DATA+AI SUMMIT 2022

DATA FUSION

DataFusion and Apache Arrow

APACHE ARROW

Supercharge Your Data Analytical Tool with a Rusty Query Engine

ORGANIZED BY databricks

Andrew Lamb
Staff Engineer, InfluxData
Apache Arrow PMC

Daniël Heres
Data Engineer, GoDataDriven
Apache Arrow PMC

Introduction

Your Speakers

Andrew

Staff Engineer @ InfluxData

Previously

- Query Optimizer @ Vertica, Oracle Database server, embedded Compilers
- Chief Architect + VP Engineering roles at ML startups

Daniël

Data/ML Engineer @ GoDataDriven

Previously

- Data / ML Engineer @ bol.com
- Startups

Why should you care?

Andrew

Recent Proliferation of Big Data systems

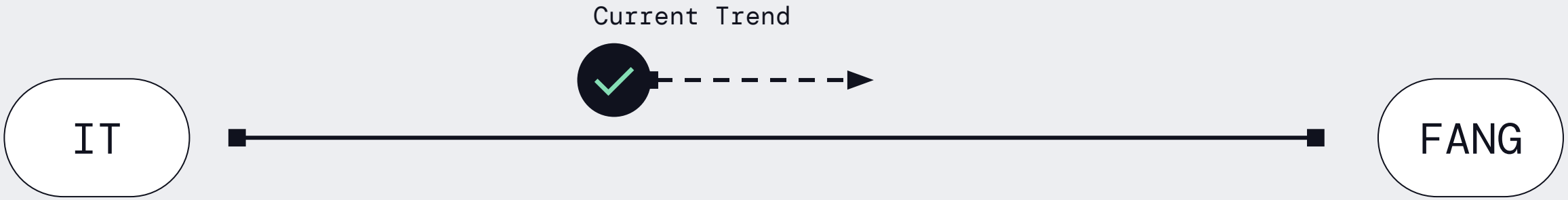
...

Recent Proliferation of Databases


Such
datatabase
so
wow
imgflip.com

What is going on?

COTS → Totally Custom



"Buy and Operate"

- Buy software from vendors
- Operate on your own hardware, with sysadmins

"Assemble and Operate"

- Assemble from open source technologies
- Operate on resources in a public cloud

"Build and Operate"

- Write software for, and operate all components
- Optimized for exact needs

Apache Arrow

Multi-language toolkit for Processing and Interchange

Founded in 2016

Apache Software Foundation

Low level / foundational technology to build fast and interoperable analytic systems

Open standard, implementations in 12+ languages

Adopted widely in industry products and open source projects

DataFusion: A Query Engine

"DataFusion is an extensible query execution framework, written in Rust, that uses Apache Arrow as its in-memory format."

- DataFusion Website

DataFusion: A Query Engine

```
SELECT status, COUNT(1)
FROM http_api_requests_total
WHERE path = '/api/v2/write'
GROUP BY status;
```

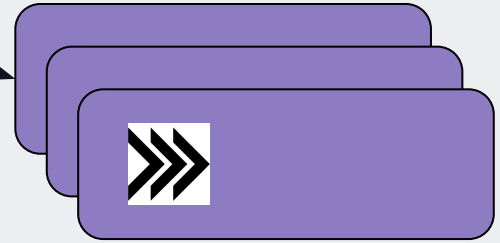
SQL Query

```
ctx.read_table("http")?
  .filter(...)?
  .aggregate(..)?;
```

DataFrame

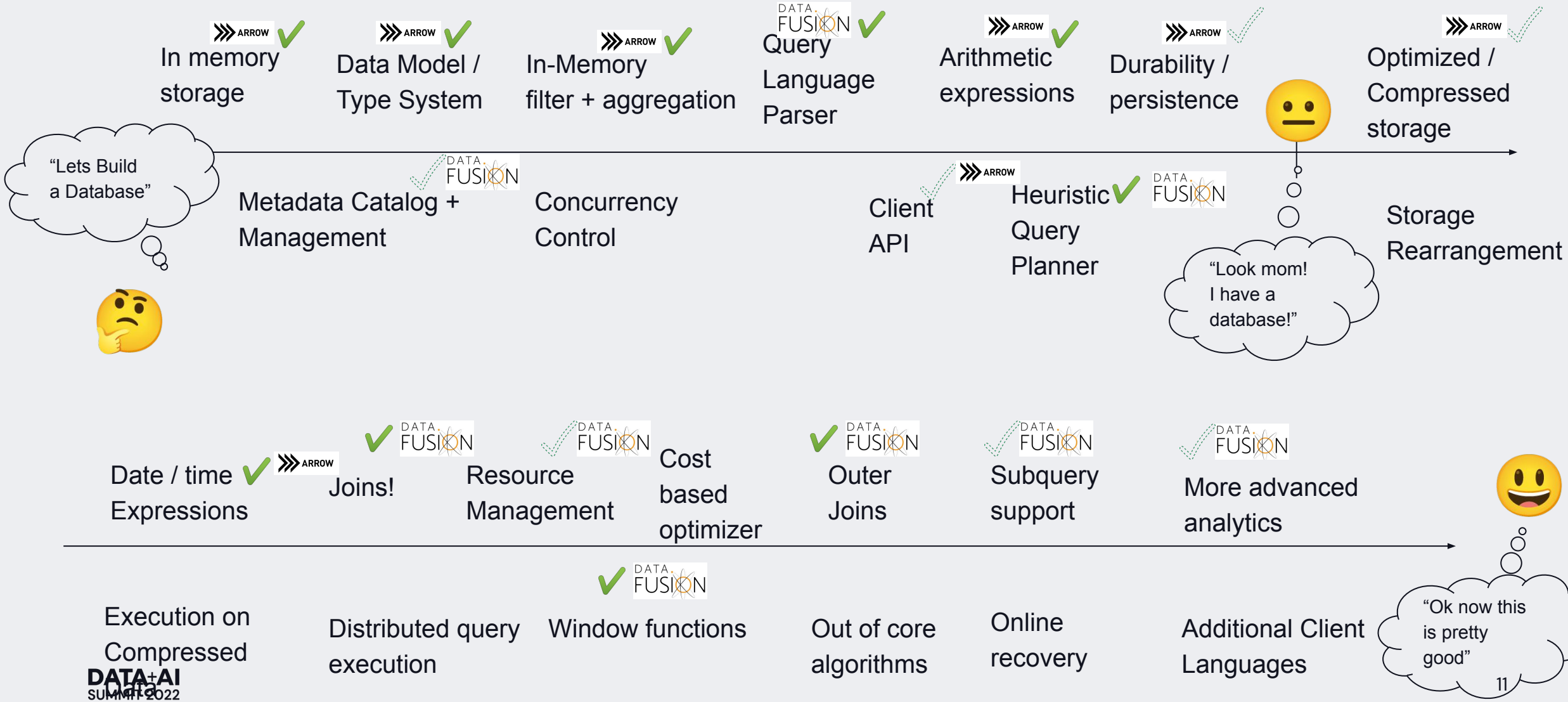
Catalog information:
tables, schemas, etc

