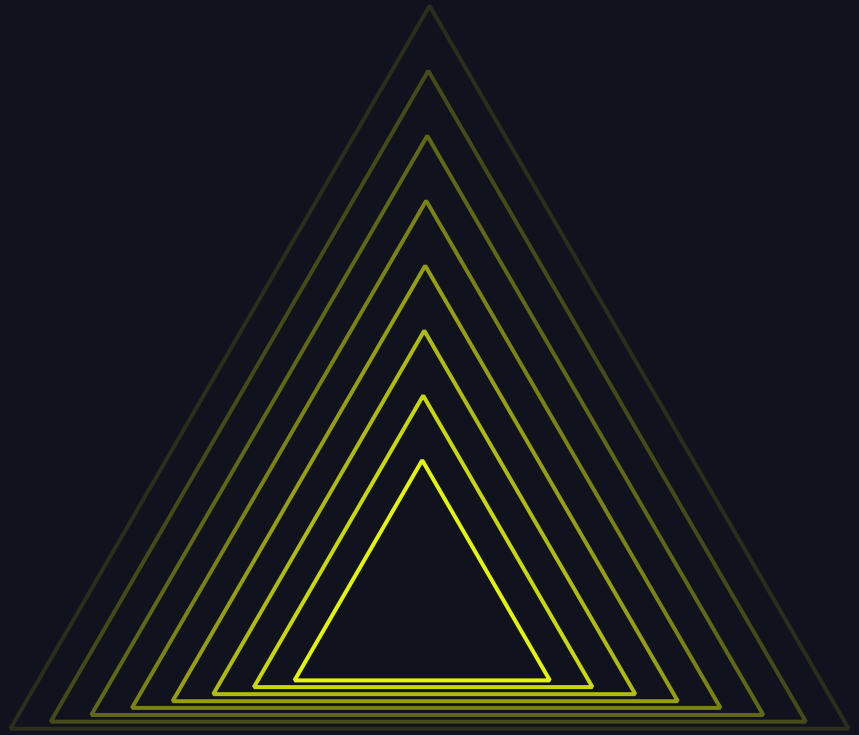
DATA+AI SUMMIT BY databricks

# DATA UTILITY ARCHITECTURE



Eric Feunekes
Date

# WHAT IS THE DATA UTILITY ARCHITECTURE

A comprehensive framework prioritizing data to power enterprise operations

- **Data-First**: Prioritizes data as a first-class citizen in enterprise architecture.
- **Built on the Lakehouse**: Store raw data and surface it in data products for any purpose.
- **Unified Data Planes**: Combines analytical and operational data planes.
- **Expands into a Mesh**: Ensures easy access to data across the organization while keeping it closest to those who know it best.
- **Scalability and Flexibility**: Designed to handle diverse data types and volumes, scaling both horizontally and vertically to meet evolving business needs.

# WHY SHOULD YOU USE THIS?

## Enhancing the value of your most important asset

### Grow the value of your data

- Data derives its value from how it is used
- The DAU treats data as a product to ensure its value is maximized for each consumer
- Combining sources creates even more value because it creates "new" types of data
- Flexibility to reuse data across use cases ensures the greatest return on your data

### Fast time to value

- DAU can be set up with a single data source feeding a single dashboard
- Choose the most valuable project, finish it, repeat.
- Eventually, your DAU will hold dozens of sources feeding dozens of applications.

### Future proof

- Lakehouse is agnostic to data type and products are agnostic to how it is used.
- Agnosticism means that future use cases are likely to be supported "out of the box".

# DATA FIRST

# APPLICATION FIRST... BAD

## The traditional enterprise architecture treats data as emissions to be captured

- "There's an app for that" mentality to solving problems
- Every app (micro or monolith) gets its own data infrastructure
- Enterprise data architecture is often an afterthought
- Once complete, app data is captured in data pipelines and shuttled to the current warehouse, data mart, lake, Lakehouse or other "thing"

# DATA FIRST... GOOD

## The data utility architecture focuses on data first so that it is always fit for purpose

- Enterprise data architecture is the first class citizen
- Where does data need to go, what steps are needed to enrich, where does it need to end up?
- If the data needs to be accessed or modified by people, think about the best interface
- Create disposable products for each use case

# THE BASICS

# KEY CONCEPTS

## Building blocks for the Data Utility Architecture

### Medallion stratified Lakehouse

- Data is stored in three layers: raw (bronze), semantic (silver), and presentation (gold).
- Transformations are never applied in the raw layer.

