Architectures & DataOps

Big Data Management

Knowledge objectives

1. Explain the problem of a spaghetti architecture
2. Explain the need of the big data architecture
3. Identify the components of the big data architecture
4. Explain the need of each component
5. Explain the difference between various architectures
6. Justify the need of a Data Lake
7. Identify the difficulties of a Data Lake

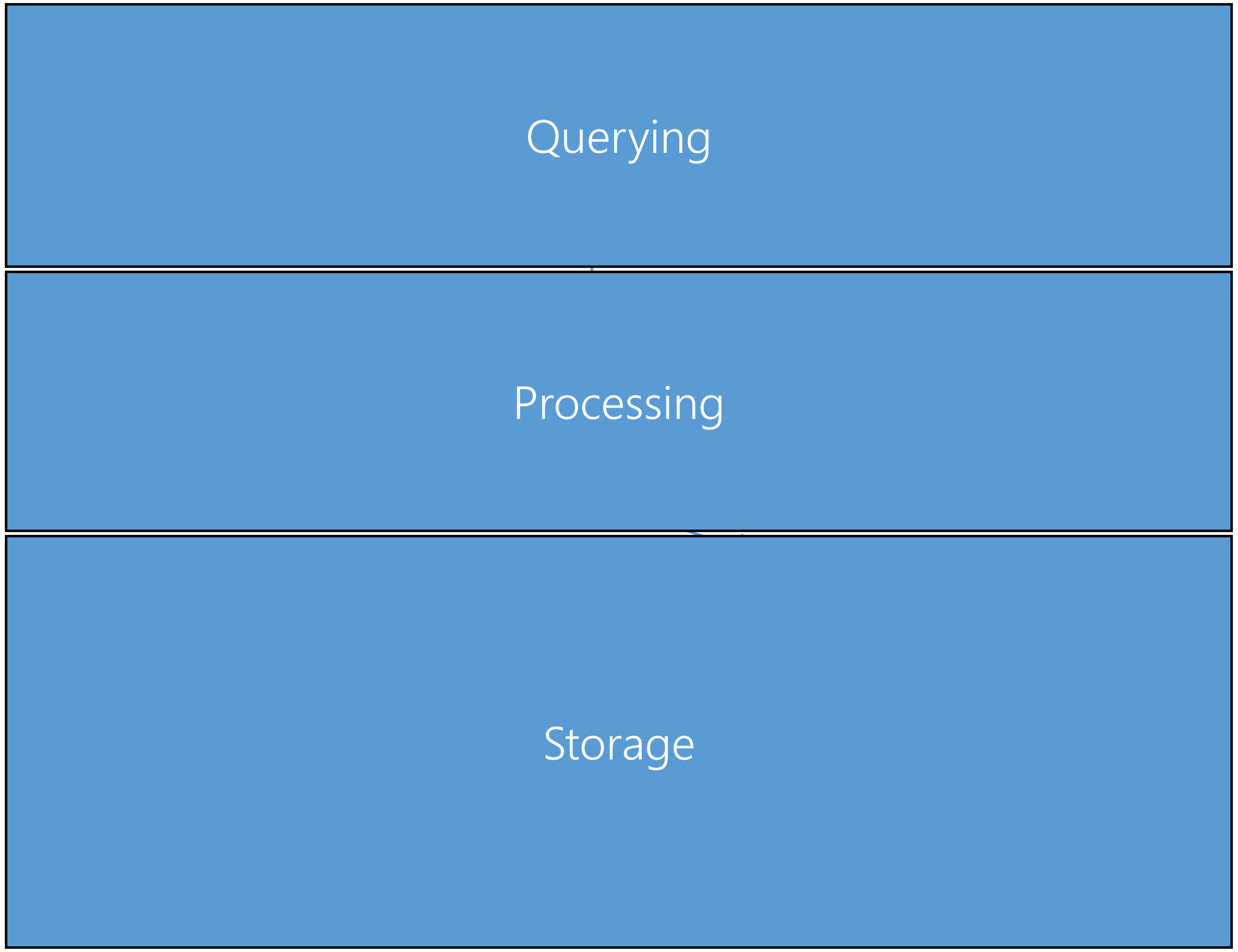
Application Objectives

1. Given a use case, define its software architecture

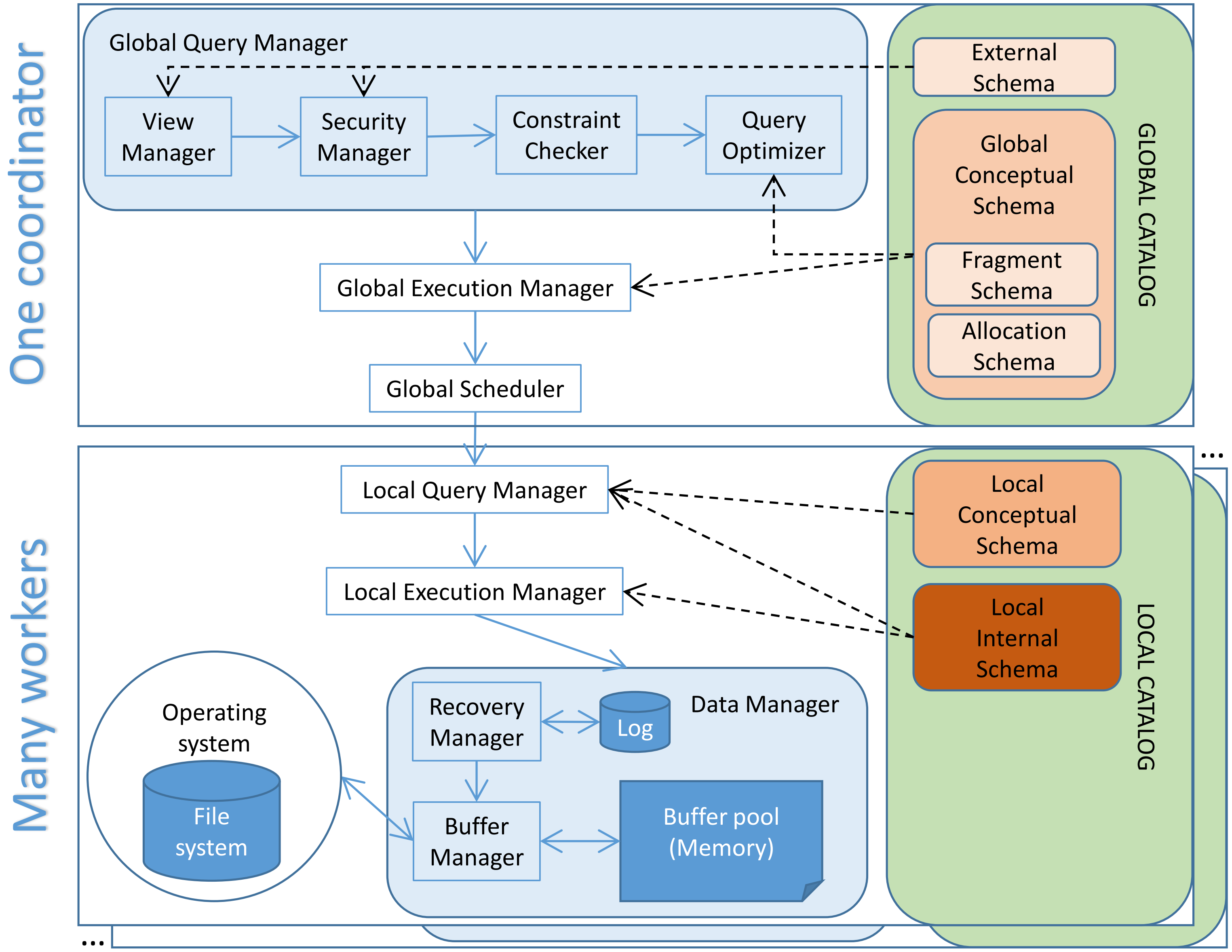
Architectures

Architectures & DataOps

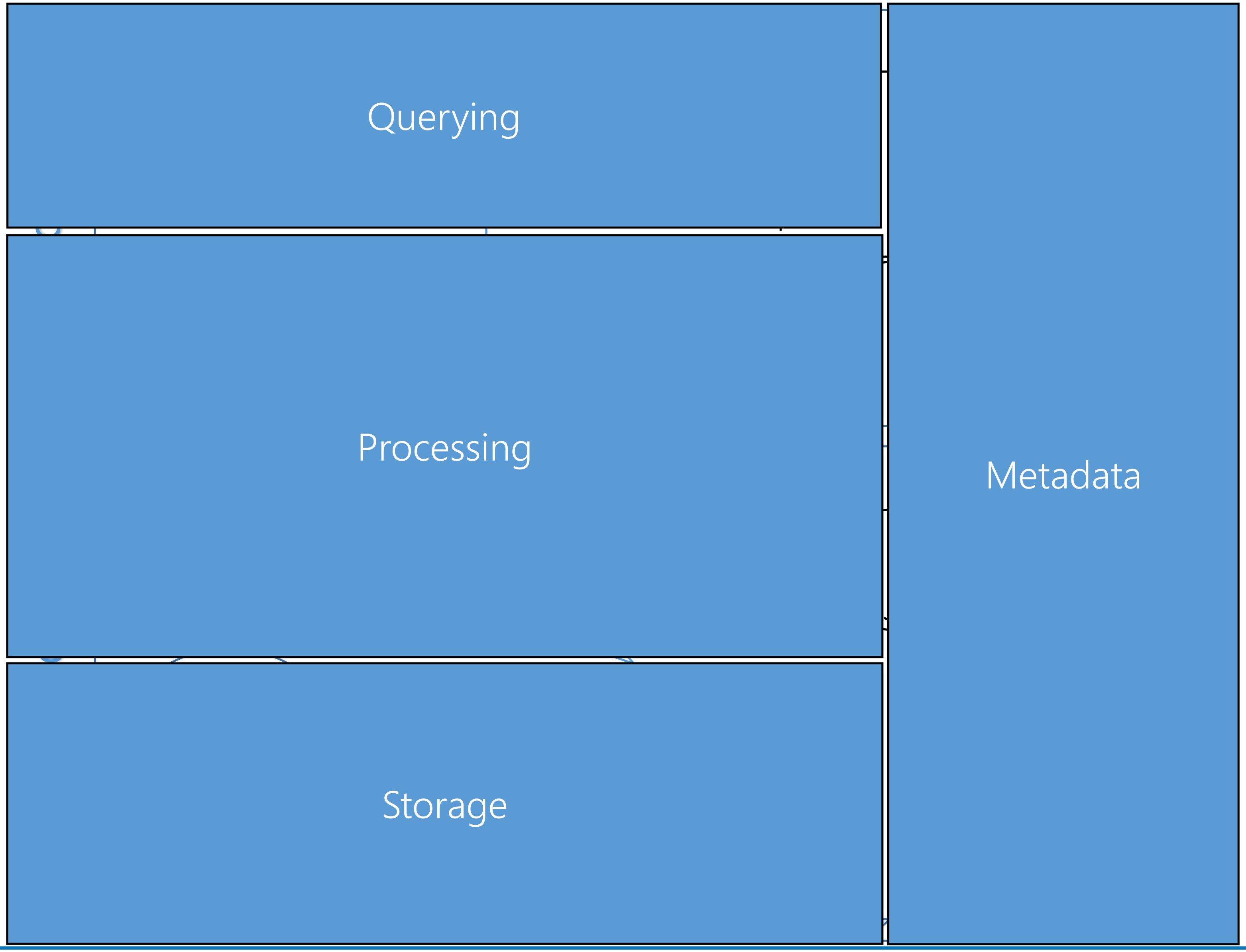
Traditional Architecture: Centralized DBMS