Data Batches



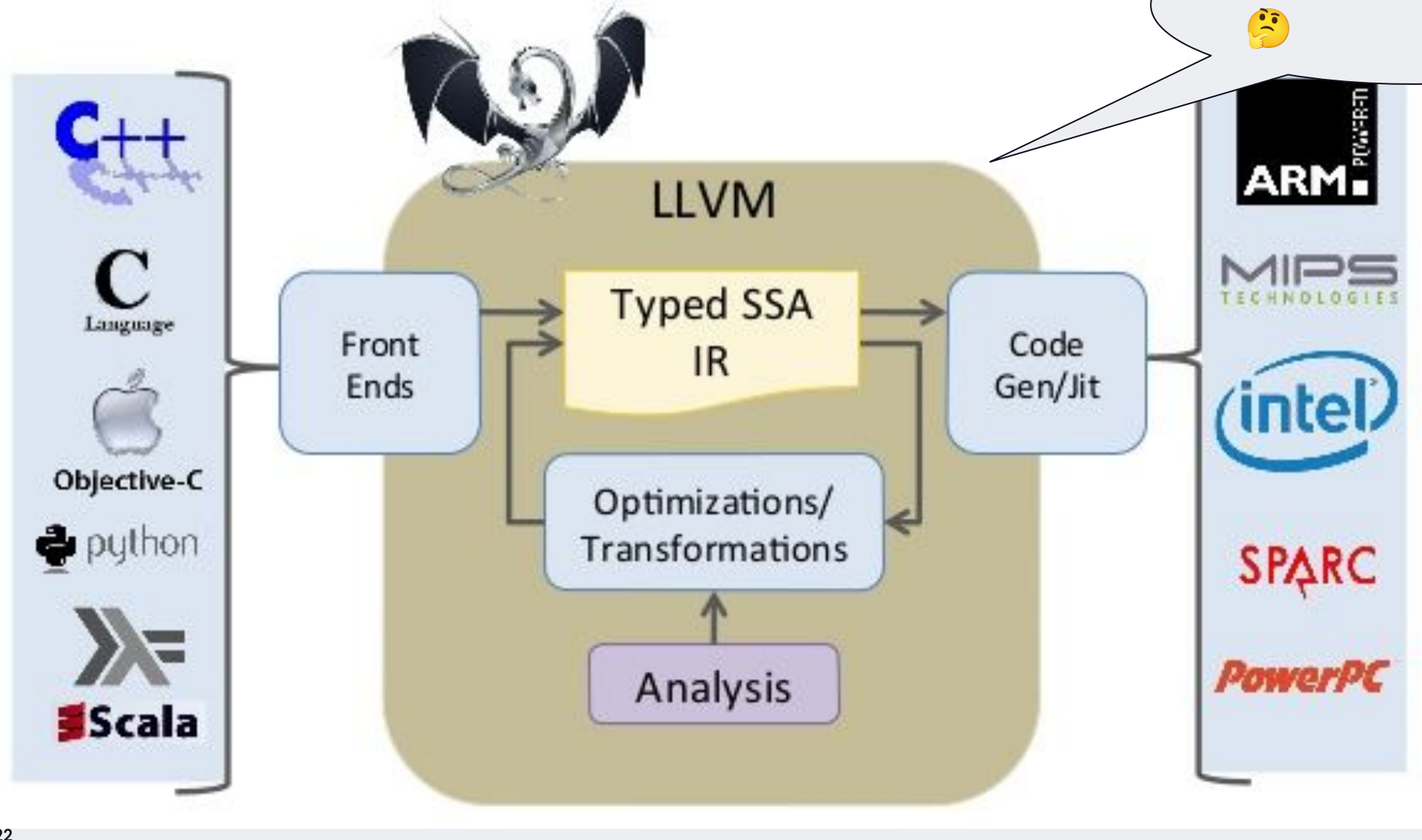
Data Batches

Implementation timeline for a new Database system

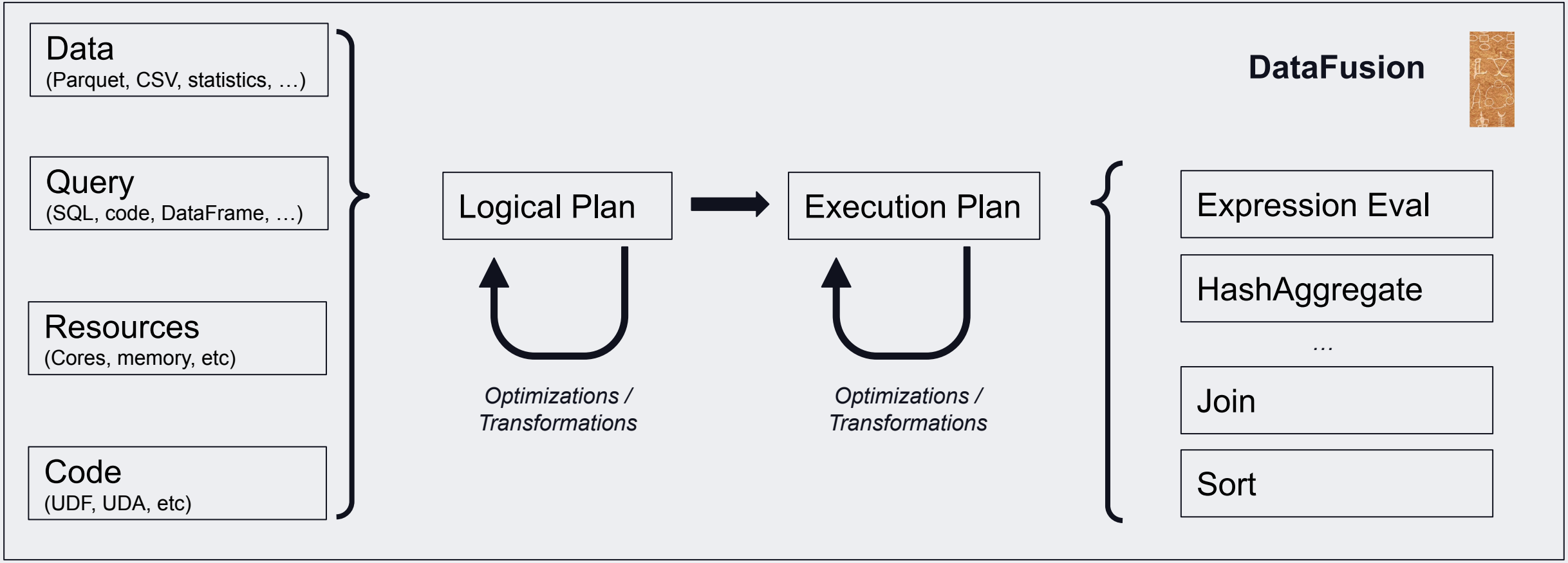


LLVM Compiler Infrastructure

[Lattner et al.]



LLVM-like Infrastructure for Databases

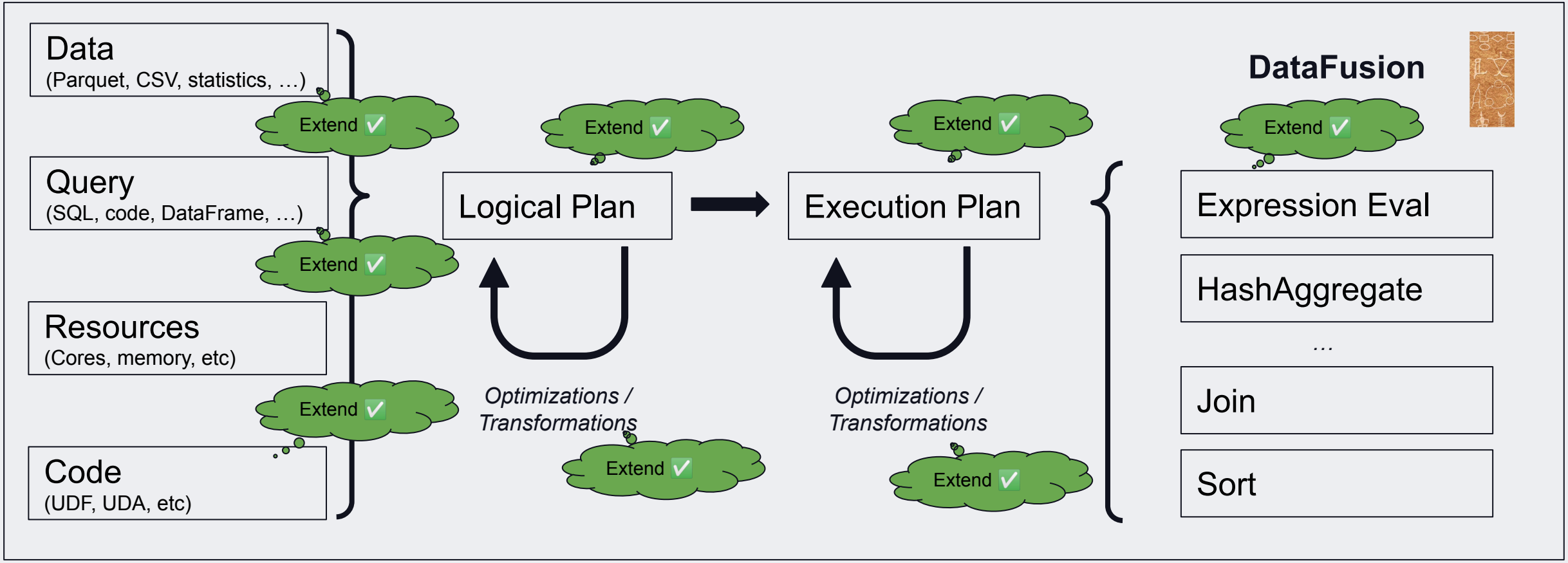


Inputs

Plan Representations (DataFlow Graphs)

Optimized Execution Operators (Arrow Based)