### Event sourced data streams

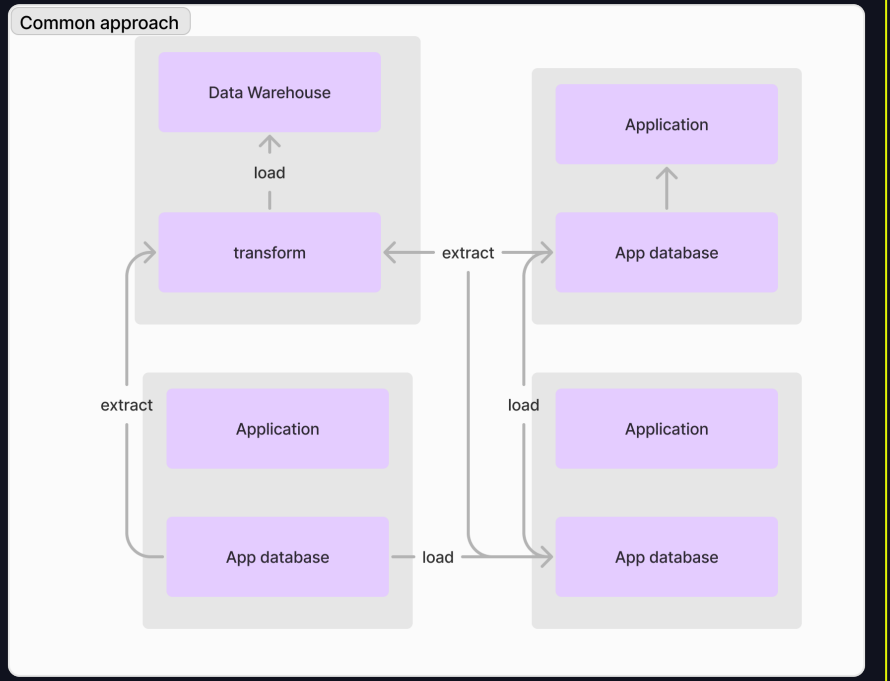
- Raw data should be event sourced
- Build streams to transform data through the layers
- Structured Streaming can be run in batch or streaming mode, making it cost effective and scalable