Distributed DBMS Architecture

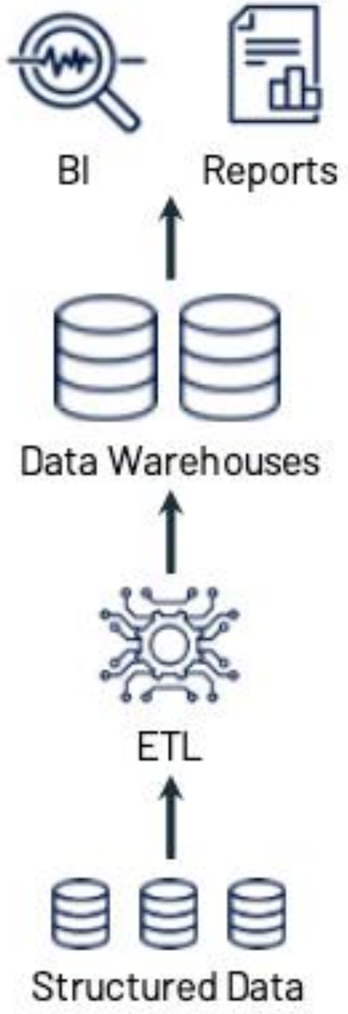


Distributed DBMS Architecture



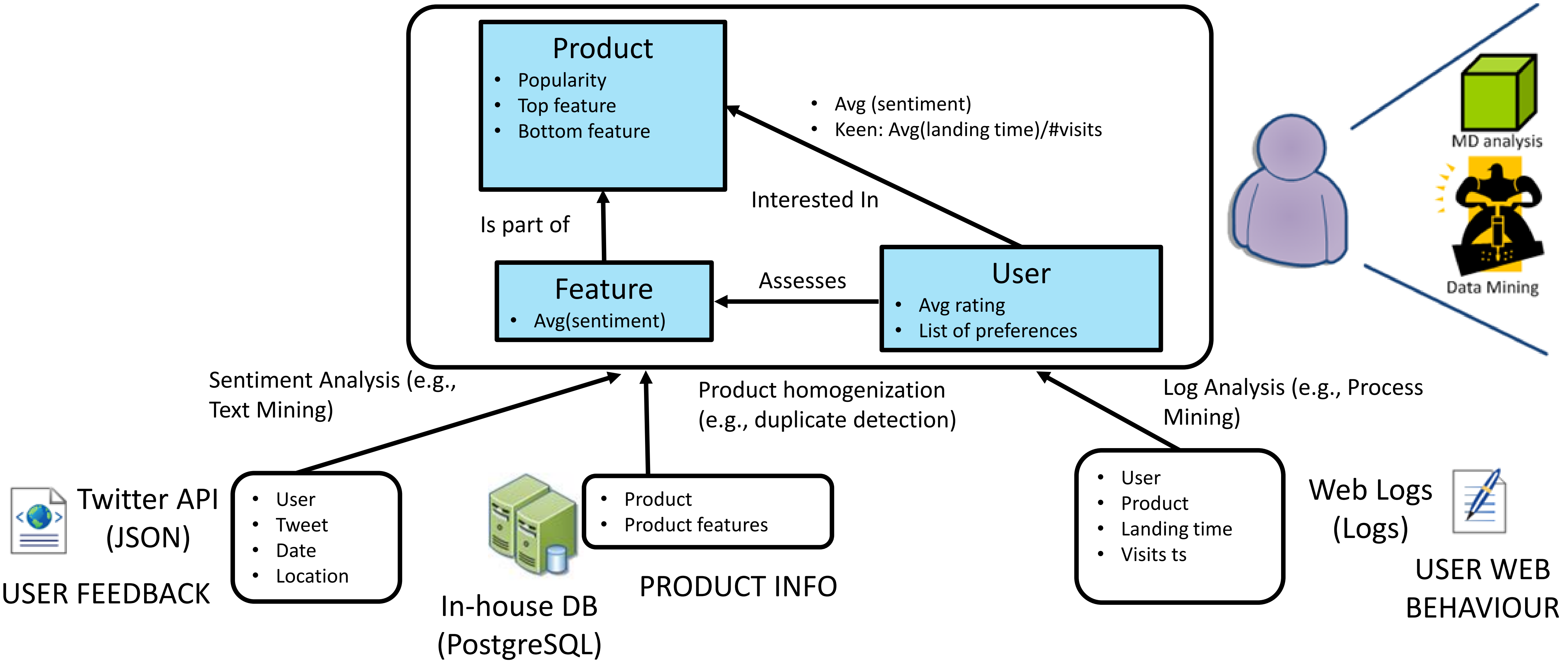
Data Warehouses

- Aid the business via analytical insights
  - Extract data from operational databases
  - Transform and load them into centralized DWs
    - Schema-on-write
    - The data model is optimized for BI operations
- Some challenges
  - Compute and storage in on-premise appliances
    - Requires to provision and pay for peak workloads / large datasets
  - Unstructured data
    - Video, audio, text documents
      - DWs cannot deal with these formats



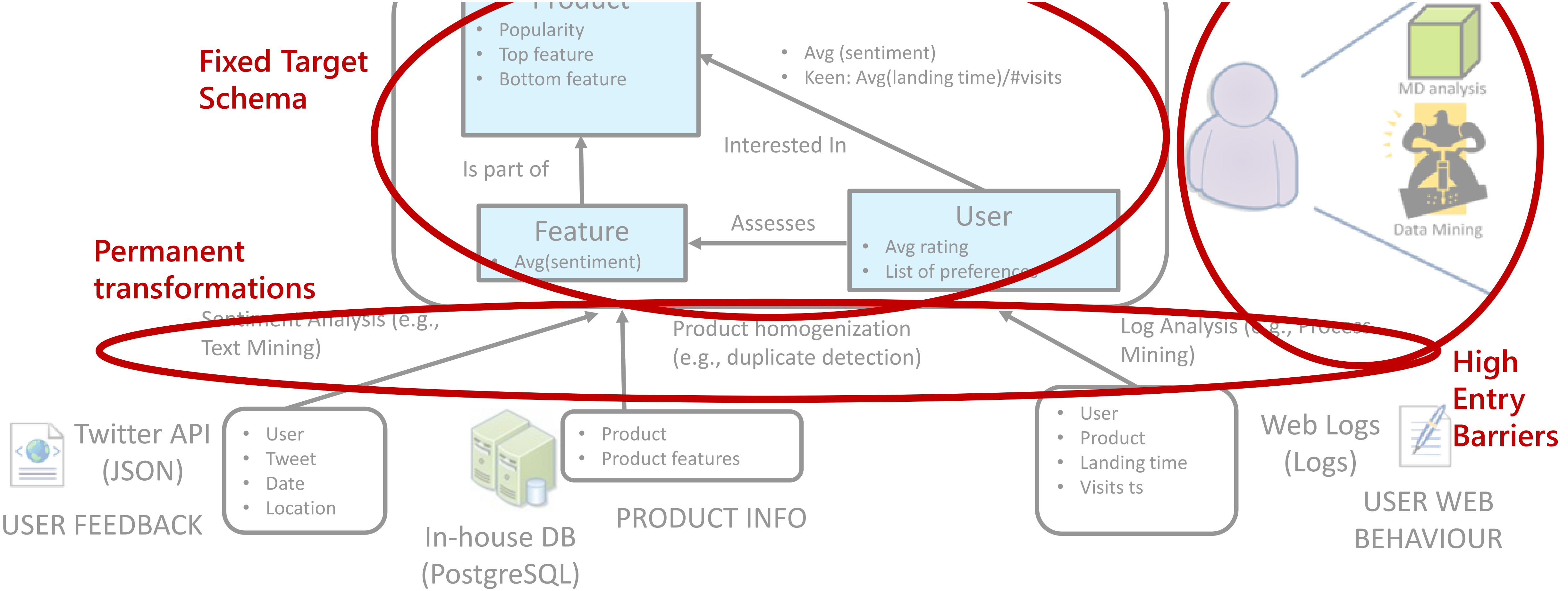
(a) First-generation platforms.

Model-First (Load-Later)



Drawbacks of Model-First (Load-Later)

- **Costly**: on-premise appliance, provisioned and payed for the peak of user load
- **Unstructured data**: more and more datasets are completely unstructured, DWs cannot cope well
- **Modelling at load time restricts the potential analysis that can be done later (Big Analytics)**



Definition

Big Data Architecture (BDA) is a framework built out of different tools and techniques that have the ability to *ingest*, *store*, *process*, and *analyze* big data sets.

Expected Workload:

- Batch processing (BD @ rest)
- Real-time processing (BD in motion)
- Interactive exploration (of BD)
- Predictive analytics and ML

BDA

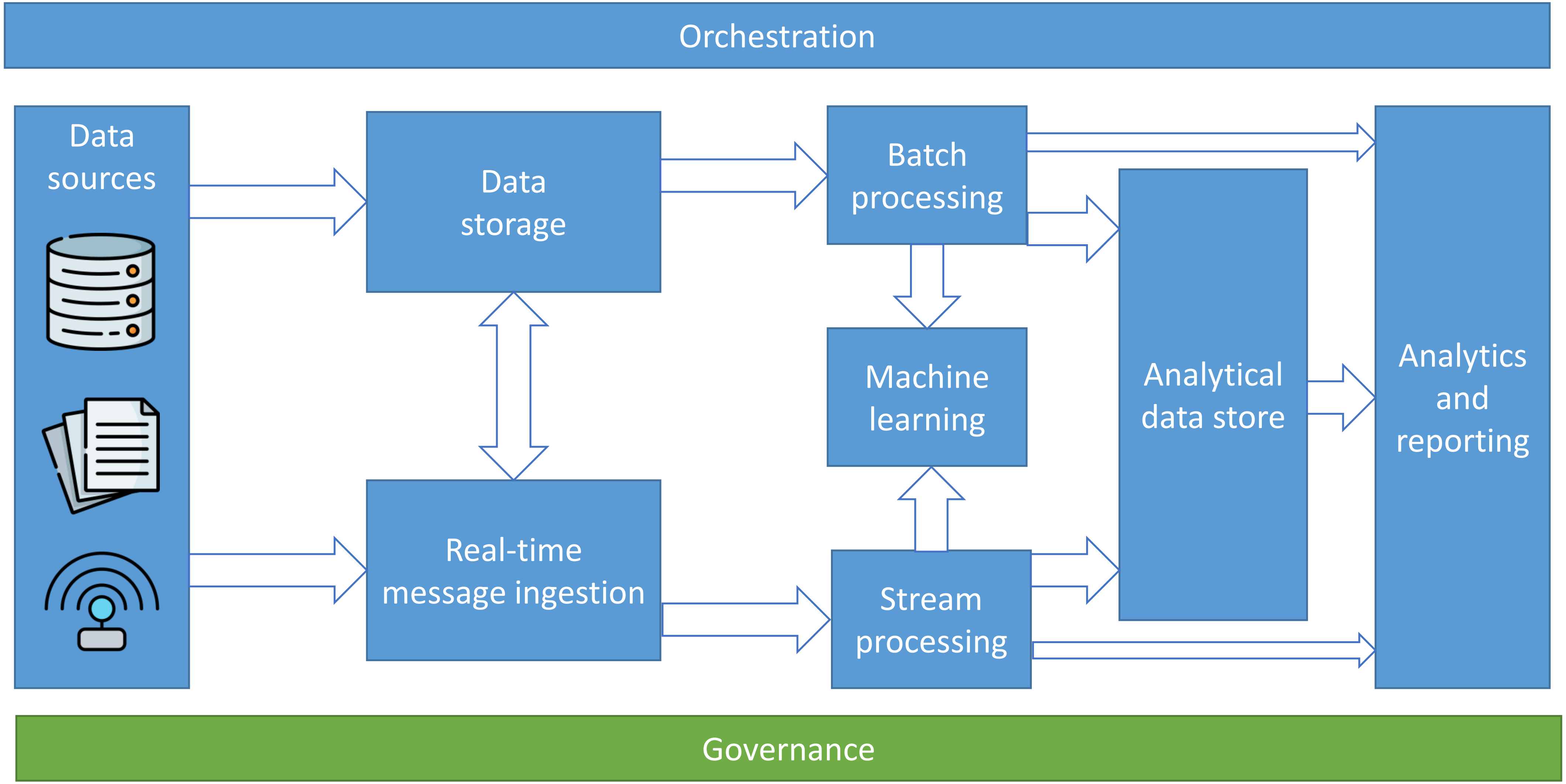
Why?