DataFusion: Totally Customizable

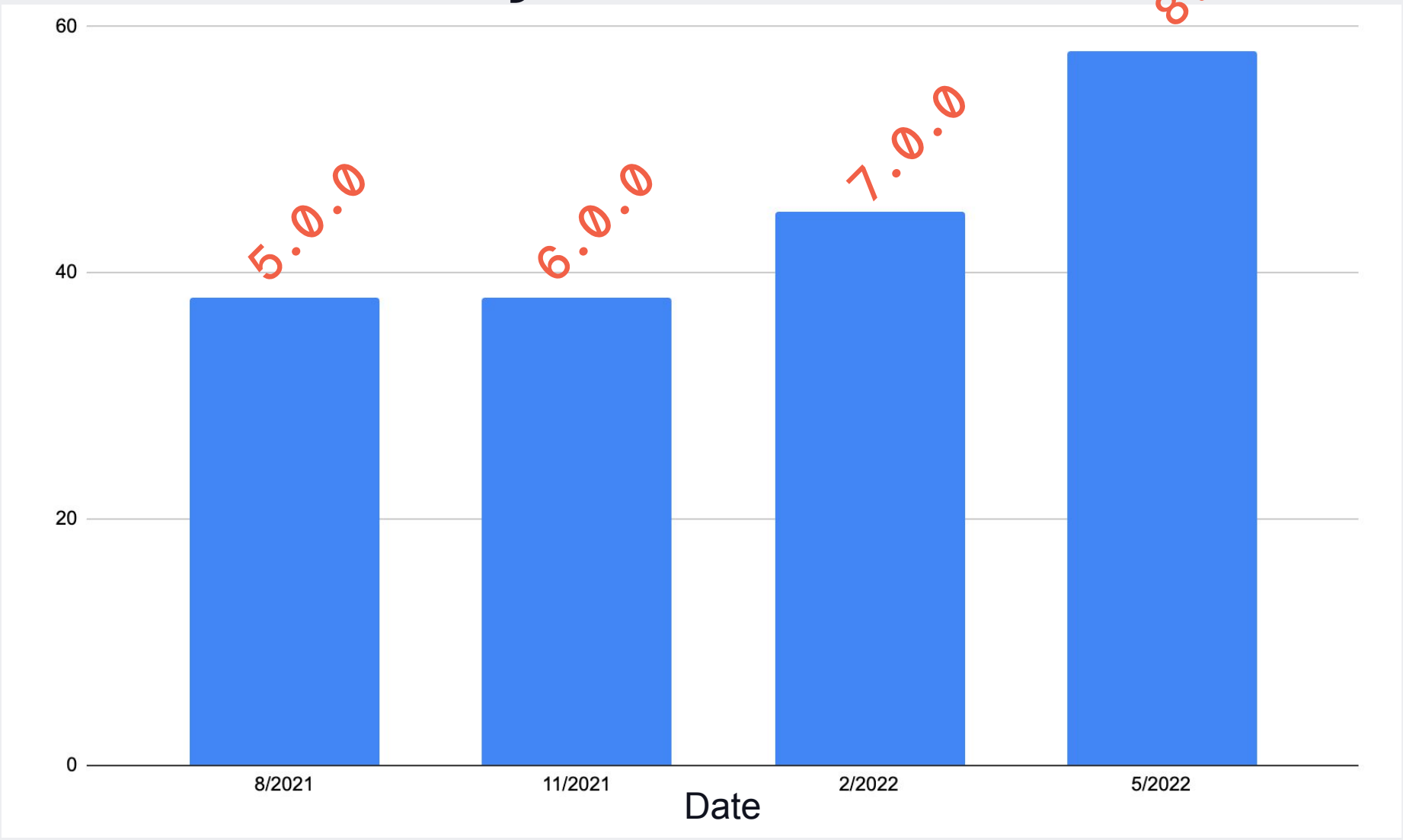


Inputs

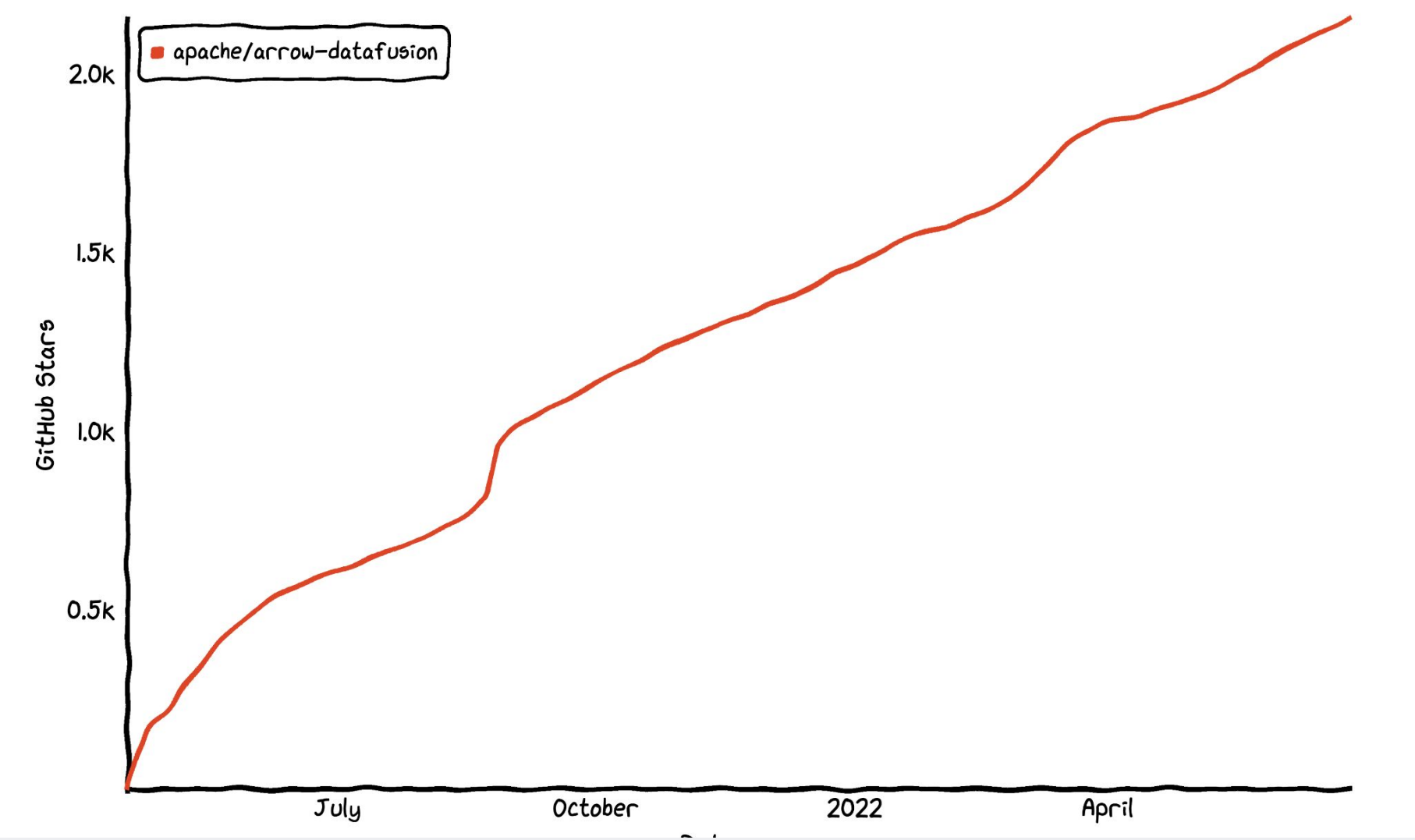
Plan Representations (DataFlow Graphs)

Optimized Execution Operators (Arrow Based)

DataFusion Project Growth



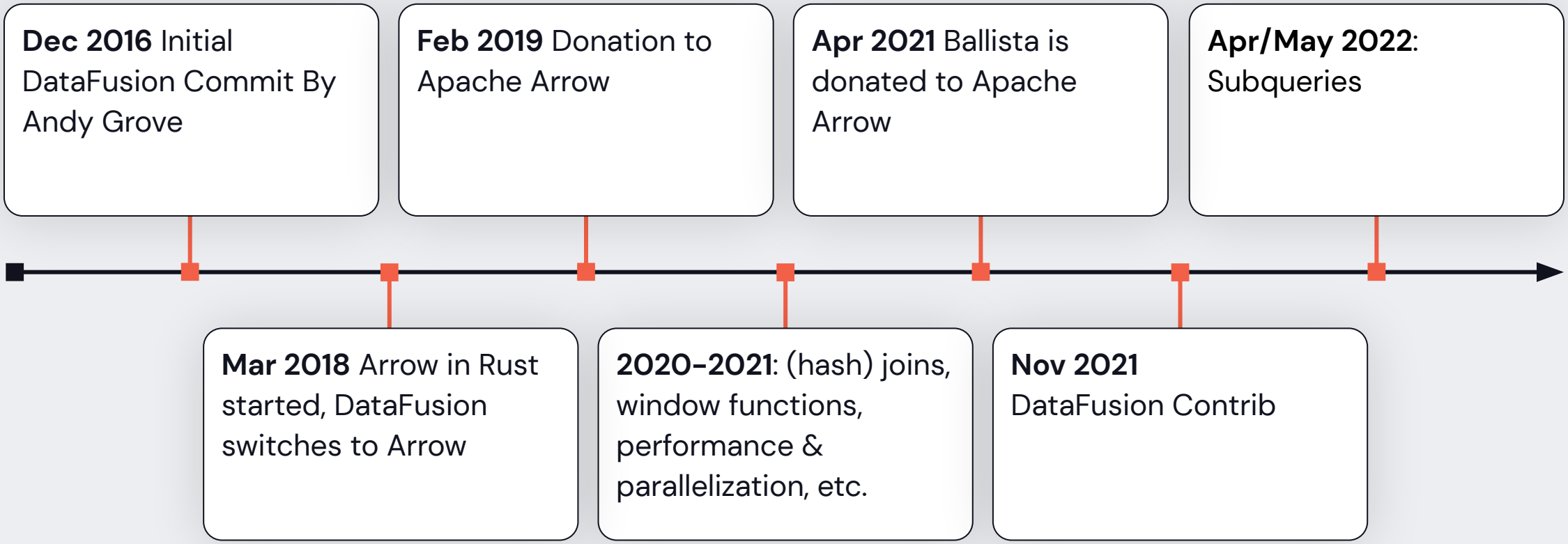
DataFusion Project Growth



https://star-history.com/#apache/arrow-datafusion&Date

DataFusion Milestones: Time to Mature

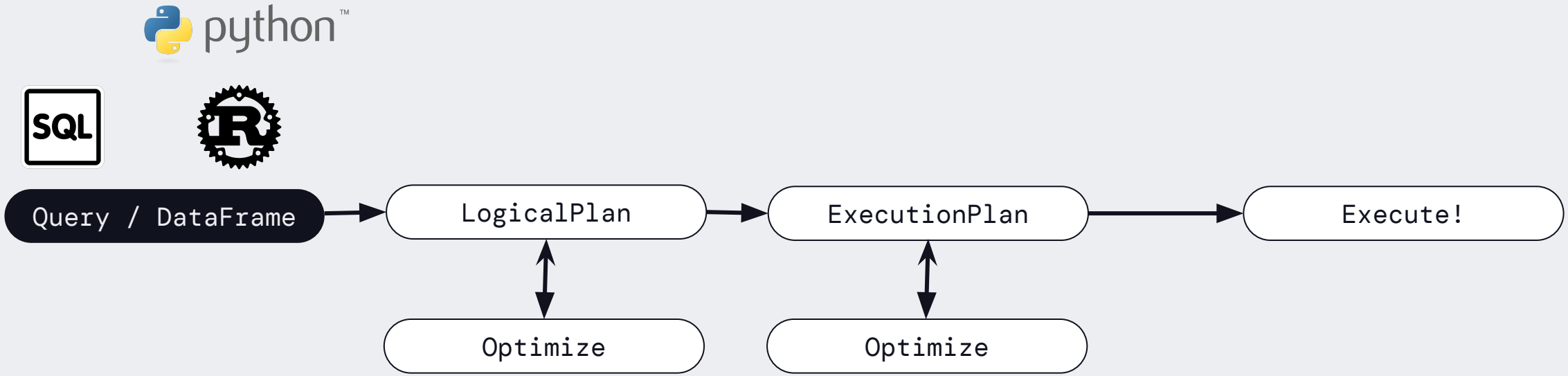
5+ year labor of love



Overview of Apache Arrow DataFusion

Daniël

From Query to Results



From Query to Results

an example

```
1  select
2      count(*) num_visitors,
3      job_title
4  from
5      visitors
6  where
7      city = "San Francisco"
8  group by
9      job_title
10
```

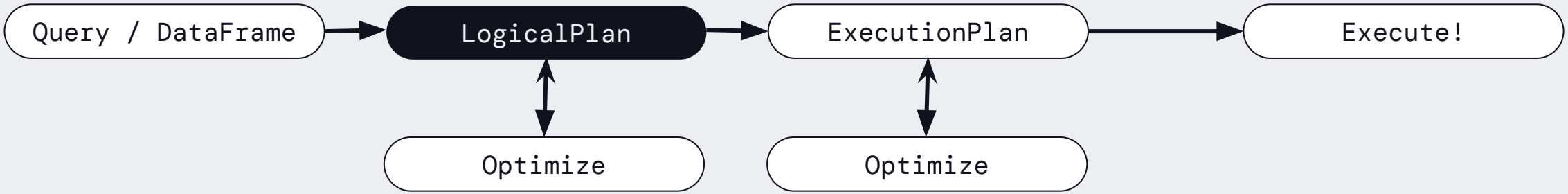
From Query to Results

datafusion package available via PyPI

```
visitors = ctx.table("visitors")
df = (
    visitors.filter(col("city") == literal("San Francisco"))
    .aggregate([col("job_title")], [f.count(literal(1))])
)
batches = df.collect() # collect results into memory (Arrow batches)
```