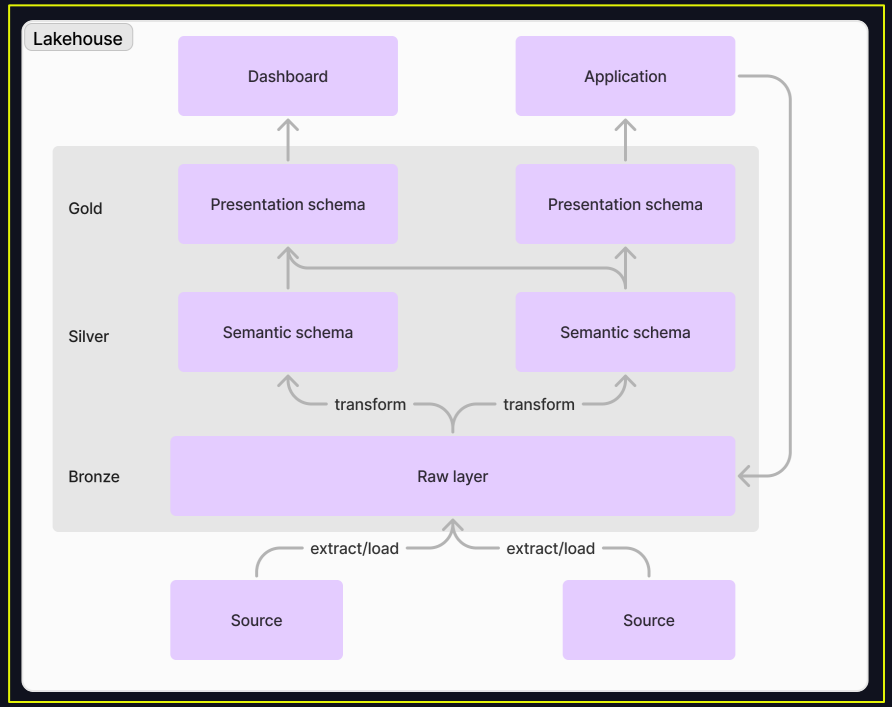
### Build data products

- Data "product" is flexible—whatever the customers need to succeed.
- Can include SLOs/SLAs on latency and quality
- Usually includes defined schema with guarantees on data types (e.g. no nulls).
- Provide as many types of connections as possible (e.g. ODBC, REST)

# MEDALLION STRATEFIED LAKEHOUSE

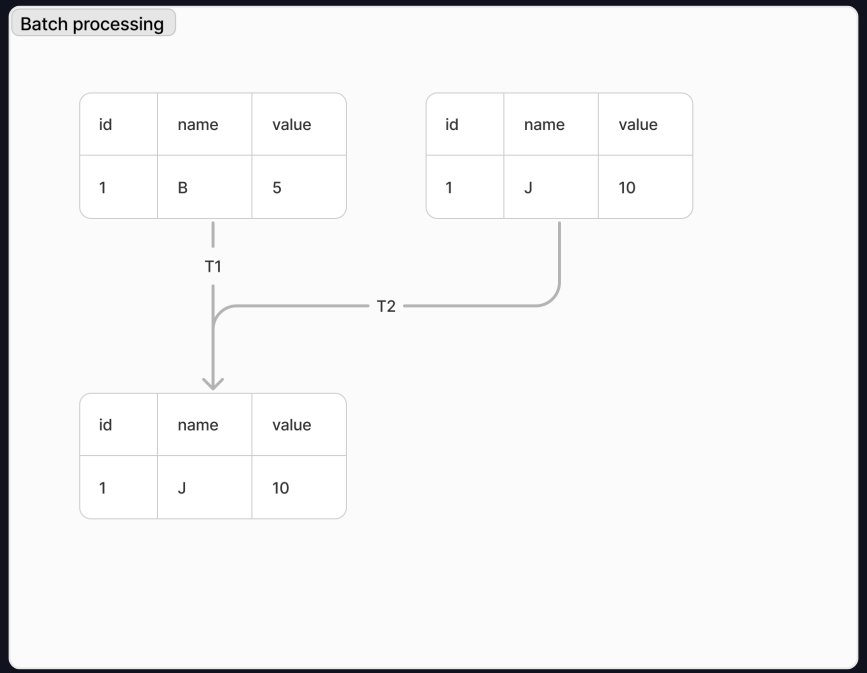
## Store raw data, transform it for a specific purpose, present it to your consumer