- New expectations from data
- Cheaper computation cost
- Growing data sources

When?

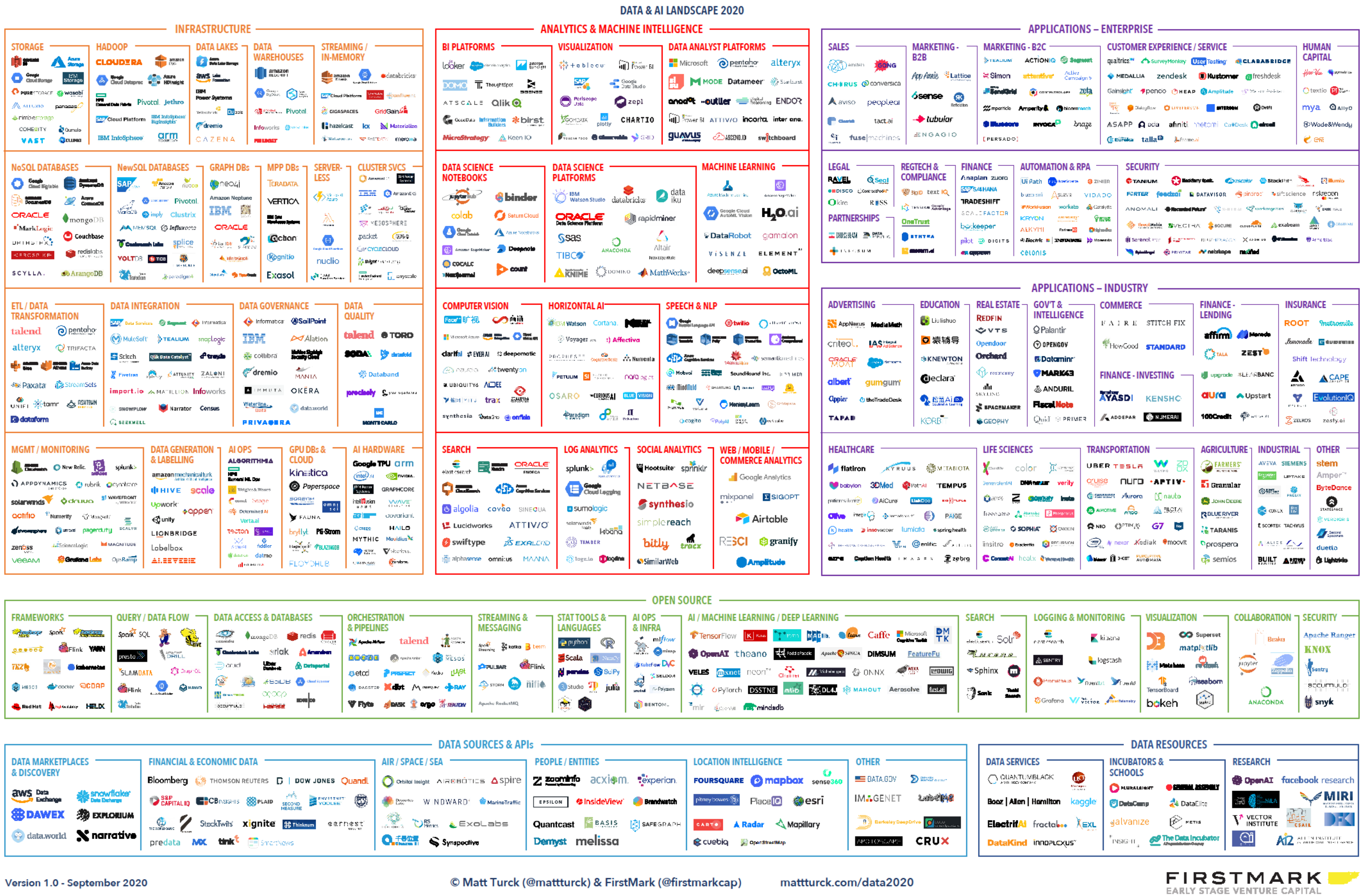
- Depends on users/tools
- Intended value
- Volume, Velocity, Variety, …
- Unstructured data
- Unbounded data streams
- Traditional DBMS can't handle
  - >100s GBs

BDA: Functional Components



Individual solutions may not contain every component.

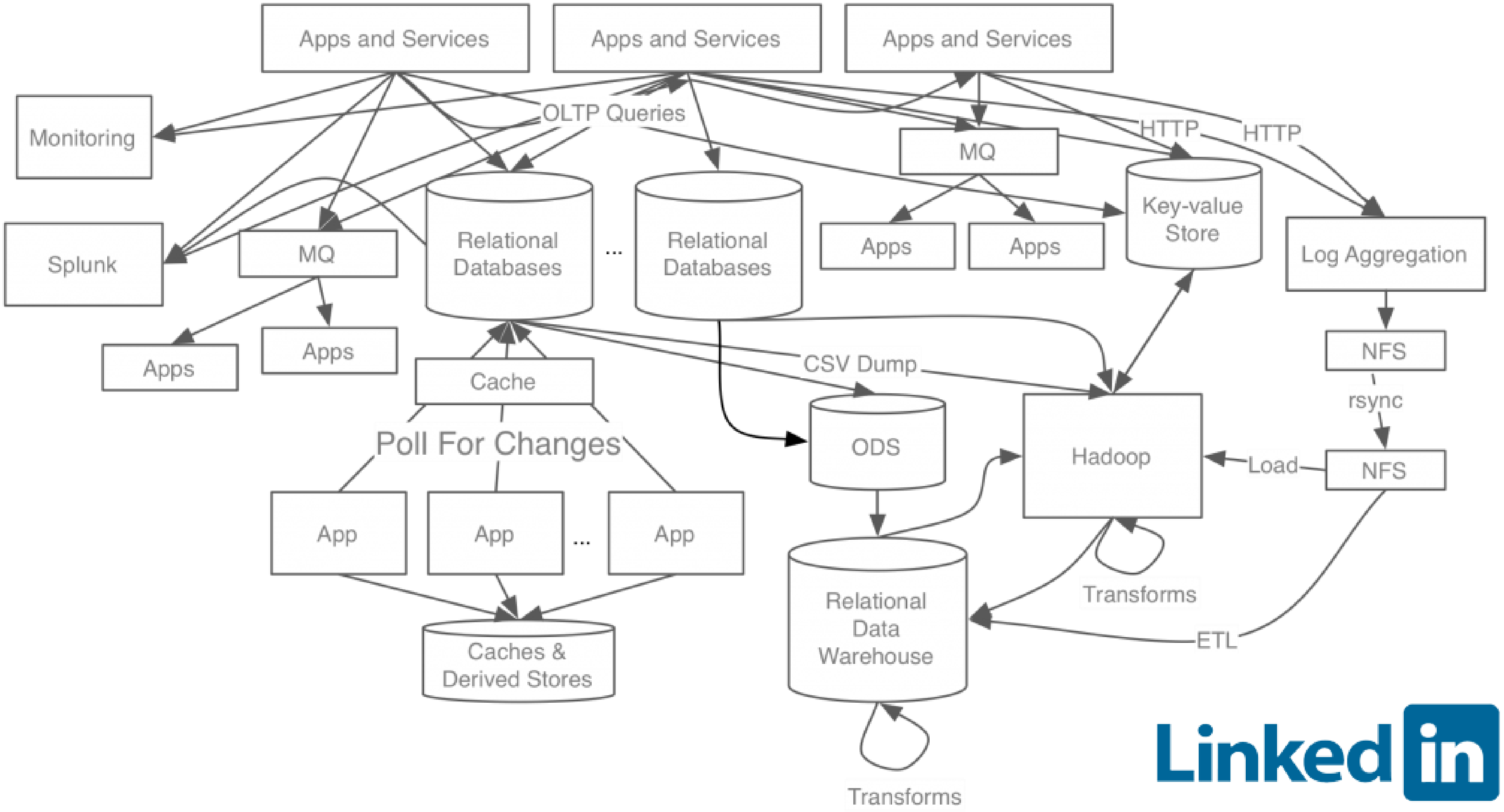
Big Data Landscape



UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

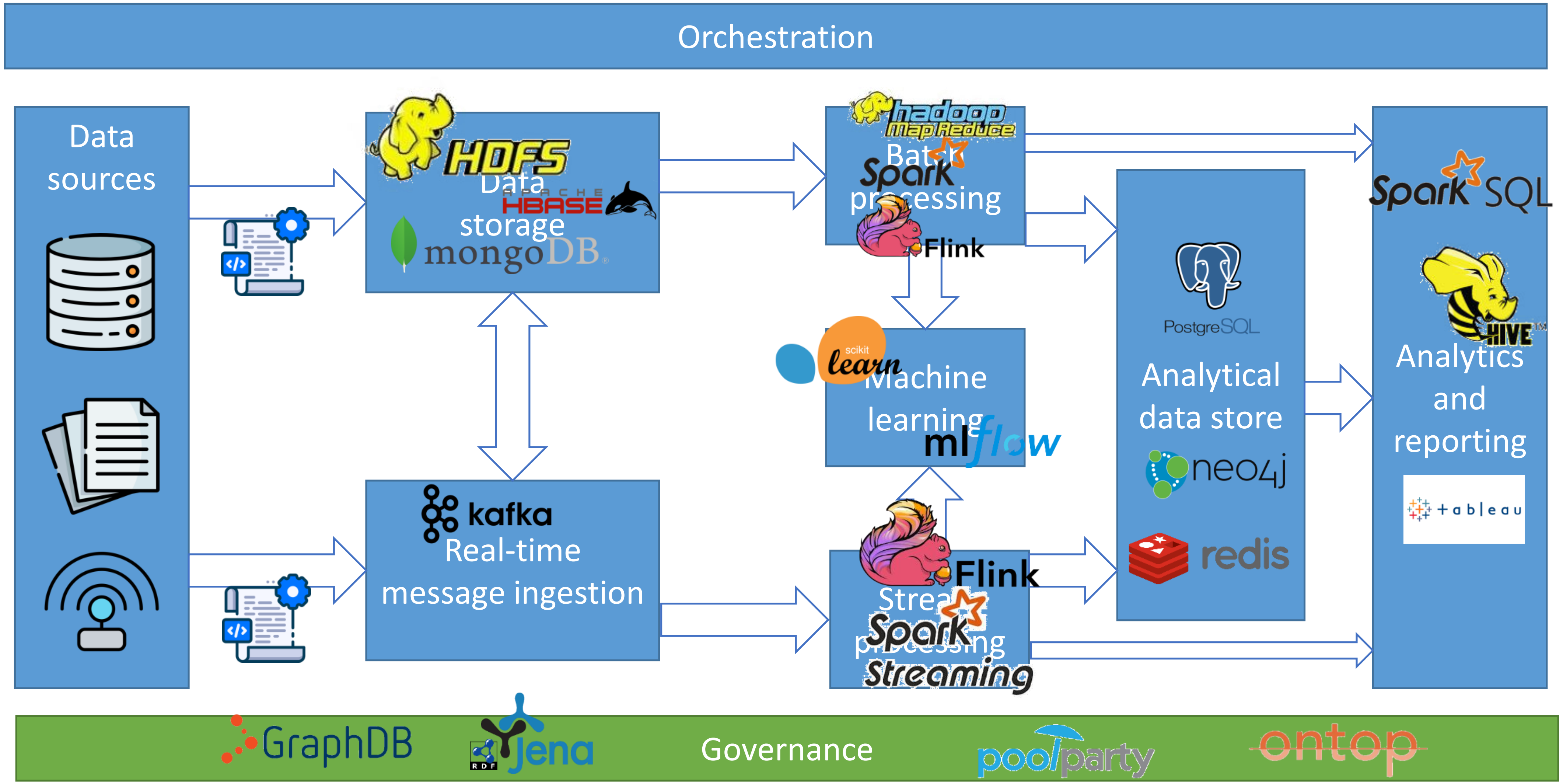
www.essi.upc.edu/dtim

Spaghetti Architecture



https://www.confluent.io/blog/event-streaming-platform-1

BDA: Functional Components



Load-First, Model-Later or Extract, Load, Transform (ELT)