From Query to Results

Logical Plan represents the *what*

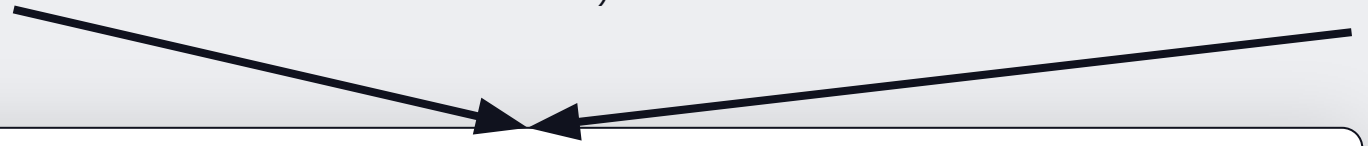


Initial Logical Plan

SQL is *parsed*, then *translated* into a initial Logical Plan.

```
select count(*) num_visitors, job_title
from visitors
where city = "San Francisco"
group by job_title
```

```
visitors = ctx.table("visitors")
df = (
    visitors.filter(col("city") == literal("San Francisco"))
    .aggregate([col("job_title")], [f.count(literal(1))])
)
```

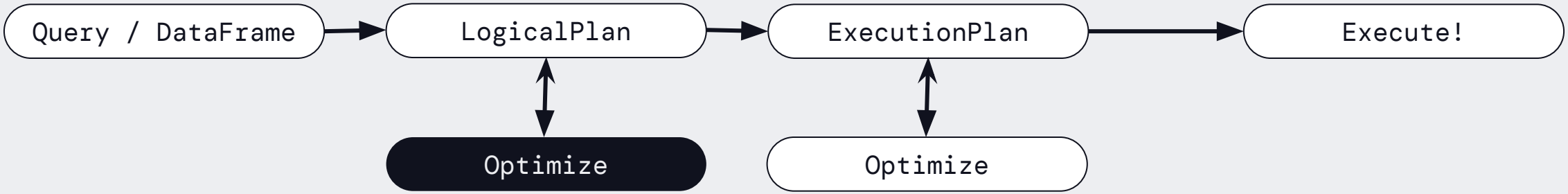


```
Projection: #COUNT(UInt8(1)) AS num_visitors, #visitors.job_title
  Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
    Filter: #visitors.city = Utf8("San Francisco")
      TableScan: visitors projection=None
```

(Read plan from bottom to top)

Let's Optimize!

- Massively speed up execution times (10x, 100x, 1000x) by rewriting queries to a equivalent, optimized version
- 14 built-in optimization passes in DataFusion, adding more each version
- **Add custom optimization passes**



Let's Optimize!

Projection Pushdown

Minimizing IO (especially useful for formats like Parquet), processing

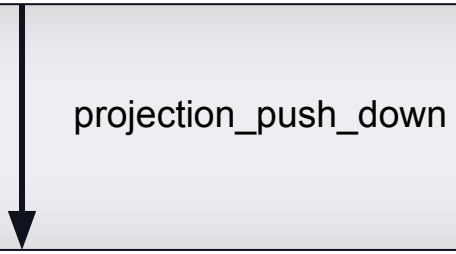
```
Projection: #COUNT(UInt8(1)) AS num_visitors, #visitors.job_title
   Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
      Filter: #visitors.city = Utf8("San Francisco")
         TableScan: visitors projection=None
```

Let's Optimize!

Projection Pushdown

Minimizing IO (especially useful for formats like Parquet), processing

```
Projection: #COUNT(UInt8(1)) AS num_visitors, #visitors.job_title
   Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
      Filter: #visitors.city = Utf8("San Francisco")
         TableScan: visitors projection=None
```




```
Projection: #COUNT(UInt8(1)) AS num_visitors, #visitors.job_title
   Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
      Filter: #visitors.city = Utf8("San Francisco")
         TableScan: visitors projection=Some([0, 1])
```

Let's Optimize!

Filter Pushdown

Minimizing IO (especially useful for formats like Parquet), processing

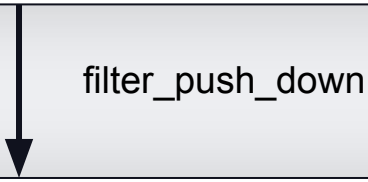
```
Projection: #COUNT(UInt8(1)) AS n, #visitors.job_title
   Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
      Filter: #visitors.city = Utf8("San Francisco")
         TableScan: visitors projection=Some([0, 1])
```

Let's Optimize!

Filter Pushdown

Minimizing IO (especially useful for formats like Parquet), processing

```
Projection: #COUNT(UInt8(1)) AS n, #visitors.job_title
   Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
      Filter: #visitors.city = Utf8("San Francisco")
         TableScan: visitors projection=Some([0, 1])
```

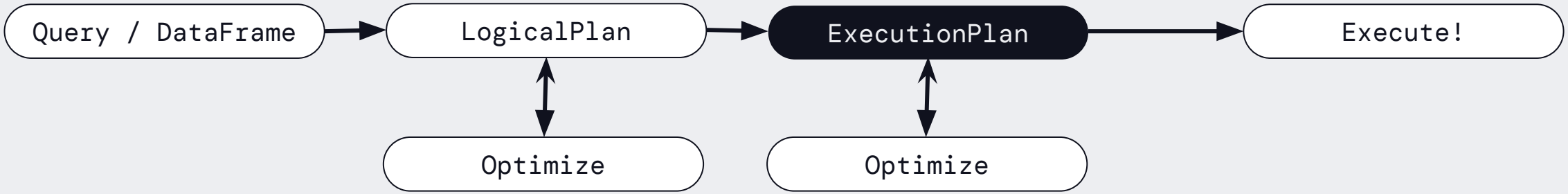



```
Projection: #COUNT(UInt8(1)) AS n, #visitors.job_title
   Aggregate: groupBy=[[#visitors.job_title]], aggr=[[COUNT(UInt8(1))]]
      Filter: #visitors.city = Utf8("San Francisco")
         TableScan: visitors projection=Some([0, 1]), partial_filters=[#visitors.city = Utf8("San Francisco")]
```

Let's Create...

The ExecutionPlan

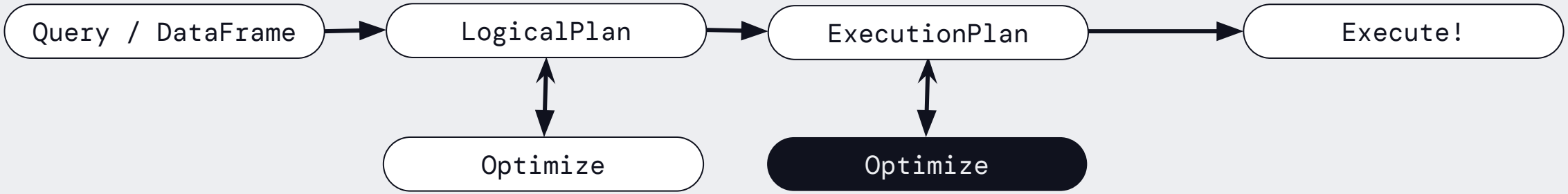
The Execution Plan represents the *where* and *how*



The Initial Execution Plan

```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
      HashAggregateExec: mode=Partial, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
        FilterExec: city@1 = San Francisco
          CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```

And... Optimize!



Optimize

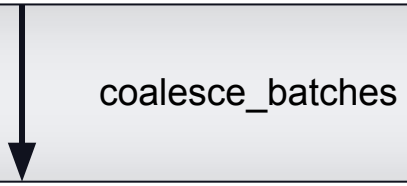
CoalesceBatches: Avoiding small batch size

```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
      HashAggregateExec: mode=Partial, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
        FilterExec: city@1 = San Francisco
          CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```

Optimize

CoalesceBatches: Avoiding small batch size

```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
      HashAggregateExec: mode=Partial, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
        FilterExec: city@1 = San Francisco
          CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```




```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    CoalesceBatchesExec: target_batch_size=4096
      RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
        CoalesceBatchesExec: target_batch_size=4096
          FilterExec: city@1 = San Francisco
            CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```

Optimize

Repartition: Introducing parallelism

```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    CoalesceBatchesExec: target_batch_size=4096
      RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
        CoalesceBatchesExec: target_batch_size=4096
          FilterExec: city@1 = San Francisco
            CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```

Optimize

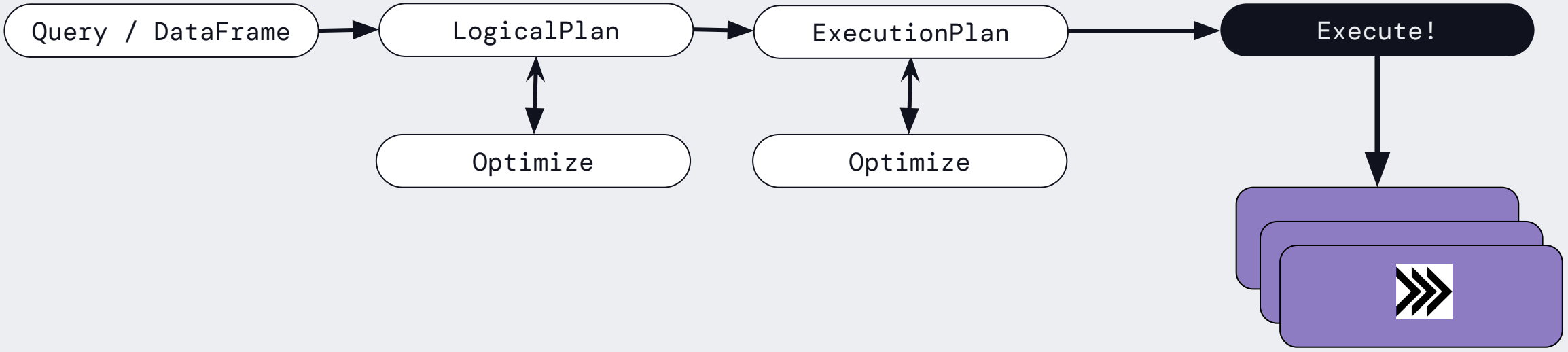
Repartition: Introducing parallelism

```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    CoalesceBatchesExec: target_batch_size=4096
      RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
        CoalesceBatchesExec: target_batch_size=4096
          FilterExec: city@1 = San Francisco
            CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```


```
ProjectionExec: expr=[COUNT(UInt8(1))@1 as number_visitors, job_title@0 as job_title]
  HashAggregateExec: mode=FinalPartitioned, gby=[job_title@0 as job_title], aggr=[COUNT(UInt8(1))]
    CoalesceBatchesExec: target_batch_size=4096
      RepartitionExec: partitioning=Hash([Column { name: "job_title", index: 0 }], 16)
        CoalesceBatchesExec: target_batch_size=4096
          FilterExec: city@1 = San Francisco
            RepartitionExec: partitioning=RoundRobinBatch(16)
              CsvExec: files=[./data/visitors.csv], has_header=true, limit=None, projection=[job_title, city]
```

Getting results

Return record batches (or write results)



DataFusion Features

- Mostly complete SQL implementation (aggregates, joins, window functions, etc)
- DataFrame API (Python, Rust)
- High performance vectorized, native, safe, multi-threaded execution
- Common file formats: Parquet, CSV, JSON, Avro
- Highly extensible / customizable
- Large, growing community driving project forward

SQL Support

https://arrow.apache.org/datafusion/user-guide/sql/sql_status.html#supported-sql

Projection (SELECT), Filtering (WHERE), Ordering (ORDER BY), Aggregation (GROUP BY)

Aggregation functions (COUNT, SUM, MIN, MAX, AVG, APPROX_PERCENTILE, etc)

Window functions (OVER .. ([ORDER BY ...] [PARTITION BY ..])