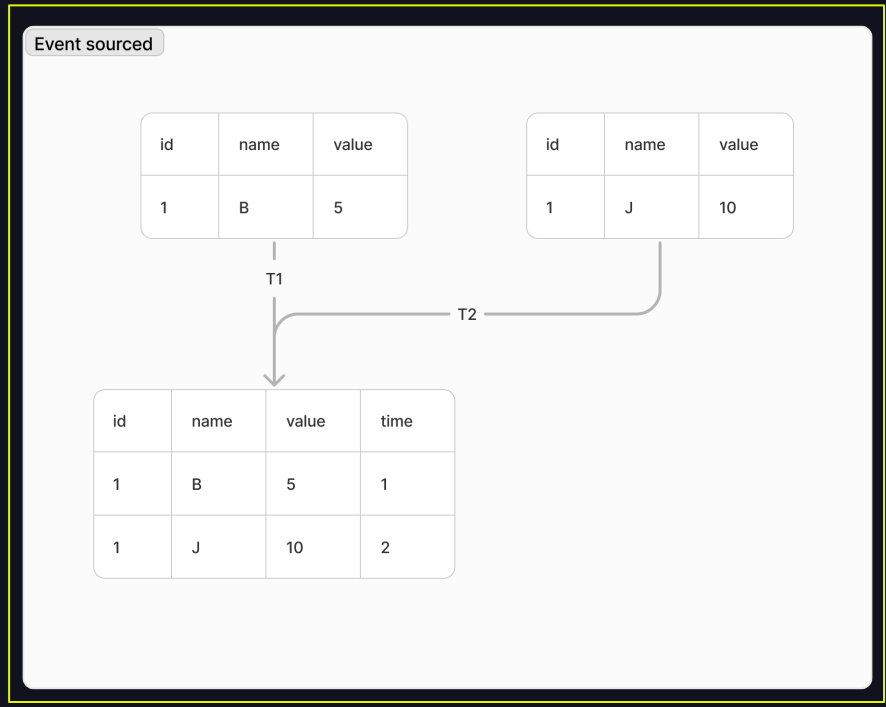


# EVENT SOURCED DATA STREAMS

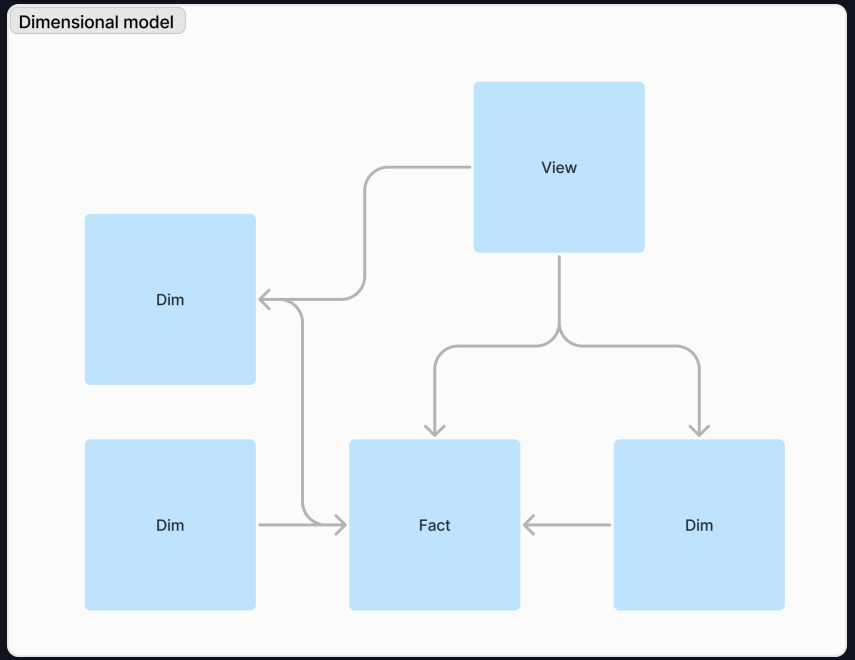
## Keeping a record of changes allows recreation of diverse models