Orchestrators

- Technologies to automate repeated workflows for data processing, transformation, and movement.
- Current workflow orchestrators include: Apache Oozie, sqoop, Airflow, Kaestra

- Does a similar job to global execution manager of traditional distributed RDBMS

BDA: Pros and Cons

Benefits

- Parallel computing
- Scalability
- Freedom of choice (flexibility)
- Interoperability

Challenges

- Security
- Complexity
- Evolving technologies
  - Choice paralysis
- Specialized skill set

BDA: Examples

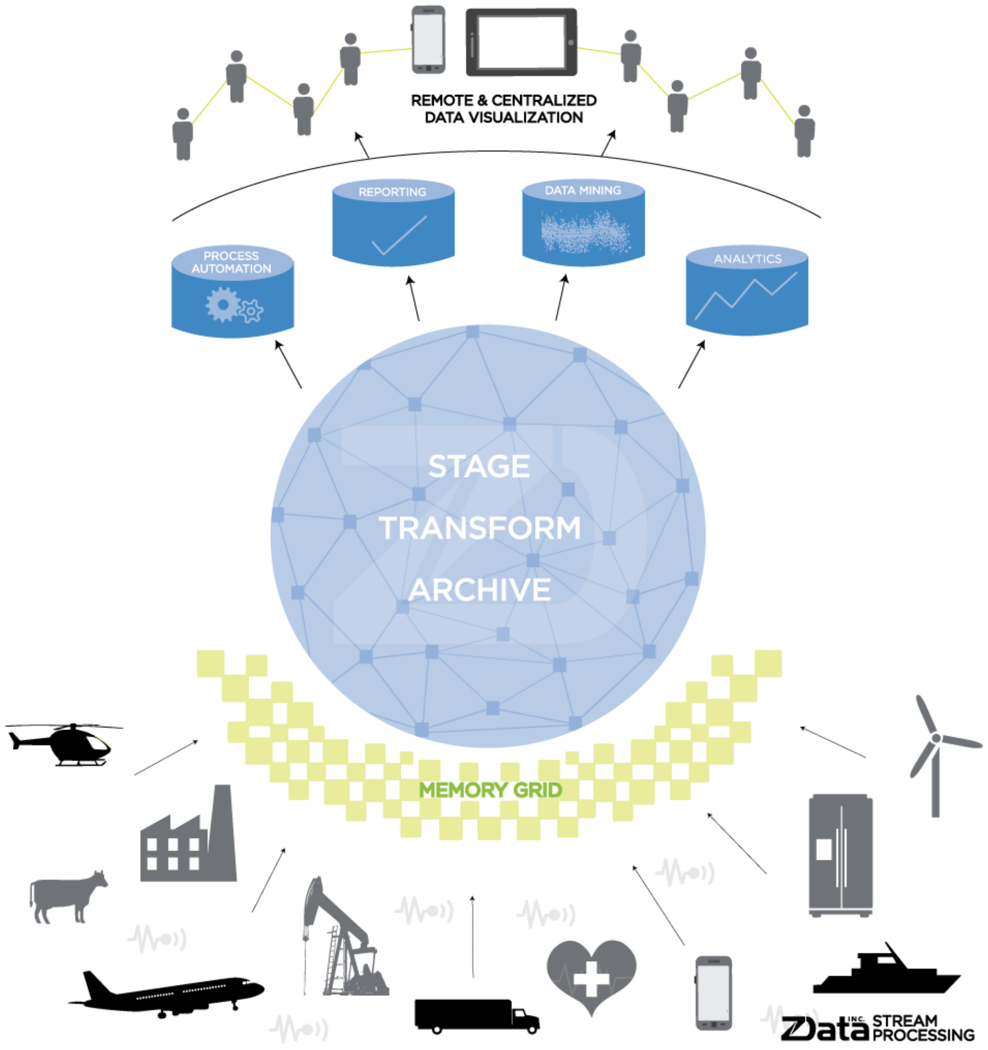
- Data lake architecture
- Lakehouse architecture
- Lambda architecture
- Kappa architecture
- Batch architecture
- Streaming (Event-driven) architecture
- Hybrid architecture

BDA: Examples

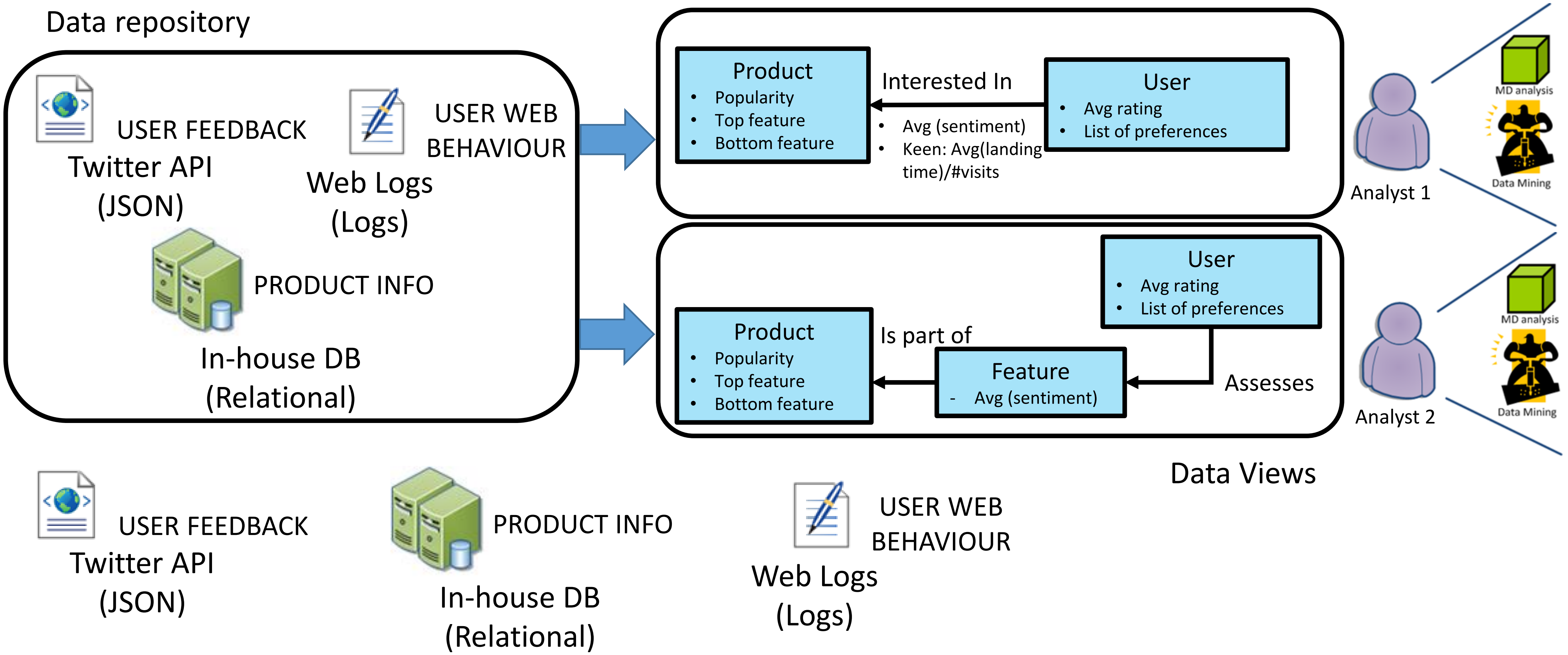
- Data lake architecture
- Lakehouse architecture
- Lambda architecture
- Kappa architecture
- Batch architecture
- Streaming (Event-driven) architecture
- Hybrid architecture

Data Lake Architecture (DLA)

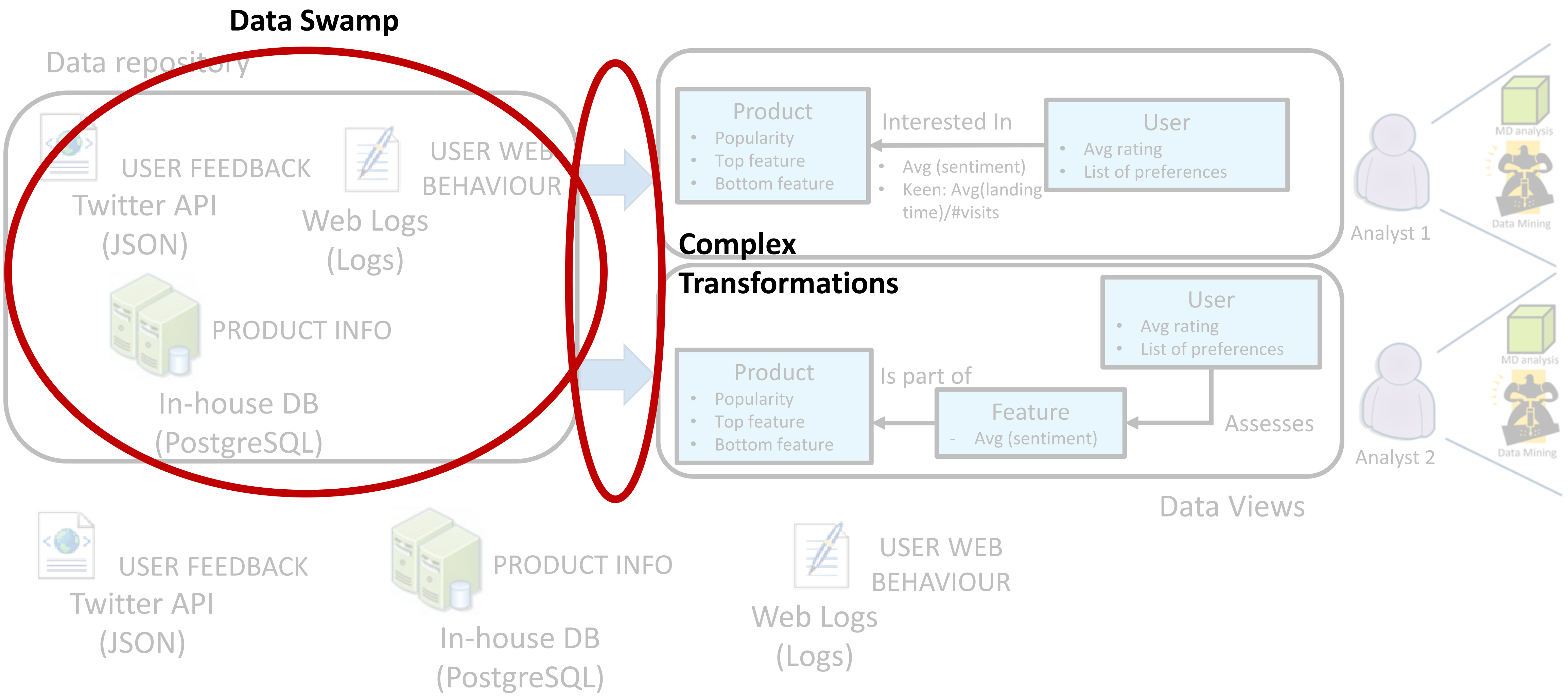
- **Idea: Load-First, Model-Later**
- Modelling at load time restricts the potential analysis that can be done later (Big Analytics)
- Characteristics:
  a) Store raw data
  b) Provide governing functionalities
  c) Create on-demand views to handle precise analysis needs



DLA: Load-First (Model-Later)