Set functions: UNION (ALL), INTERSECT (ALL), EXCEPT

Scalar functions: string, Date/time,... (basic)

Joins (INNER, LEFT, RIGHT, FULL OUTER, SEMI, ANTI)

Subqueries, Grouping Sets

Extensibility

Customize DataFusion to your needs

User Defined **Functions**

User Defined **Aggregates**

User Defined **Optimizer** passes

User Defined **LogicalPlan** nodes

User Defined **ExecutionPlan** nodes

User Defined **TableProvider**

User Defined **FileFormat**

User Defined **ObjectStore**

Systems Powered by DataFusion

Andrew

FLOCK

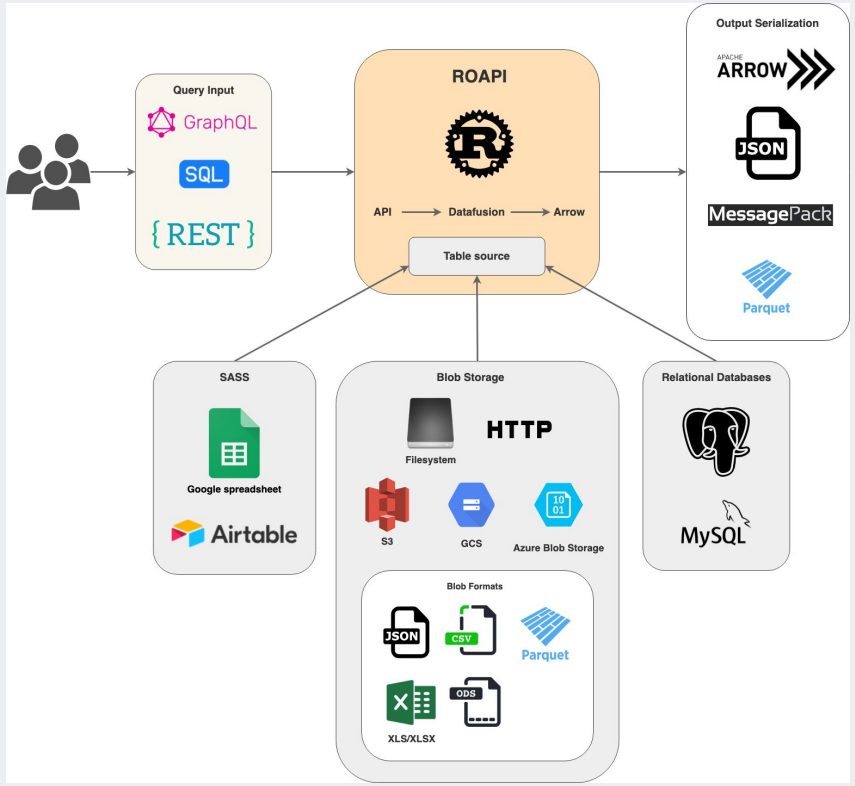
https://github.com/flock-lab/flock

- Overview:
  - Low-Cost Streaming Query Engine on FaaS Platforms
  - Project from UMD Database Group, runs streaming queries on AWS Lambda (x86 and arm64/graviton2).
- Use of DataFusion
  - **SQL API:**
  - **DataFrame API:** To build plans
  - **Optimized native plan execution**

ROAPI

https://roapi.github.io/

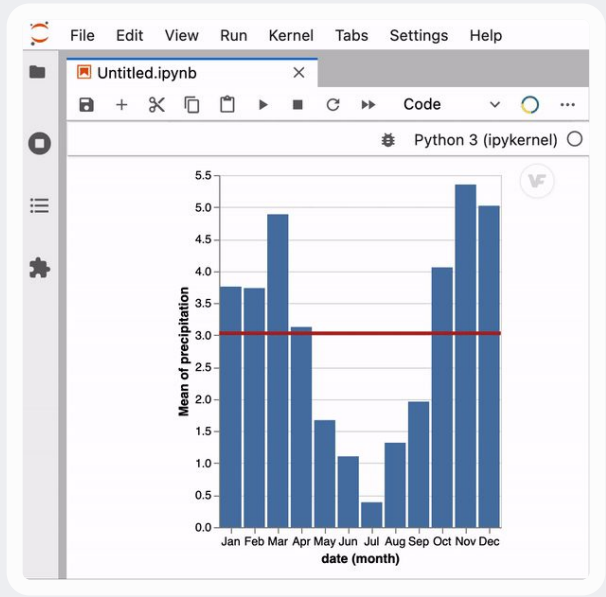
- Overview:
  - read-only APIs for static datasets without code
  - `columnq-cli`: run sql queries against CSV files
- Use of DataFusion
  - **SQL API:**
  - **DataFrame API:** (to build plans for GraphQL)
  - **File formats:** CSV, JSON, Parquet, Avro
  - **Optimized native plan execution**



VegaFusion

https://vegafusion.io/

- Overview:
  - Accelerates execution of (interactive) data visualizations
  - Compiles Vega data transforms into DataFusion query plans.
- Use of DataFusion:
  - **DataFrame API:** To build plans
  - **UDFs:** to implement some Vega expressions
  - **Optimized native plan execution**



Cube.js / Cube Store

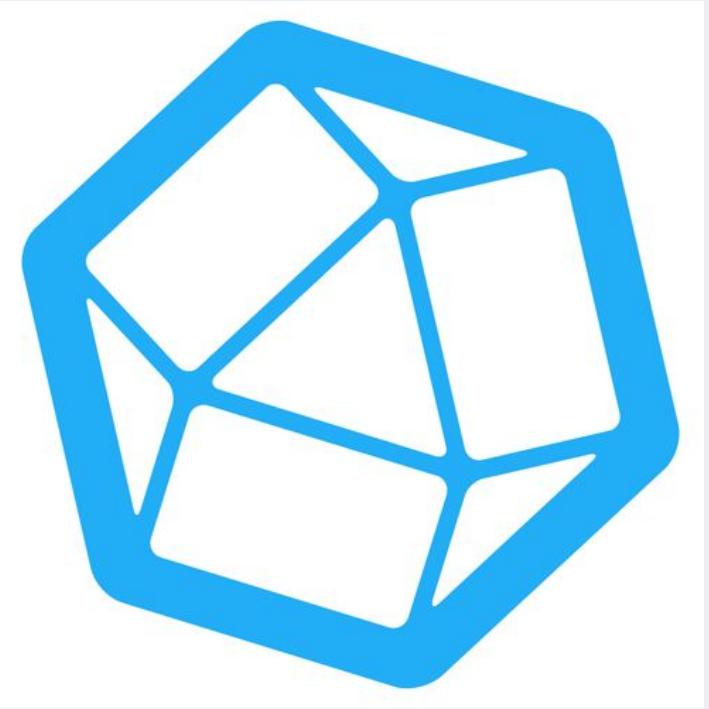
https://cube.dev/

- Overview:
  - Headless Business Intelligence
  - `cubestore` pre-aggregation storage layer
- Use of DataFusion (fork)
  - **SQL API** (with custom extensions)
  - **Custom Logical and Physical Operators**
  - **UDFs:** custom functions
  - **Optimized native plan execution**

InfluxDB IOx

https://github.com/influxdata/influxdb_iox

- Overview:
  - In-memory columnar store using object storage, future core of InfluxDB; support SQL, InfluxQL, and Flux
  - Query and data reorganization built with DataFusion
- Use of DataFusion:
  - **Table Provider:** Custom data sources
  - **SQL API**
  - **PlanBuilder API**: Plans for custom query language
  - **UD Logical and Execution Plans**
  - **UDFs**: to implement the precise semantics of influxRPC
  - **Optimized native plan execution**



Coralogix

https://coralogix.com/

Coralogix

- Overview:
  - Stateful streaming analytics with machine learning enables teams to monitor and visualize observability data in real-time before indexing
- Use of DataFusion:
  - **Table Provider**: custom data source
  - **User Defined Logical** and **Execution Plans**: to implement a custom query language
  - **User Defined ObjectStore**: for queries over data in object storage
  - **UDFs**: for working with semi-structured data
  - **Optimized native plan execution**

blaze-rs

https://github.com/blaze-init/blaze

- Overview:
  - High performance, low-cost native execution layer for Spark: execute the physical operators with Rust
  - Translates Spark Exec nodes into DataFusion Execution Plans
- Use of DataFusion
  - **Optimized native plan execution**
  - **HDFS Object Store Extension**

Ballista Distributed Compute

https://github.com/apache/arrow-ballista

- Overview:
  - Spark-like distributed Query Engine (part of Arrow Project)
  - Adds distributed execution to DataFusion plans
- Use of DataFusion:
  - **SQL API**
  - **DataFrame API**
  - **Optimized native plan execution**
  - **File formats:** CSV, JSON, Parquet, Avro

What's Next?

Daniël

Future Directions

- Embeddability
  - More regular releases to crates.io, more modularity
- Broader SQL features
  - Subqueries, more date/time functions, struct / array types
- Improved Performance
  - Query directly from Object Storage
  - More state of the art tech: JIT, NUMA aware scheduling, hybrid row/columnar exec
- Ecosystem integration
  - FlightSQL, Substrait.io
  - Databases
- GPU support

Come Join Us

We ❤️ Our Contributors

- Contributions at all levels are encouraged and welcomed.
- Learn Rust!
- Learn Database Internals!
- Have a great time with a welcoming community!