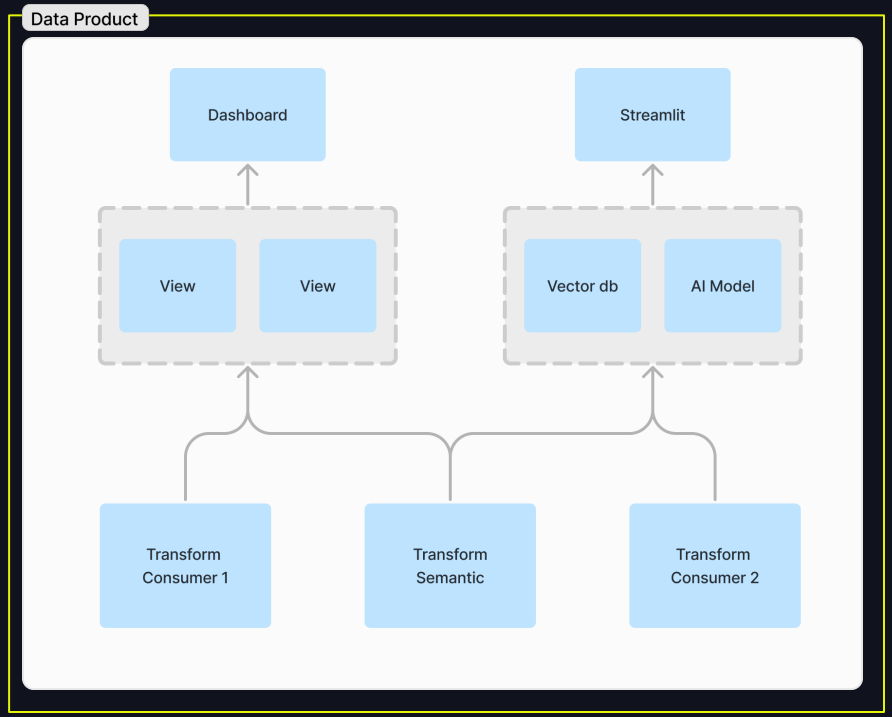


# DATA PRODUCTS

## Build products to solve targeted problems for your consumers



# OPTIONAL DATA MESH

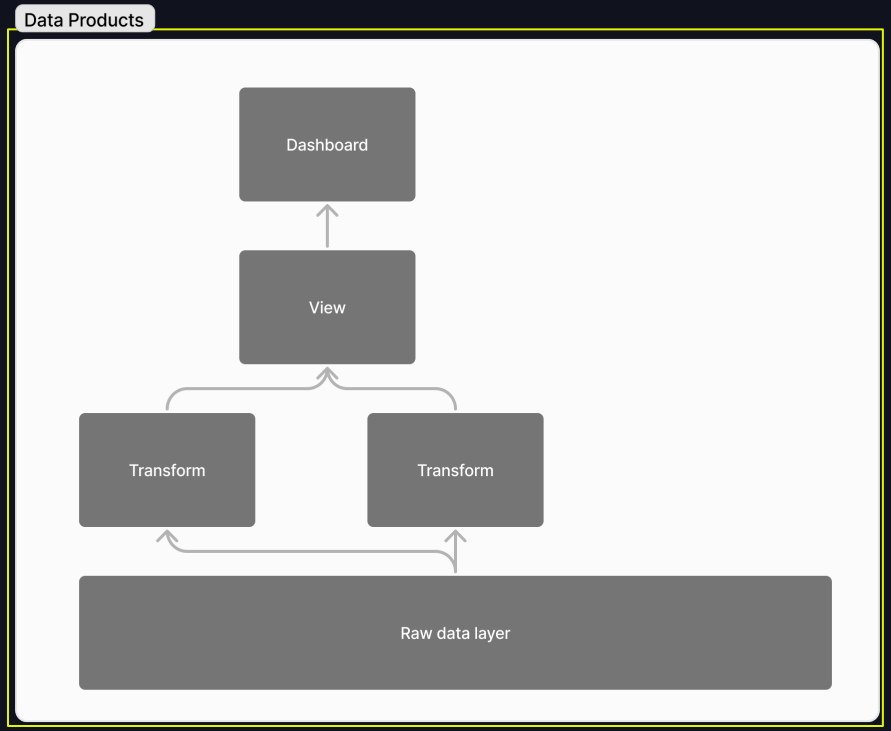
## Extend your Lakehouse across the enterprise

- The Lakehouse is very easy to extend into a data mesh using data products
- Catalogs are not required for discovery—leverage your data product experts
- Connections can be through open standards like Delta Sharing