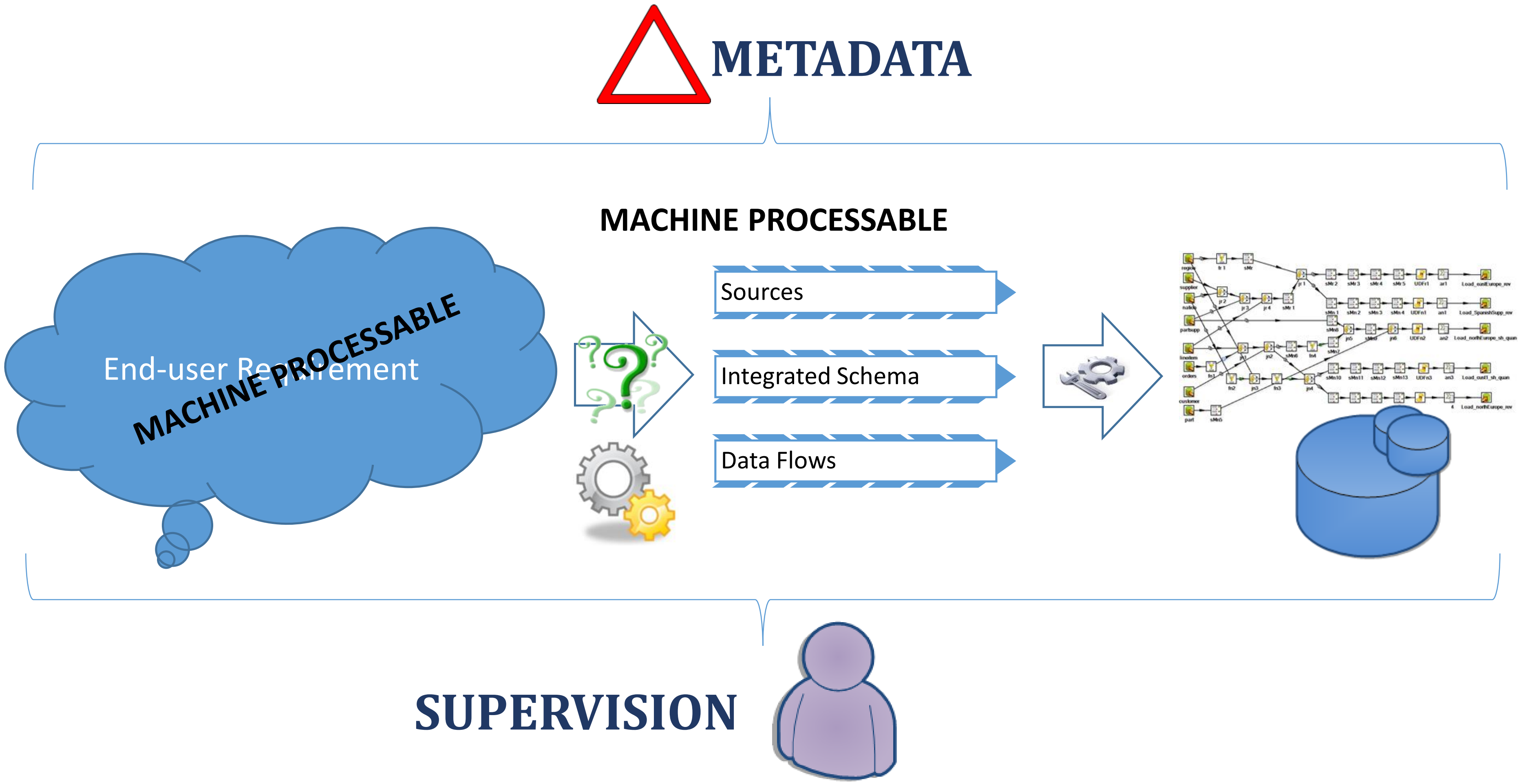


Drawbacks of DLA

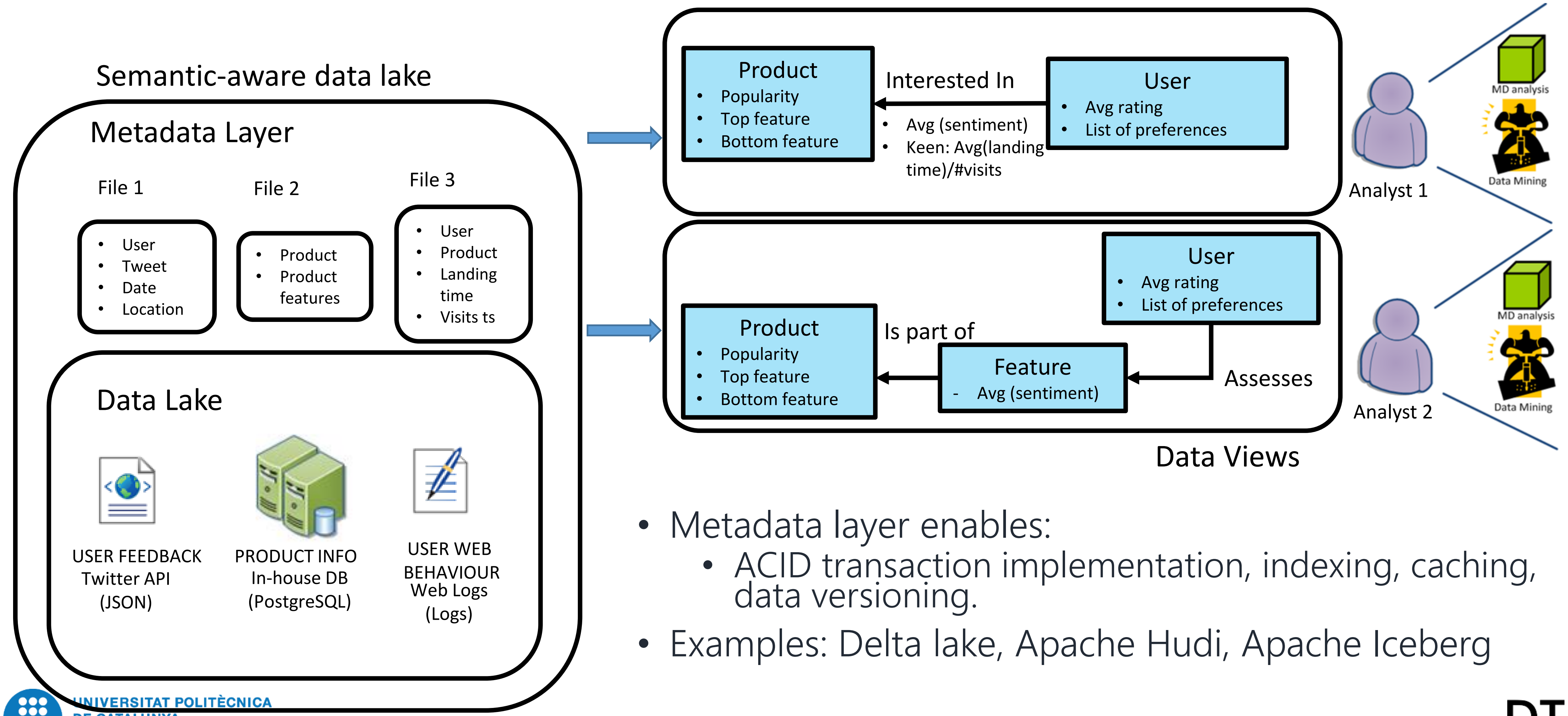


Stonebraker (2014)

DLA Missing Link: Metadata

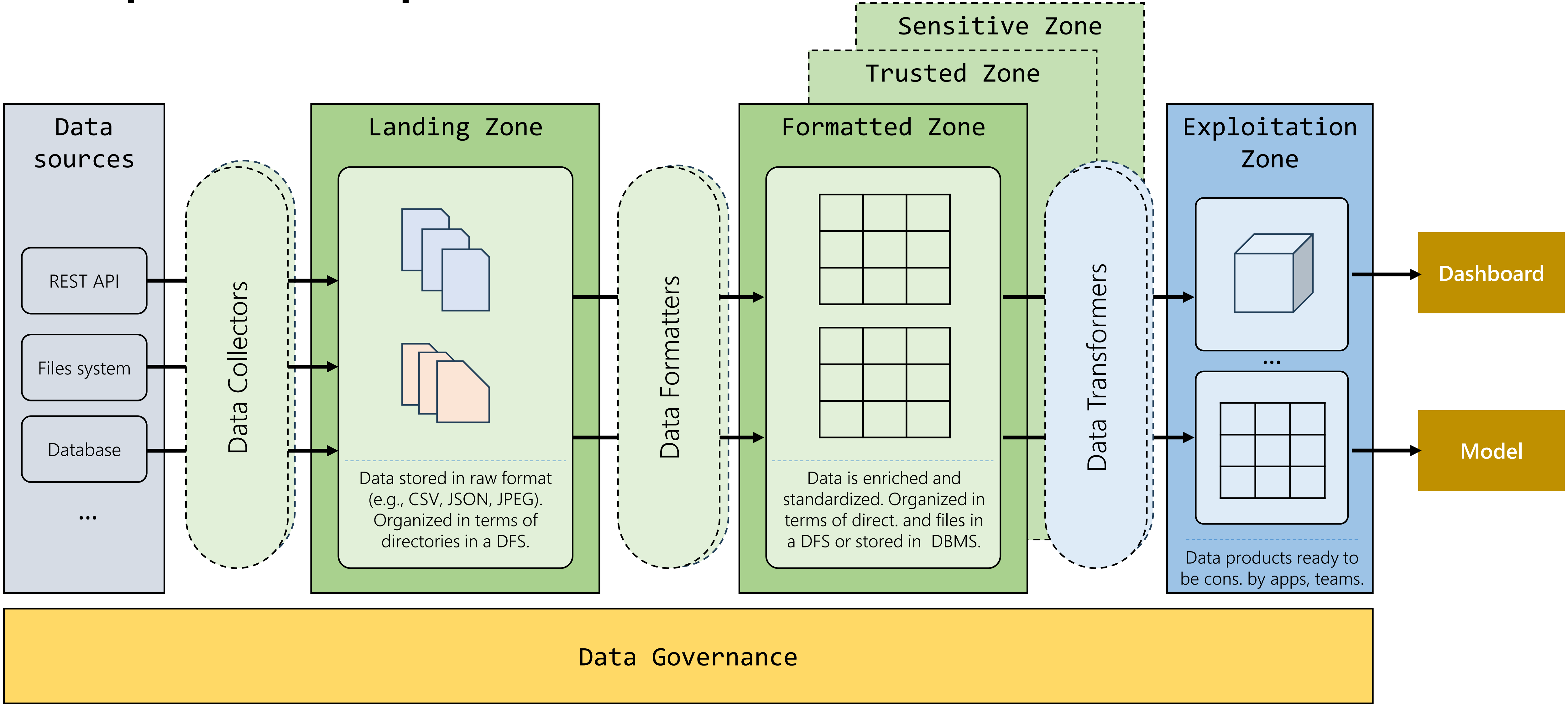


Data Lakehouse Architecture (LHA)