More details:

https://arrow.apache.org/datafusion/community/communication.html

DATA+AI
SUMMIT 2022

Thank you

arrow.apache.org/datafusion

github.com/apache/arrow-datafusion

Andrew Lamb
Staff Engineer, InfluxData
Apache Arrow PMC

Daniël Heres
Data Engineer, GoDataDriven
Apache Arrow PMC

Backup Slides

Thank You!

arrow.apache.org/datafusion

github.com/apache/arrow-datafusion

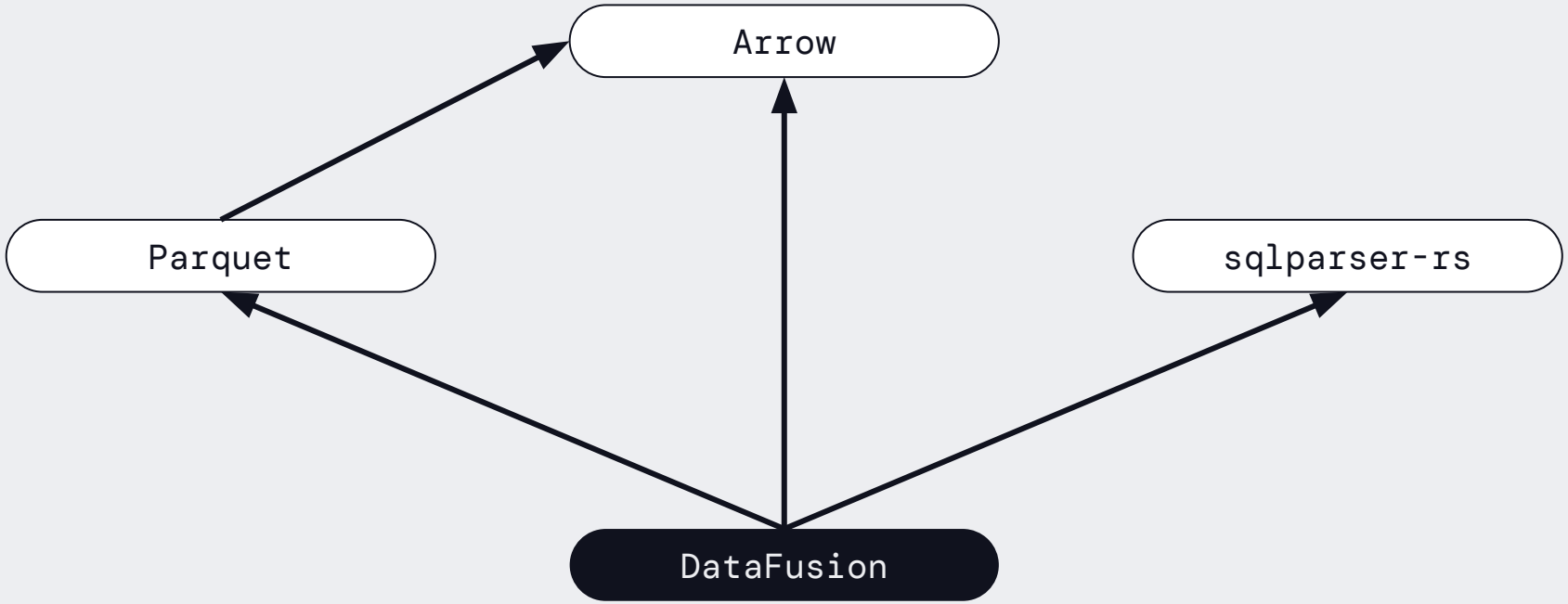
Andrew Lamb
Staff Engineer,
InfluxData

Apache Arrow
PMC

Daniël Heres
Data Engineer,
GoDataDriven

Apache Arrow
PMC

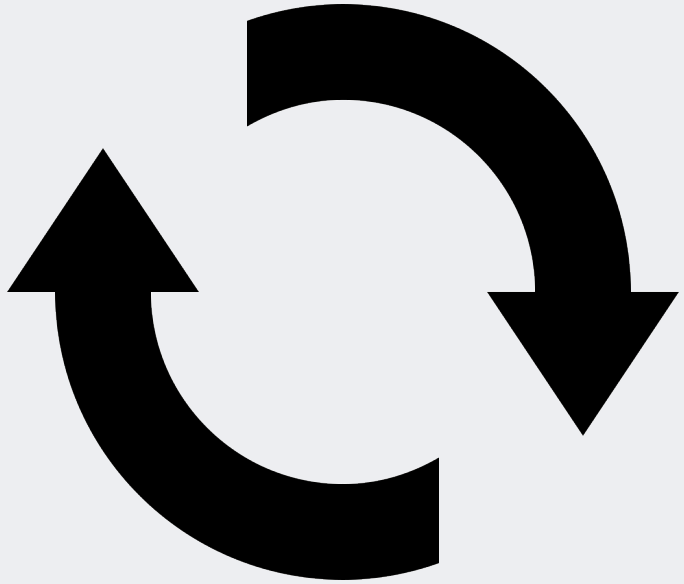
DataFusion / Arrow / Parquet



A Virtuous Cycle

Increased Use of Drives Increased Contribution

Increased **use** of
open source systems



Increased **capacity**
for maintenance and
contribution

DataFusion, and Apache Arrow are key open source technologies for building interoperable open source systems

delta-rs

https://github.com/delta-io/delta-rs

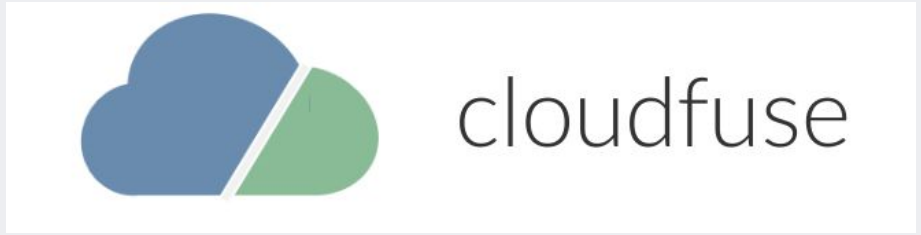
- Overview:
  - Native Delta Lake implementation in Rust
- Use of DataFusion
  - **Table Provider API**: allows other DataFusion users to read from Delta tables

DISCLAIMER: Not yet cleared / verified with project team

Cloudfuse Buzz

https://github.com/cloudfuse-io/buzz-rust

- Serverless cloud-based query engine
  - map using cloud functions (AWS Lambda)
  - aggregate using containers (AWS Fargate)
- Project (expected to be) continued from june



DISCLAIMER: Not yet cleared / verified with project team

dask-sql

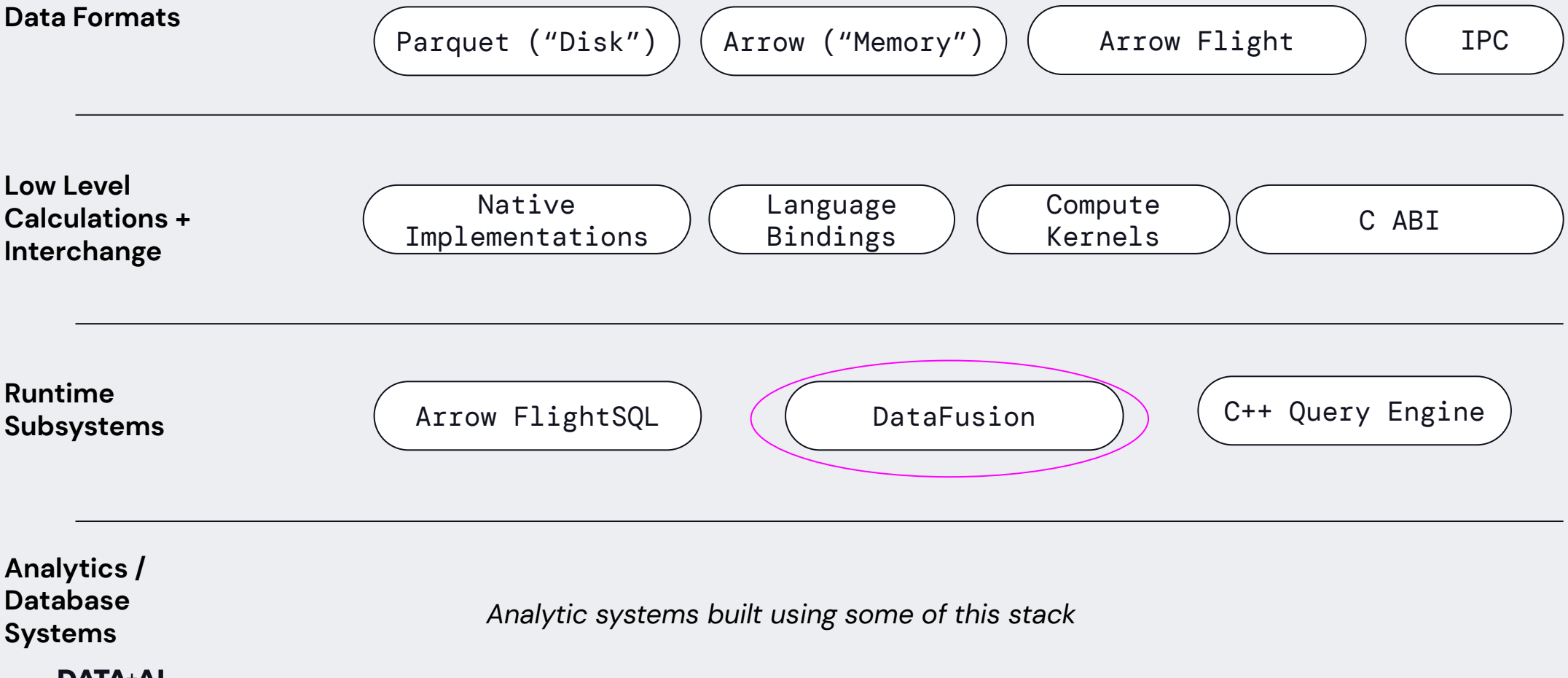
https://github.com/dask-contrib/dask-sql

- Overview:
  - TBD
-
- Use of DataFusion:
  - WIP https://github.com/dask-contrib/dask-sql/issues/474

DISCLAIMER: Not yet cleared / verified with project team

Apache Arrow Analytics Toolkit

Where does DataFusion fit?