# IMPLEMENTATION PATTERNS

# CREATE YOUR FIRST DATA PRODUCT

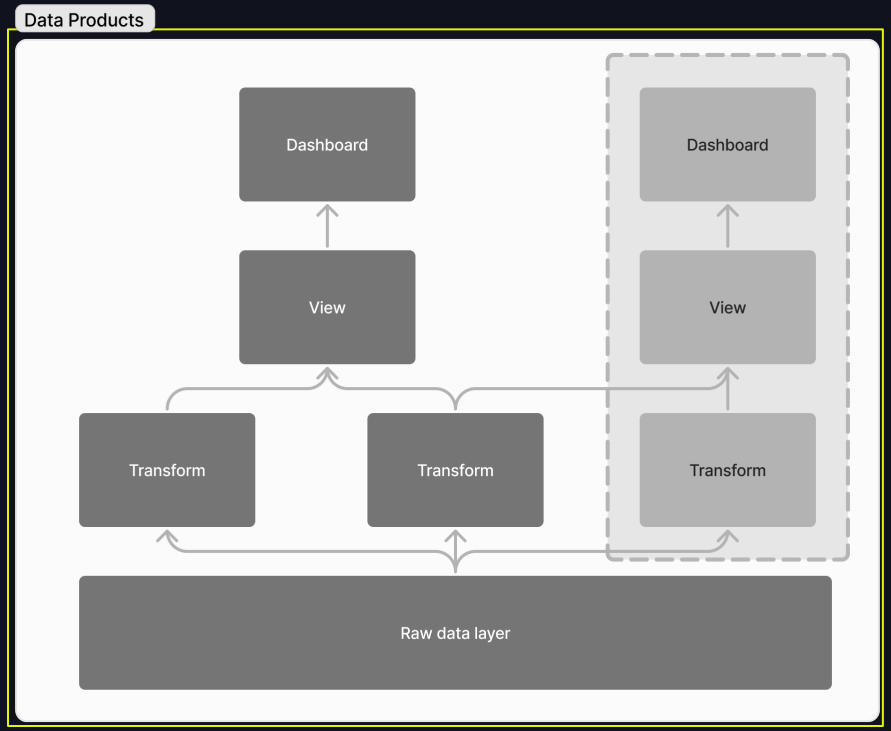
## Problem: how do you efficiently create a data product in this framework?




- From the raw layer, create streams of data for transformation/enhancement
- Typically one silver schema per consumer.
- May include additional schemas for repeated transformations
- Views on the semantic layer data are a good way to join data for consumers
- Streams can be made faster or slower based on demand

# CREATE YOUR FIRST DATA PRODUCT

## Problem: how do you efficiently create a data product in this framework?

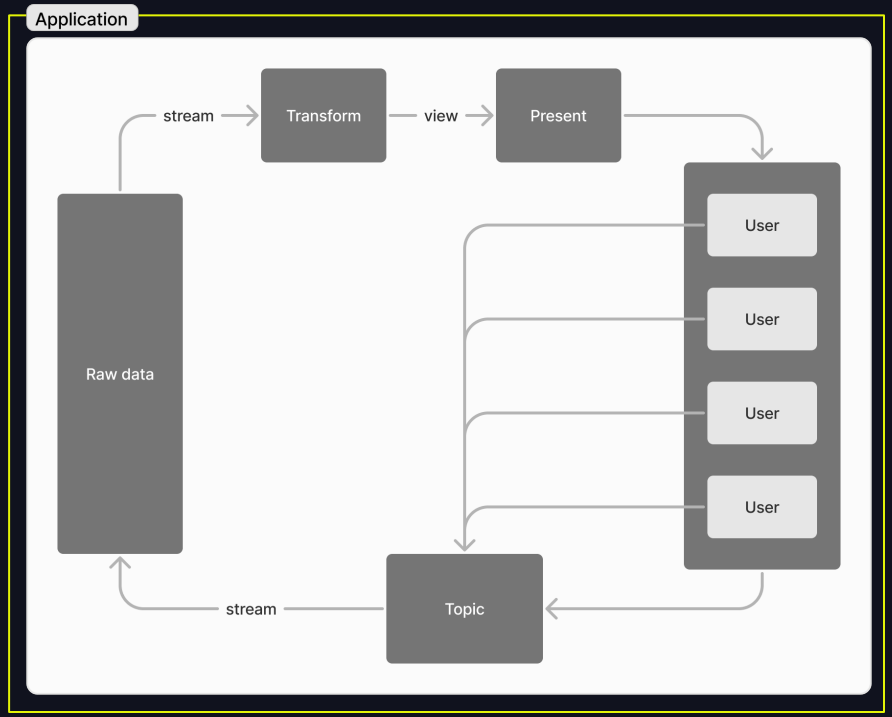



- From the raw layer, create streams of data for transformation/enhancement
- Typically one silver schema per consumer.
- May include additional schemas for repeated transformations
- Views on the semantic layer data are a good way to join data for consumers
- Streams can be made faster or slower based on demand