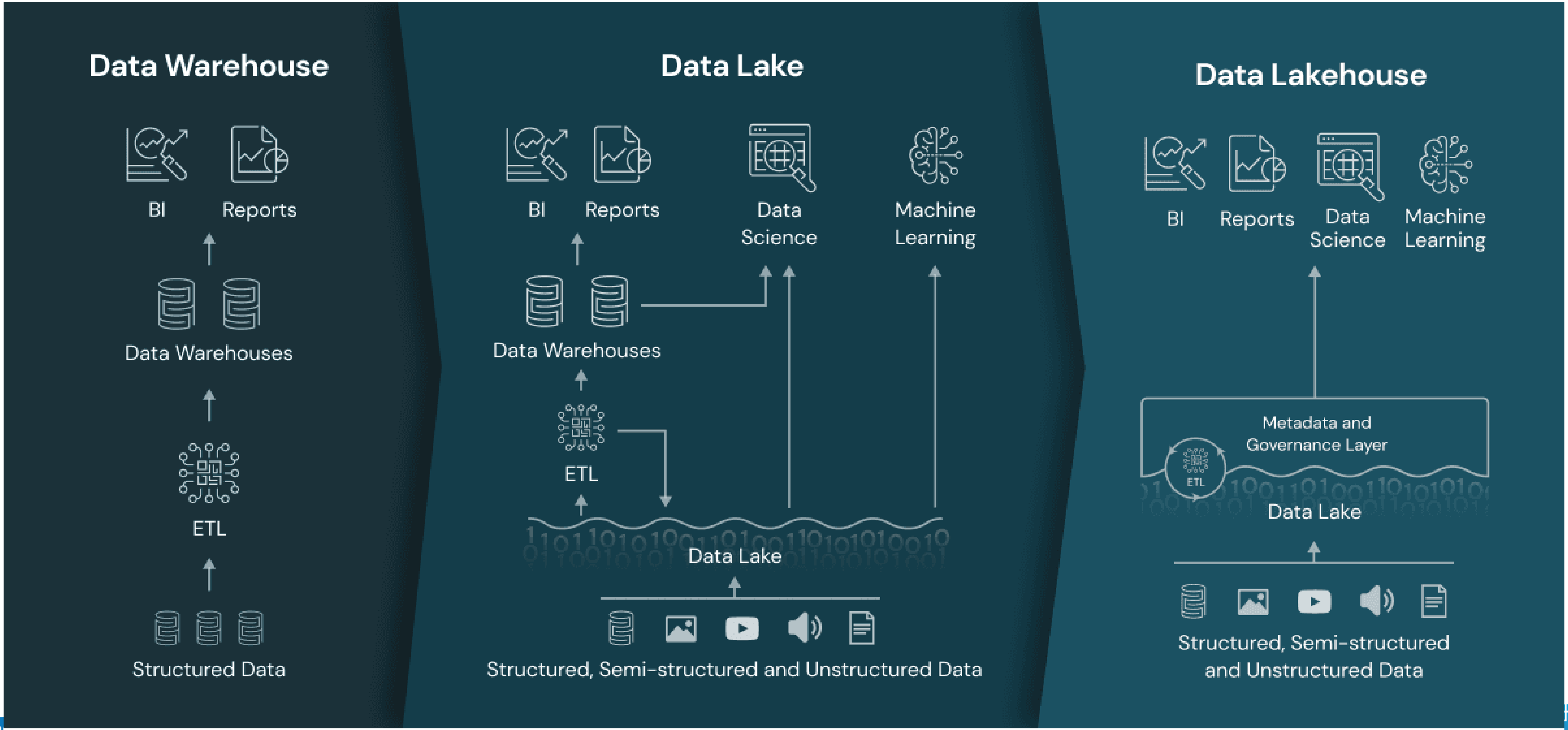
- Metadata layer enables:
  - ACID transaction implementation, indexing, caching, data versioning.
- Examples: Delta lake, Apache Hudi, Apache Iceberg

DLA: possible implementation



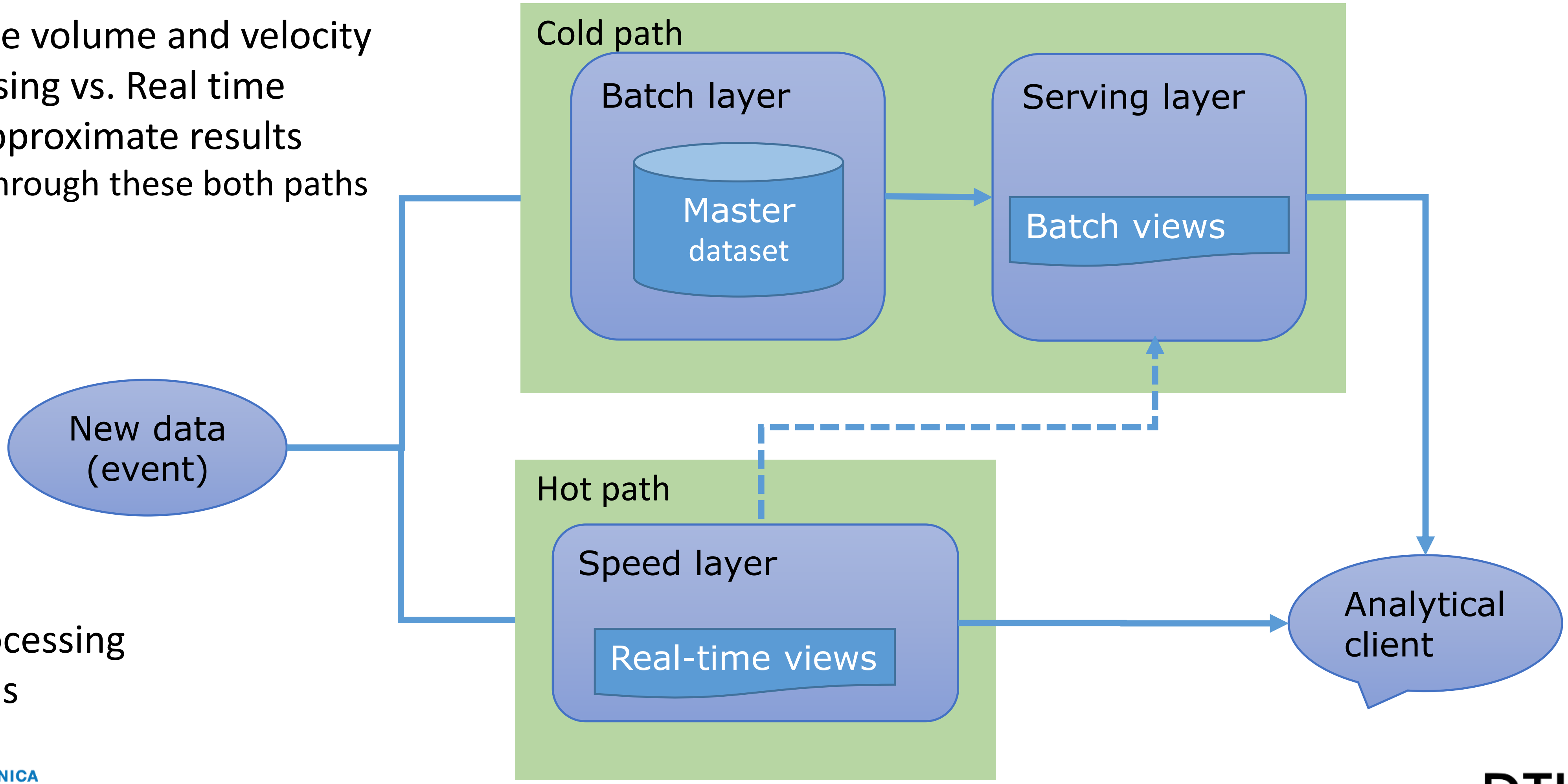
Sergi Nadal, Petar Jovanovic, Besim Bilalli, and Oscar Romero. Operationalizing and automating Data Goverance. **Journal of Big Data 2022**.

Architectural Evolution



Lambda Architecture

- Accommodate volume and velocity
- Batch processing vs. Real time
- Precise vs. Approximate results
- All data goes through these both paths

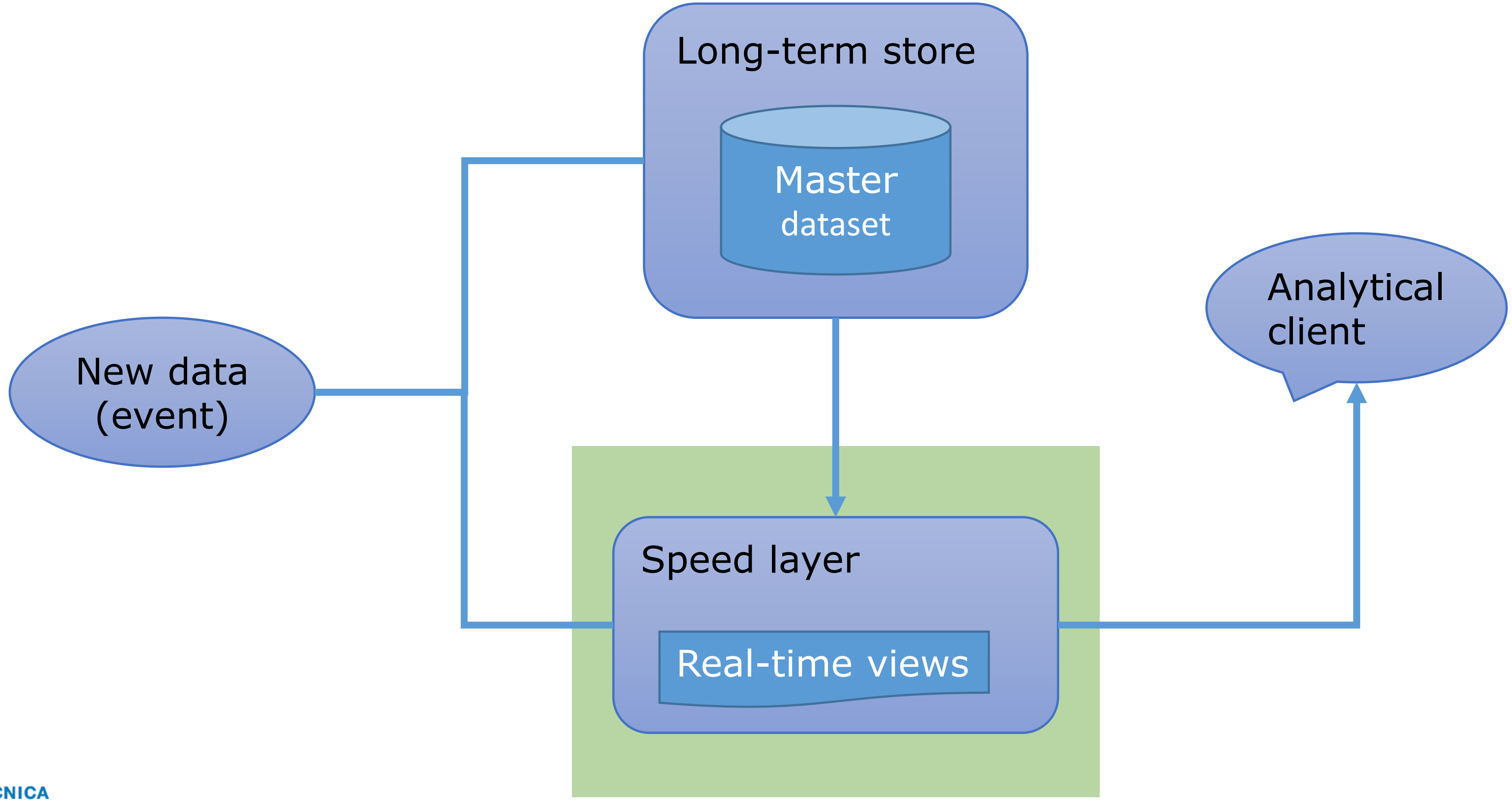


Drawbacks:

- Duplicate processing
- Different tools
- Complexity

Kappa Architecture

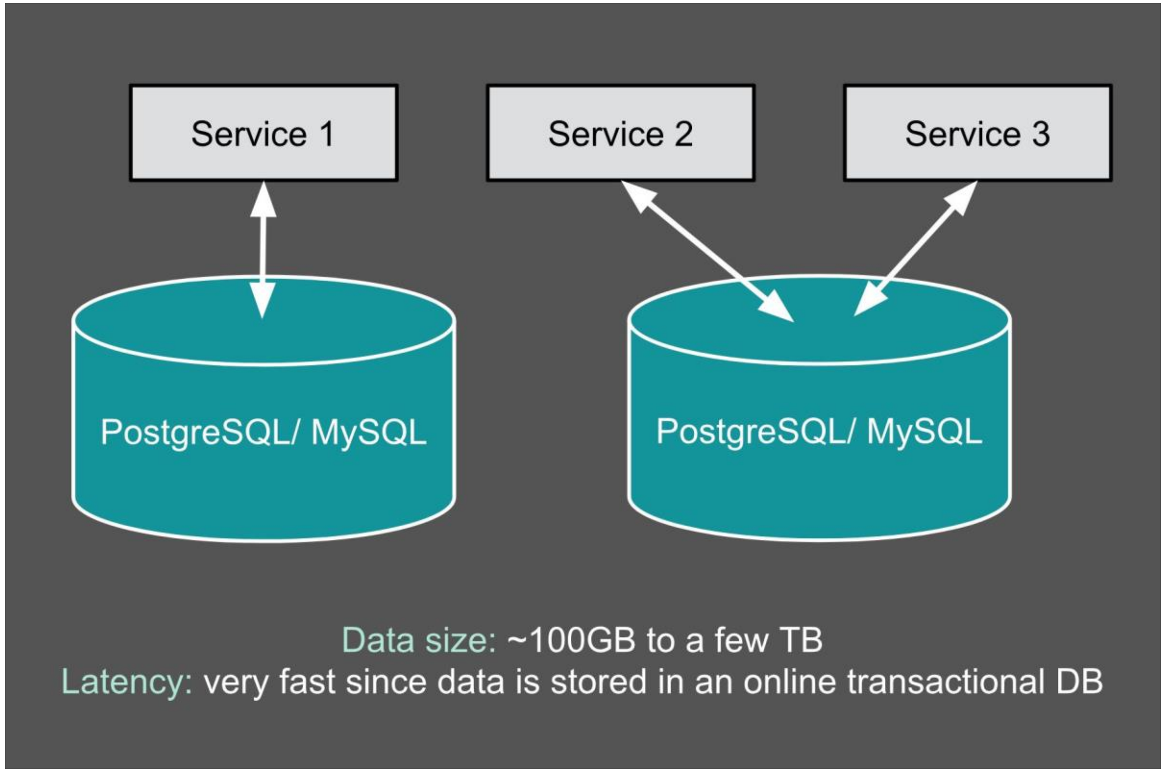
- All data goes through these one path



Real-world Case Example

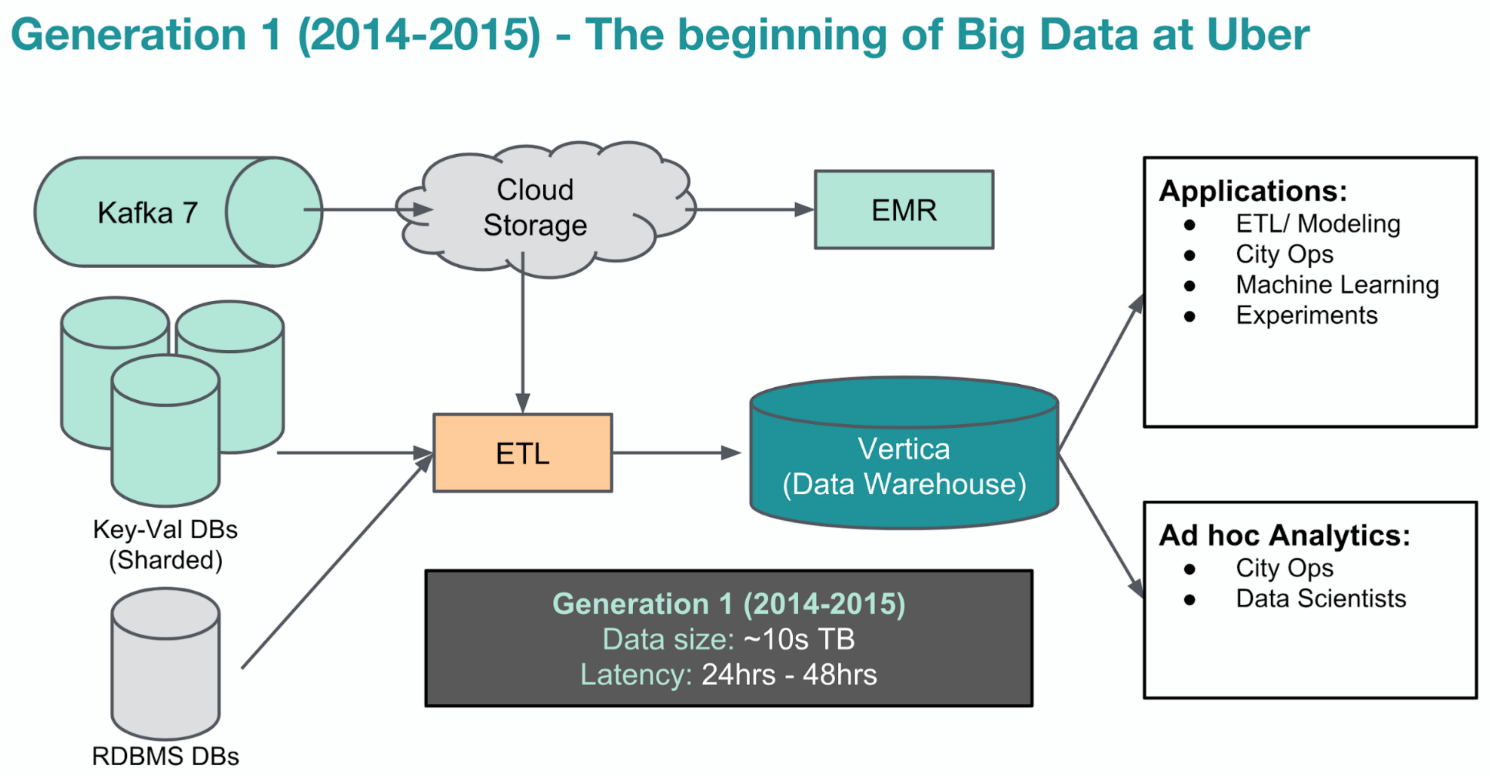
Evolution of Uber's Big Data Architecture ([article]())

Before 2014



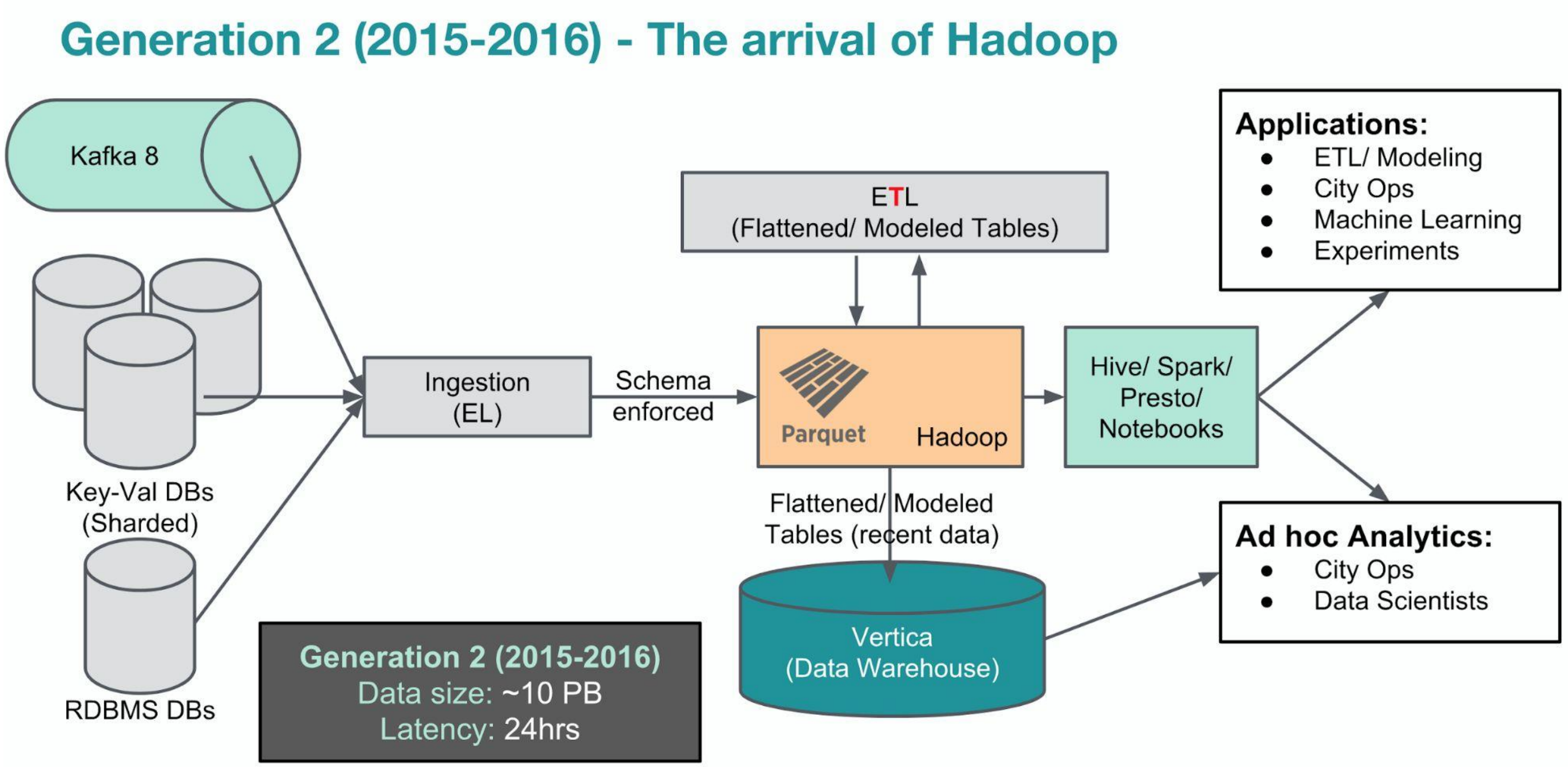
Real-world Case Example

Evolution of Uber's Big Data Architecture ([article]())