Query Engines

What is it and why do you need one?

1. Add SQL or DataFrame interface to your application's data
2. Implement a custom query language / DSL
3. Implement a new data analytic system
4. Implement a new database system (natch)

Maps Desired Computations: SQL and DataFrame (ala Pandas)

To Efficient Calculation: projection pushdown, filter pushdown, joins, expression simplification, parallelization, etc

datafusion-python

https://github.com/datafusion-contrib/datafusion-python

- Overview:
  - Python dataframe library (modeled after pyspark)
- Use of DataFusion
  - **SQL API**
  - **DataFrame API**
  - **File formats:** CSV, JSON, Parquet, Avro
  - **Optimized native plan execution**

DISCLAIMER: Not yet cleared / verified with project team

Common Themes

Come for the performance, stay for the features (?)

Native execution

Native (non JVM) of Spark/Spark like behavior

SQL interface

Projects are leveraging properties of Rustlang

SQL / DataFrame API

Better, Faster, Cheaper

Better

Faster

Cheaper

The DataFusion Query Engine is part of the commoditization of advanced analytic database technologies

Transform analytic systems over the next decade

Andrew's Notes

Proposal: Data + AI Summit talk

Desired Takeaways:

1. If you need a query engine (in Rust?), you should use DataFusion

Thesis: DataFusion is part of a larger trend (spearheaded by Apache Arrow) in the commoditization of analytic database technologies, which will lead to many faster / cheaper / better analytic systems over the next decade

Other decks for inspiration:

DataFusion: An Embeddable Query Engine Written in Rust

xA Rusty Introduction to Apache Arrow and how it Applies to a Time Series Database

2021-04-20: Apache Arrow and its Impact on the Database industry.pptx

Instructions: Read me!

Getting started with our slide template

When using this template, create your new slides at the very top of the slide order, above this slide. Explore the advice and example slides below to find useful layouts and graphics to pull into your design. **When your slide deck is complete, delete this slide and every slide below it.**

Presentation best practices

Less is more

Clarity over density

Don't try to cram everything onto a limited number of slides. More slides with less text per slide is easier to digest.

Make it scannable

Use text hierarchy to create order and keep your content scannable. No walls of text! Try to keep headlines short.

Get creative

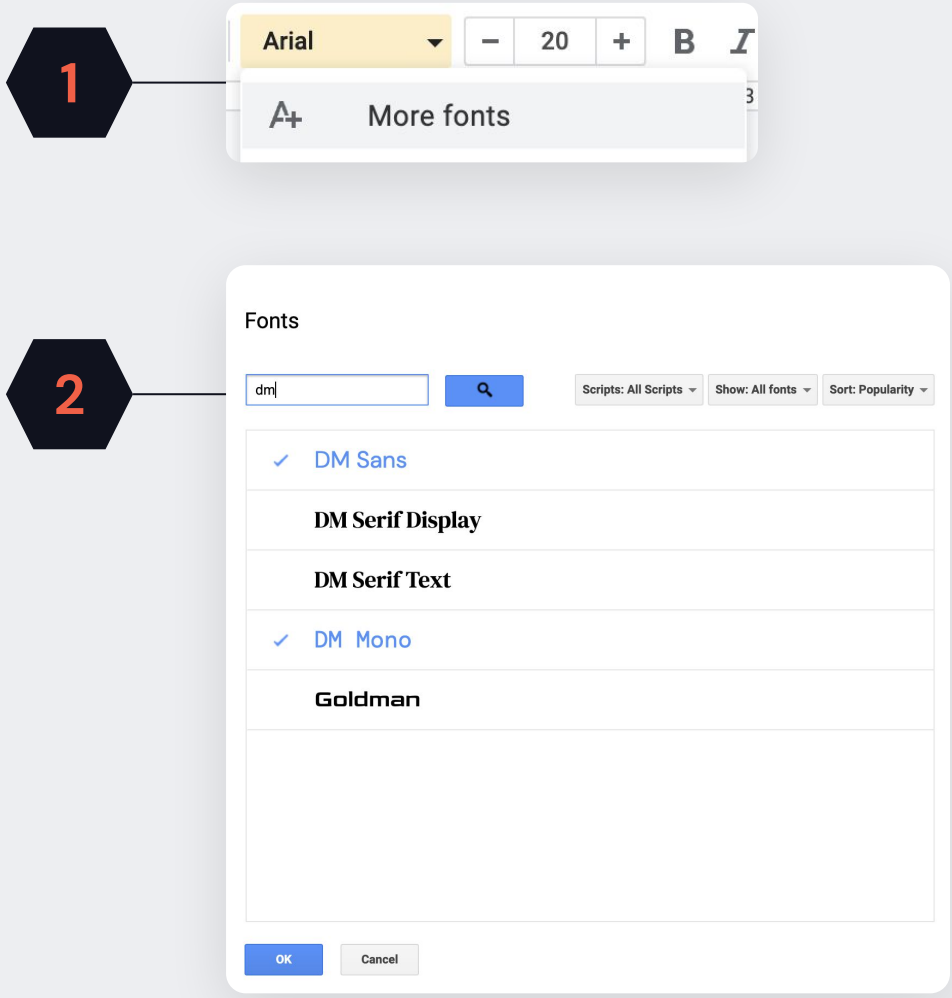
There are great baseline slides in this template, but it may not have everything you need. Don't be afraid to craft your own layouts! Just pay attention to the font and grid guidelines, and take advantage of starter shapes.

Font Guidance

Font selection

All text in our slide decks should use one of two available event brand fonts: **DM Sans** or **DM Mono**.

If you do not see these fonts in your font selection menu, they can be added by selecting “More fonts” and searching for “dm.” Click on DM Sans and DM Mono, then hit OK.



Font Guidance (Cont.)

Font sizing

Using consistent type sizing is a good way to help your slides feel uniform. When selecting type sizes, try to stick to multiple of 8, with the exceptions of 12 and 20 as in-betweens.

64 DATA+AI Summ

56 DATA+AI Summi

40 DATA+AI Summit

32 DATA+AI Summit

24 DATA+AI Summit

20 DATA+AI Summit

16 DATA+AI Summit

12 DATA+AI Summit

Grid Guidance

Keep it orderly

Your presentation template has a 12 column grid to help you organize the elements on your slides. When laying out objects, consider using the grid to help.

Toggle the grid visibility by navigating to ***View > Guides > Show Guides***.

Color Guidance

Keep it on brand

When customizing charts or adding other visual elements, do your best to stay within our defined event color palette. This will ensure that all your content looks great together and doesn't clash with the slide template design.

Always use black text when placing content over a colored background. The only exception is when using a black background. Any color text is acceptable on black.

10121E	00B6E0	85DDB5	F16047
EDEEF1	8FDDEF	AFE9CF	F3A89B

Example Slides

DATA+AI SUMMIT 2022

Choose Your Title Slide

Eighteen colorful title slide options with varying shapes

Add your Name

Add your title, company

ORGANIZED BY databricks

DATA+AI
SUMMIT 2022

Choose Your Title Slide

Eighteen colorful title slide options with varying shapes

Add your Name
Add your title, company

ORGANIZED BY databricks

DATA+AI SUMMIT 2022

Choose Your Title Slide

Eighteen colorful title slide options with varying shapes

Add your Name

Add your title, company

ORGANIZED BY databricks

Basic Content Slide

Your all-purpose zone

Use this slide as a starting point for crafting your own layouts, or for simple text slides.

Activate Dark Mode

Mix in black slides to add contrast and variety

Or make your whole presentation dark!

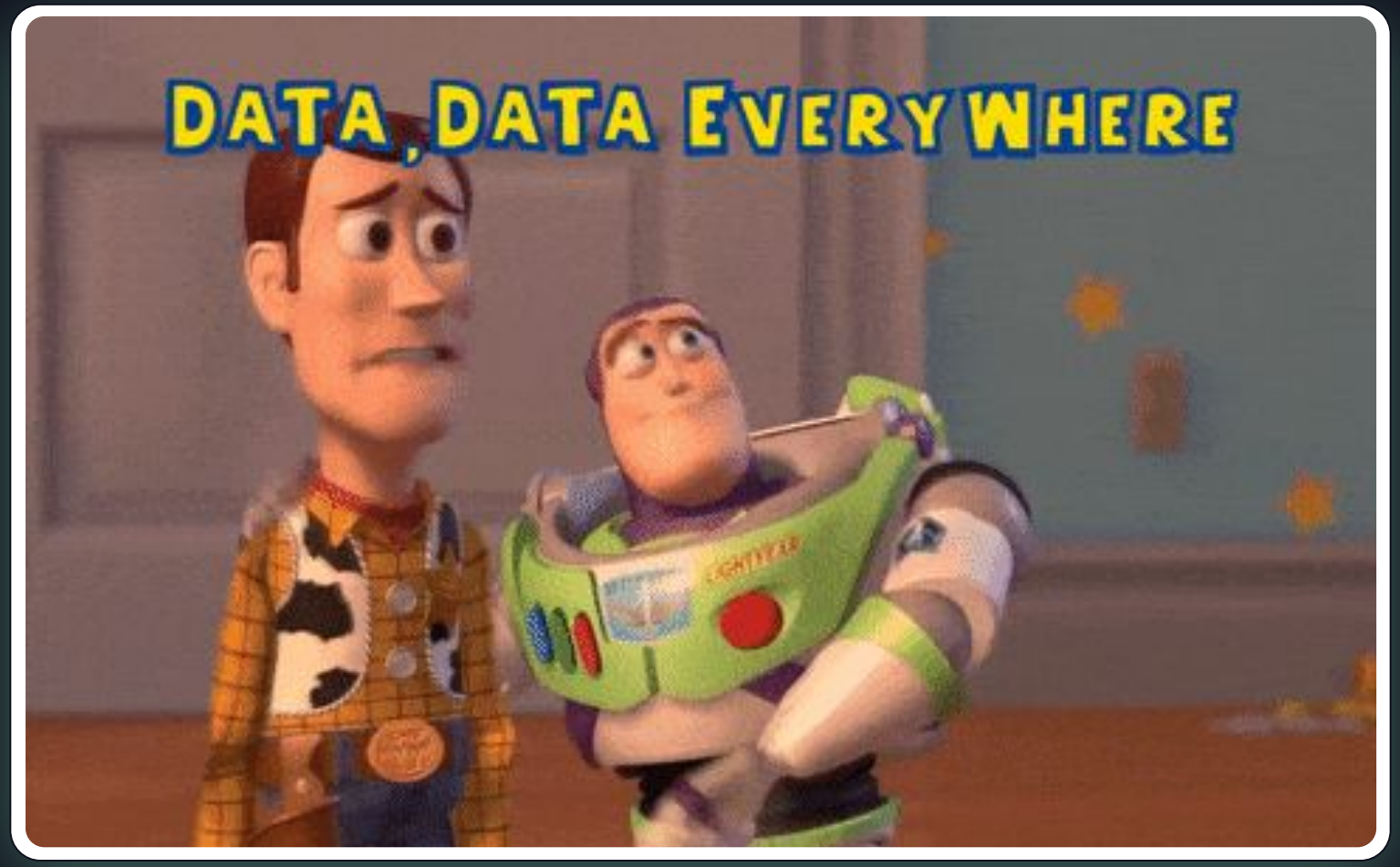
Insert your charts or images

Take advantage of the content panels

Insert Image by URL

If you want to insert a gif or other image from the web, simply navigate to *Insert > Image > by URL*.

