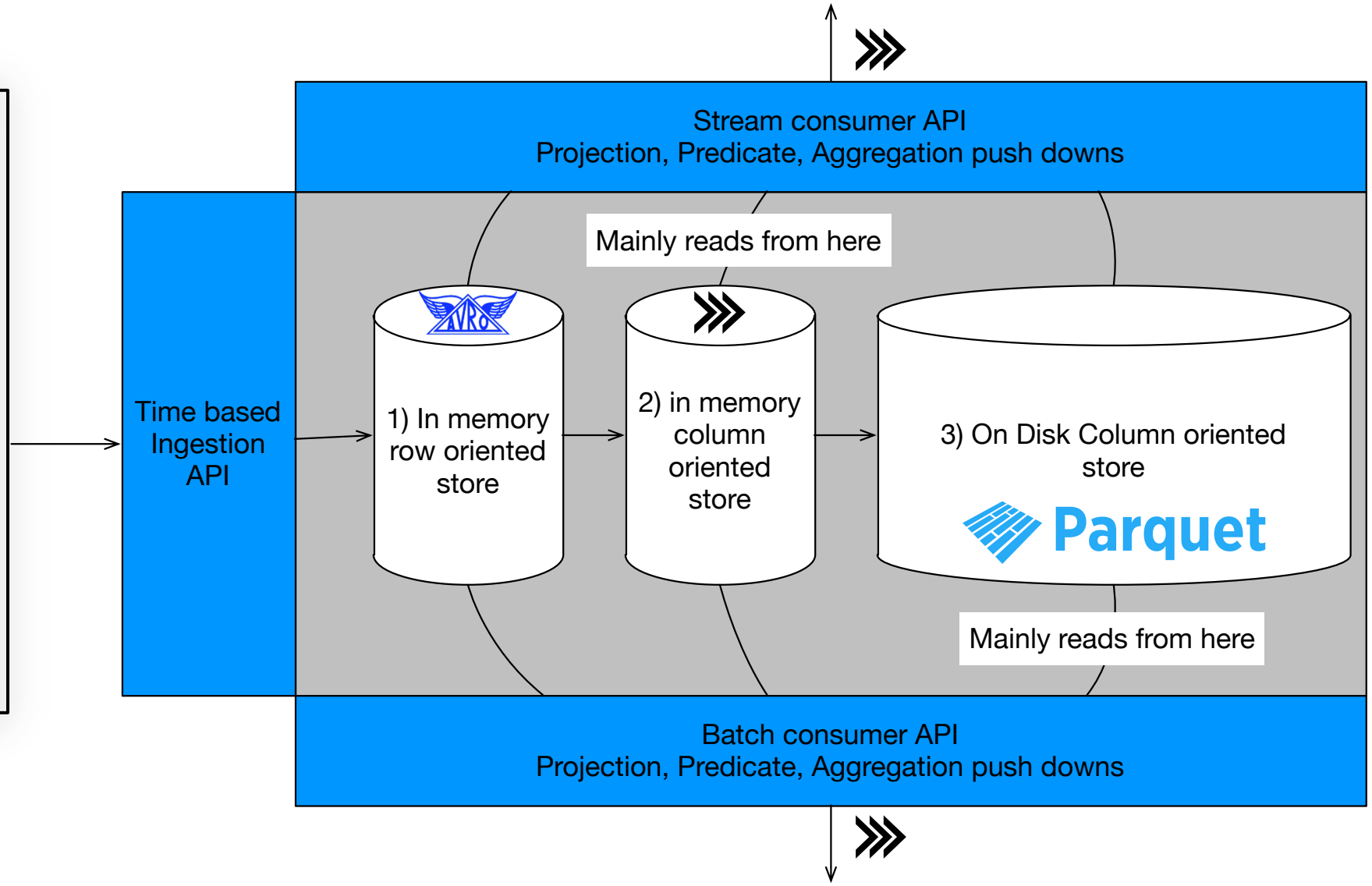


A multi tiered streaming batch storage system

Batch-Stream storage integration

Convergence of Kafka and Kudu

- Schema evolution
- Access control/anonymization
- Efficient push downs
- Efficient interoperability



Questions?

Julien Le Dem @J_ julien.ledem@wework.com
April 2018

wework

THANK YOU!

julien.ledem@wework.com

Julien Le Dem @J_
April 2018

wework