# POWER AN APPLICATION

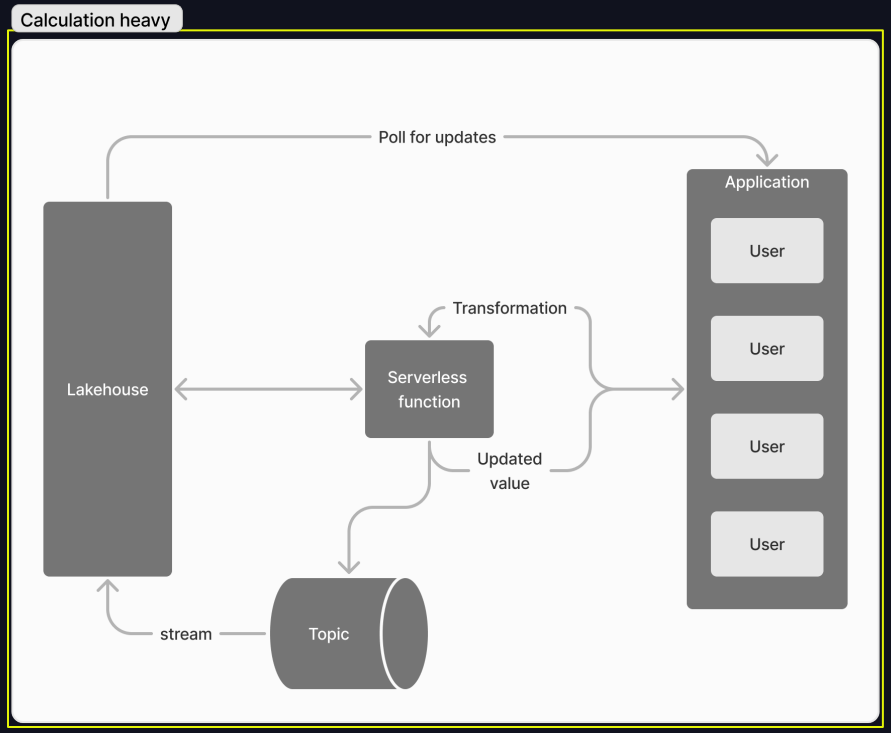
## Problem: build an application powered by a Lakehouse backend

- Command query response segregation (CQRS) means writing to one place and reading from another.
- Lean into eventual consistency
- Optimized for the use of streams and event data
- Ensures an immutable, ordered, log of events across any number of users
- Extremely fast to stand up and manage