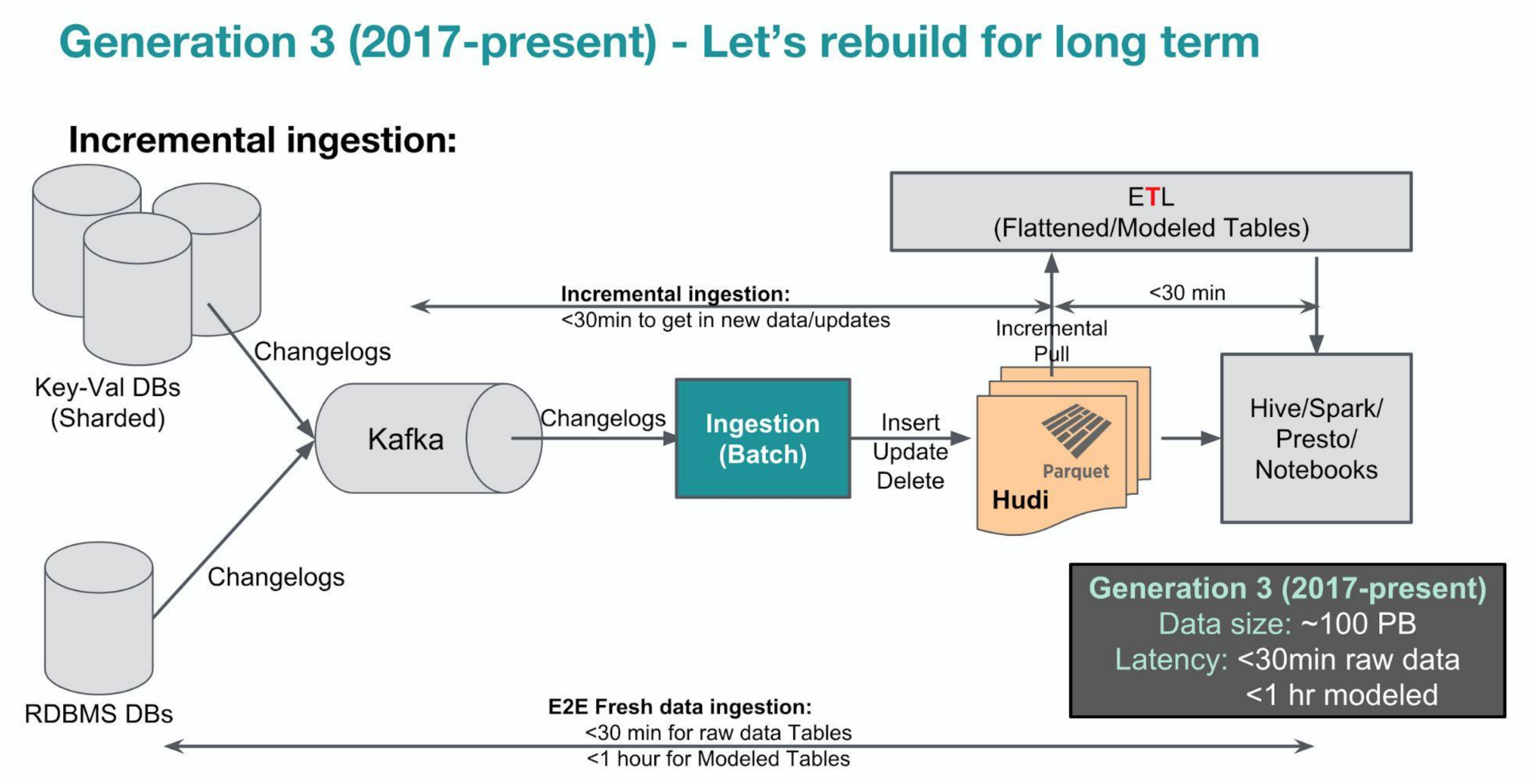
Real-world Case Example

Evolution of Uber's Big Data Architecture ([article]())



Real-world Case Example

Evolution of Uber's Big Data Architecture ([article]())



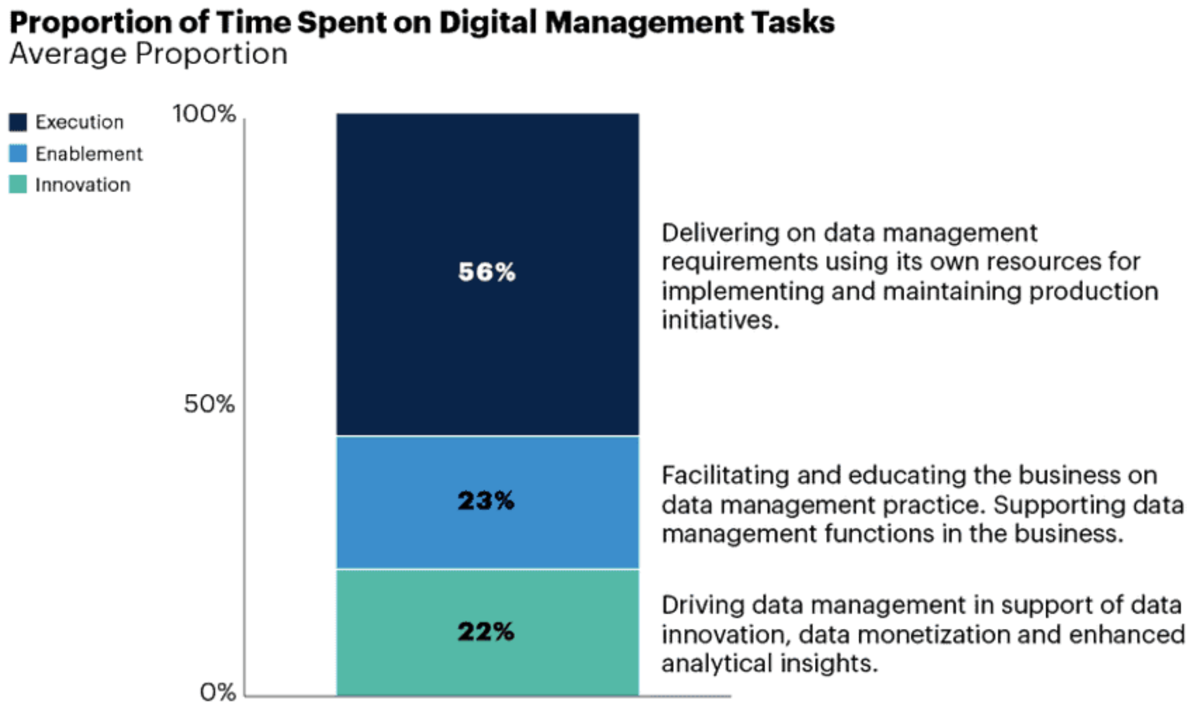
BDA: Summary

- Traditional Data Management
  - Centralized or Distributed DBMS
  - Model-first, Load-later (ETL)
- Big Data Management Architecture
  - Functional components
  - Example architectures: data lake, lakehouse, lambda, kappa
- Uber case example

DataOps

Architectures & DataOps

Problem definition



March 2020 Gartner Survey: "Data Management Struggles to Balance Innovation and Control"

Too much time on low-value tasks, such as data integration and formatting, ad hoc queries and requests, and generating reports and dashboards — activities that are ripe for **automation**.

Challenges:

- Deployment latency
- Production errors (reactive)
- Teams ☹
- Poor time usage

Definition

DataOps is a set of collaborative data management **practices** to :

- Reduces cycles of experimentations
- speed delivery
- lower error rates
- maintain quality
- foster collaboration
- provide maximum value from data.
- monitoring and feedback

DevOps: Why?

- Multiple tools
- Multiple datasets
- Multiple paths
- Multiple methods
- Multiple architectures
- Multiple customers
- Multiple people

- *Trust but verify*
- *Expectation VS Outcomes*

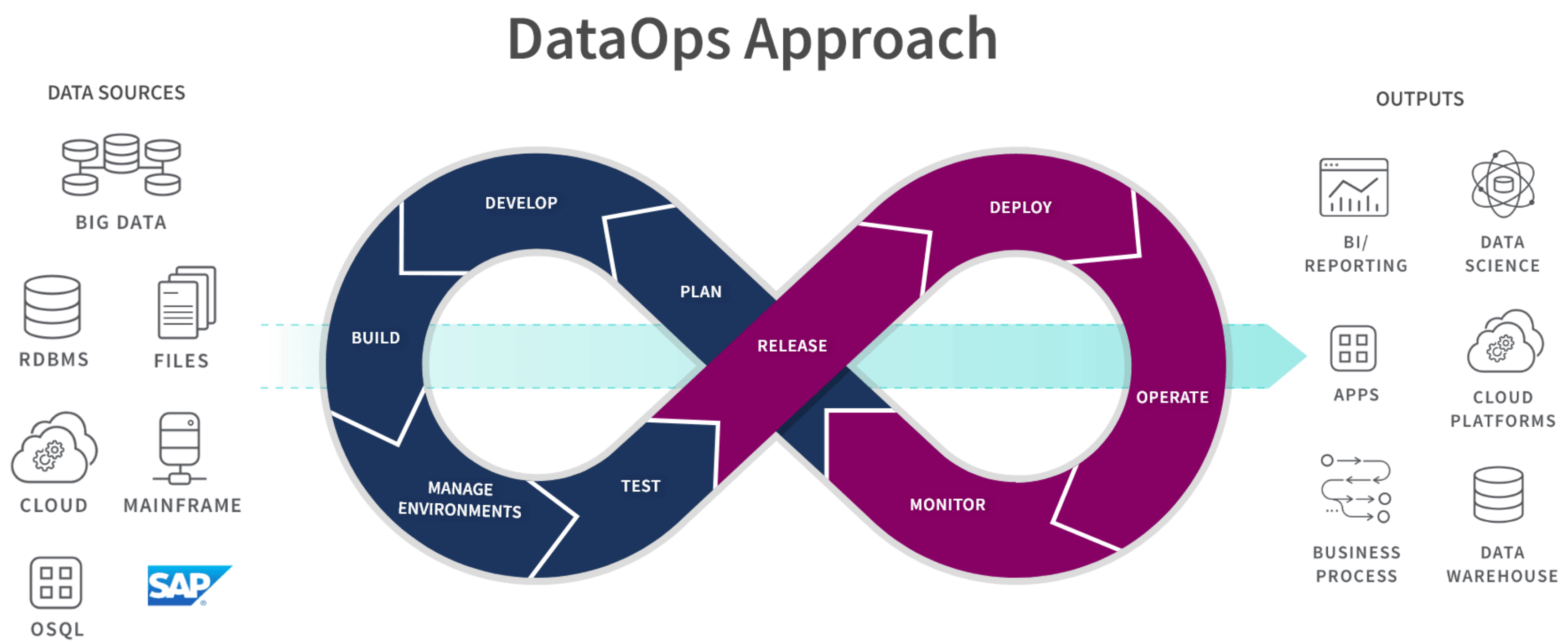
Agile Methodology

Agile methodology is a broad term that can refer to any project management methodology that uses an ***iterative and flexible*** approach.

- Principles:
  - Iterative development
  - Continuous improvement
  - Feedback
  - Adaptability

Goal: deliver value to users early and often

DataOps methodology



People, processes, technology *Data as a product*

DataOps Lifecycle

- **Plan:** Partner with product, engineering, and business teams to establish goals.
- **Develop:** Create data products and machine learning models for your data application.
- **Build:** Incorporate the code and/or data product into your existing tech or data stack.
- **Manage Environments:** segregate environments, collaborate across branches, and set environment variables.
- **Test:** Verify that your data conforms to business logic and meets operational standards.
- **Release:** Launch your data in a test environment.
- **Deploy:** Integrate your data into production.
- **Operate:** Utilize your data in applications such as dashboards and data loaders.
- **Monitor:** Continuously observe and report any irregularities in your data.

Things to Automate

- **Data curation services:** data cleansing, transformation and standardization
- **Metadata management:** lineage tracking, where data comes from, how it's transformed and how it's used
- **Data governance:** enforces data quality rules and access controls
- **Master data management:** data deduplication and synchronization across systems to ensures a single source of truth
- **Self-service interaction:** data access and exploration

DataOps Functions and Tools

- **Data ingestion:** Apache Kafka, AWS Glue, or Fivetran
- **Data orchestration**: Apache Airflow, Prefect, or Dagster
- **Data transformation:** dbt, Apache Spark, or Snowflake
- **Data catalog:** Alation, Collibra, or AWS Glue Data Catalog
- **Data observability:** Monte Carlo, Datadog, or Great Expectations.

DataOps: Summary

- Automation is key to operational efficiency
- Enables proactive error detection and resolution
- Fosters collaboration and agile adaptation to changing needs
- Promotes continuous improvement through iterative cycles
- Reduces manual workload and enhances strategic resource allocation
- Ensures data consistency, reliability, and compliance

References

- D. McCreary and A. Kelly. *Making Sense of NoSQL*. Manning, 2014
- M. Grover et al. *Hadoop Application Architectures*. O'Reilly, 2015
- N. Marz and J. Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015
- D. Sculley et al. Hidden technical debt in machine learning systems. In Advances in Neural Information Processing Systems, 28. Curran Associates, Inc., 2015.
- S. Nadal et al. *A Software Reference Architecture for Semantic-Aware Big Data Systems*. Information and Software Technology 90. Elsevier, 2017
- S. Nadal et al. *ODIN: A Dataspace Management System*. International Semantic Web Conference 2019
- S. Nadal. *Metadata-Driven Data Integration* (PhD Thesis). 2019
- R. Hai, et al. *Data lake concept and systems: a survey*. CoRR abs/2106.09592. 2021
- P. Jovanovic et al. *Quarry: A User-centered Big Data Integration Platform*. Information Systems Frontier, 2021