Crop and resize your image to fit within content panels, if you're feeling fancy.



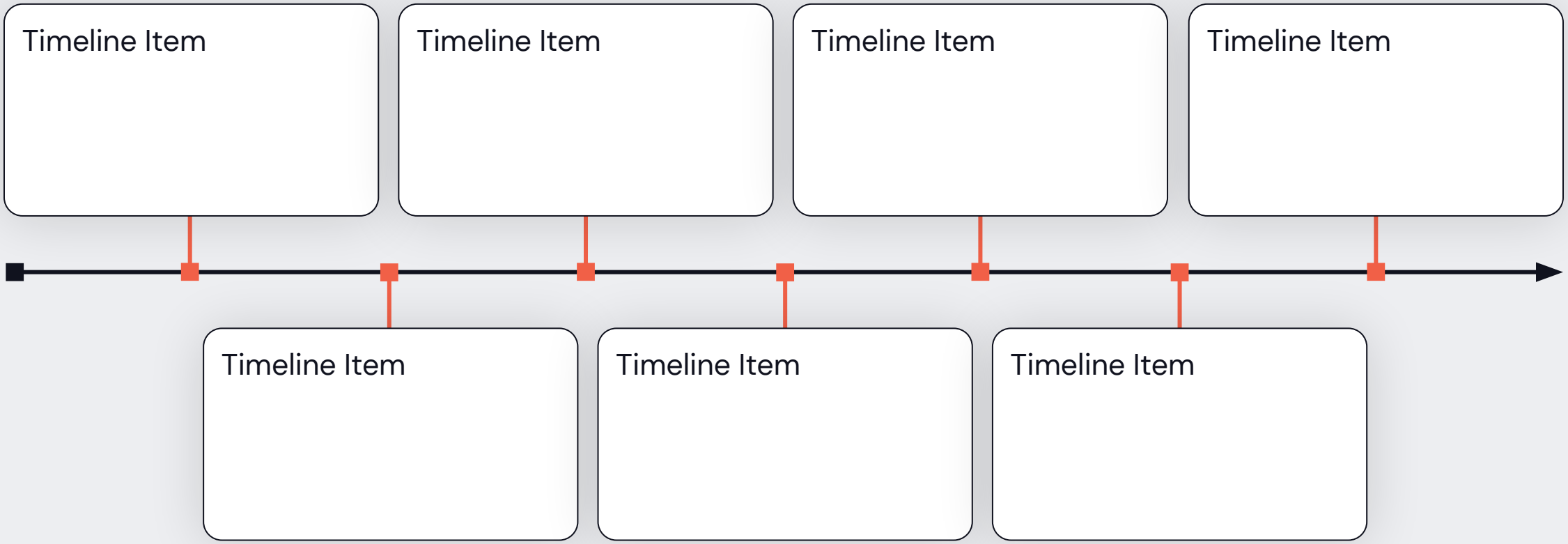
“With just a few adjustments to text size and alignment, you can use the basic content slide for other types of content such as quotes.”

Andrew Pons
Slide Designer

	Column A	Column B	Column C	Column D	Column E	Column F
Row A	You can create simple tables to help organize information.					
Row B						
Row C						
Row D						
Row E						
Row F						
Row G						
Row H						

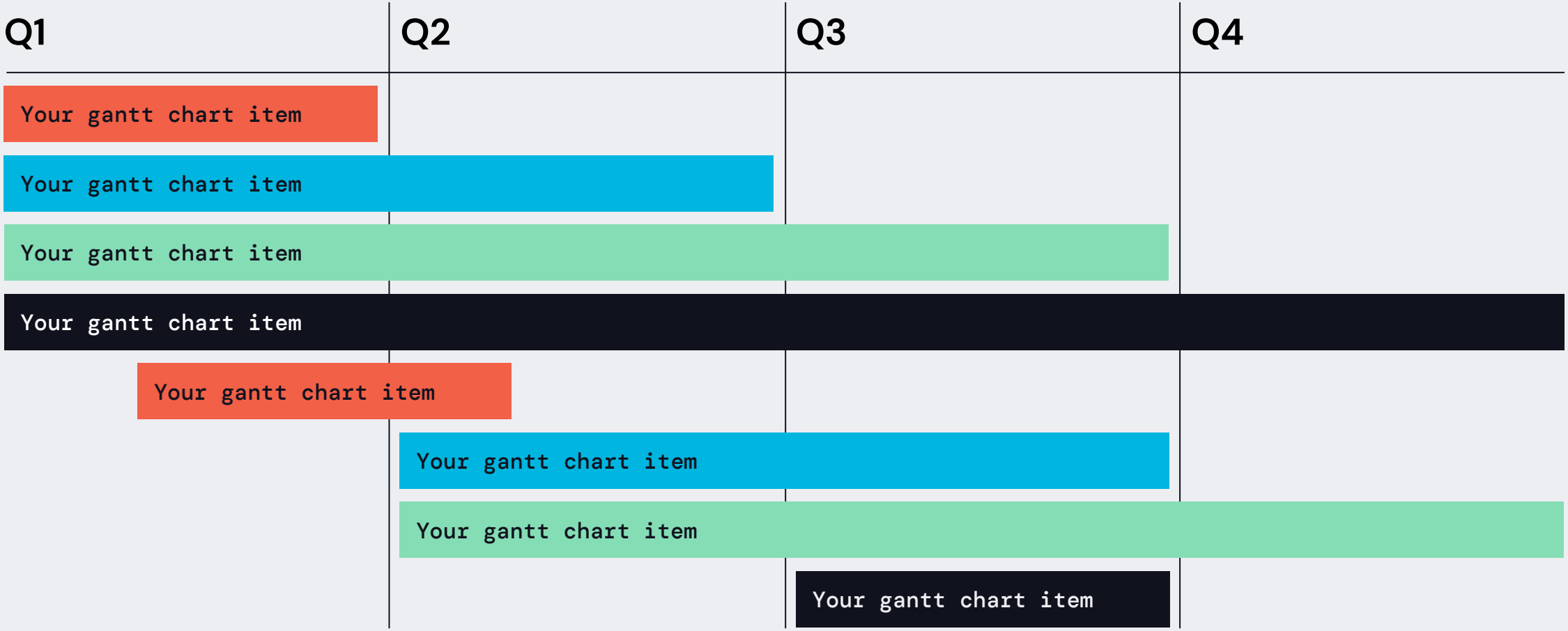
Timeline Style One

Your subtitle here



Timeline Style Two

Your subtitle here



Single Column

Content Tile

Multi-purpose

Use this panel for content, images, diagrams, or whatever else you want to include. You can use the line tool to divide this panel into multiple sections if you want.

Two Column

Content Tile

Multi-purpose

Use these slides for comparing two topics or just for splitting your content into multiple pieces.

Multi-purpose

Use these slides for comparing two topics or just for splitting your content into multiple pieces.

Three Column

Column 1

Column 2

Column 3

Four Column

Column 1	Column 2	Column 3	Column 4

Half Panel

Right aligned

Open Content

This space is great for supporting text that compliments whatever content is inside the panel.

Panel Content

This space can be for text content, images, diagrams, or whatever you need

Half Panel

Left aligned

Panel Content

This space can be for text content, images, diagrams, or whatever you need

Open Content

This space is great for supporting text that compliments whatever content is inside the panel.

⅔ Panel

Right aligned

Open Content

This space is great for supporting text that compliments whatever content is inside the panel.

Panel Content

This space can be for text content, images, diagrams, or whatever you need

⅔ Panel

Left aligned

Panel Content

This space can be for text content, images, diagrams, or whatever you need

Open Content

This space is great for supporting text that compliments whatever content is inside the panel.

Code Display

Paste snippets

```
select
    count (*),
    age
from
    visitors
where location="SanFrancisco"
group by job_title
```

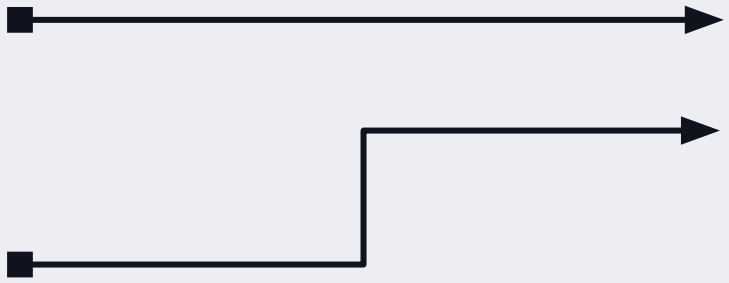
Use breaker slides to divide your deck into sections

Use breaker slides to divide your deck into sections

Starter Shapes

Copy and paste these wherever you need them

Floating panel for text or graphics

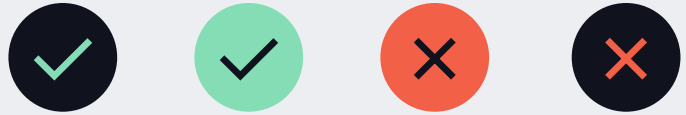


Medium pill label

Medium pill label

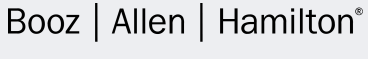
SMALL PILL LABEL

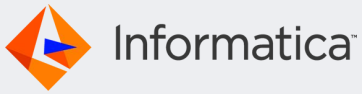
SMALL PILL LABEL



Logos

Partners and cloud platforms





Logos

Open source projects