# HEAVY CALCULATION APPLICATION

## Problem: Client application is running large calculations

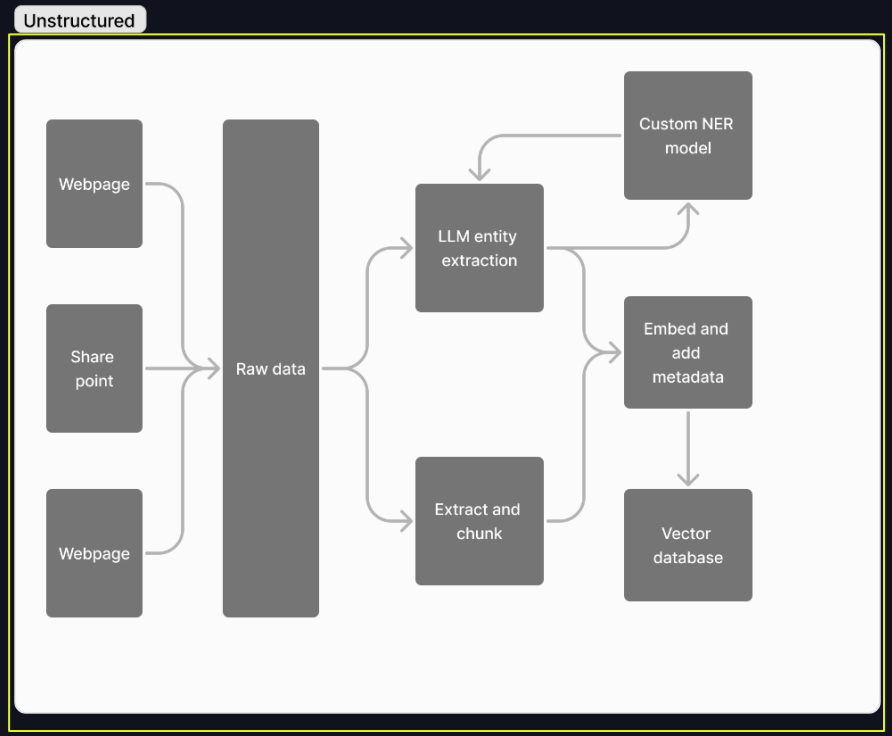



- Serve subset of data to client (lazy load)
- Serverless function uses Spark Connect to run transformations on a cluster in the Lakehouse
- Transformations are also passed to a topic, ingested in the Lakehouse, in order.
- Data is updated by polling the gold table.

# ADDING UNSTRUCTURED DATA

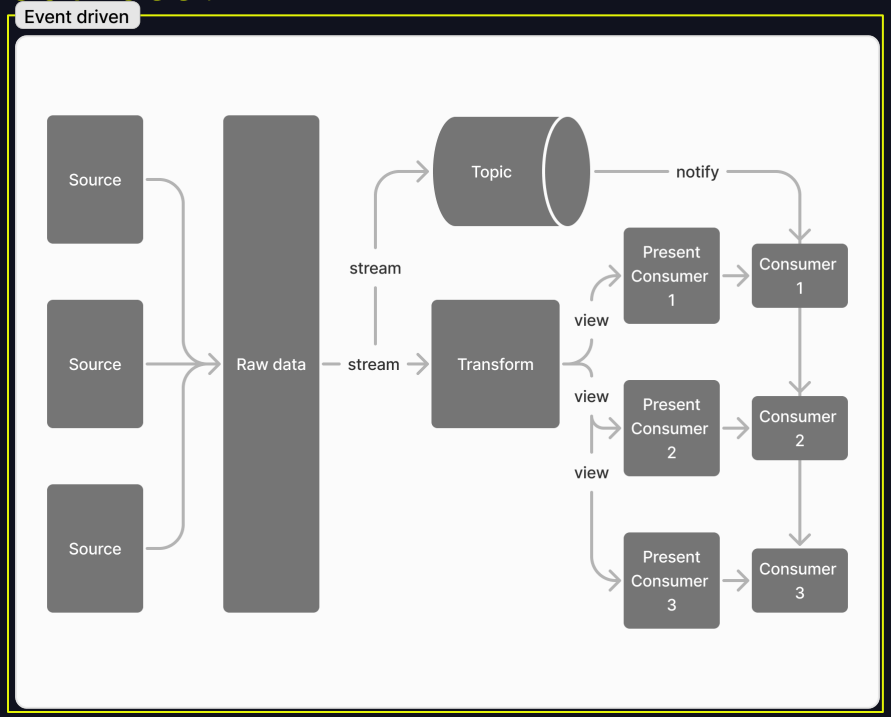
## Problem: how do you bring unstructured data into the DAU?

- There is no change to the infrastructure, just the techniques
- Introduce new tools like clustering, embedding, and other NLP approaches to enhance data
- Data products can be expanded to include model and vector databases
- Maintaining the medallion architecture ensures data



# EVENT DRIVEN UPDATES

## Problem: how do can a consumer get immediate updates from non-streaming sources?




- Create a streaming materialized table in the semantic layer
- Dynamic views with row level security can segregate data for consumers
- forEachBatch in the stream can update a partitioned event topic to tell consumers when their data is updated
- Consumers can implement event triggered query of their product