Big Data Architectures

Big Data Management

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

DTIM
www.essi.upc.edu/dtim

Knowledge objectives

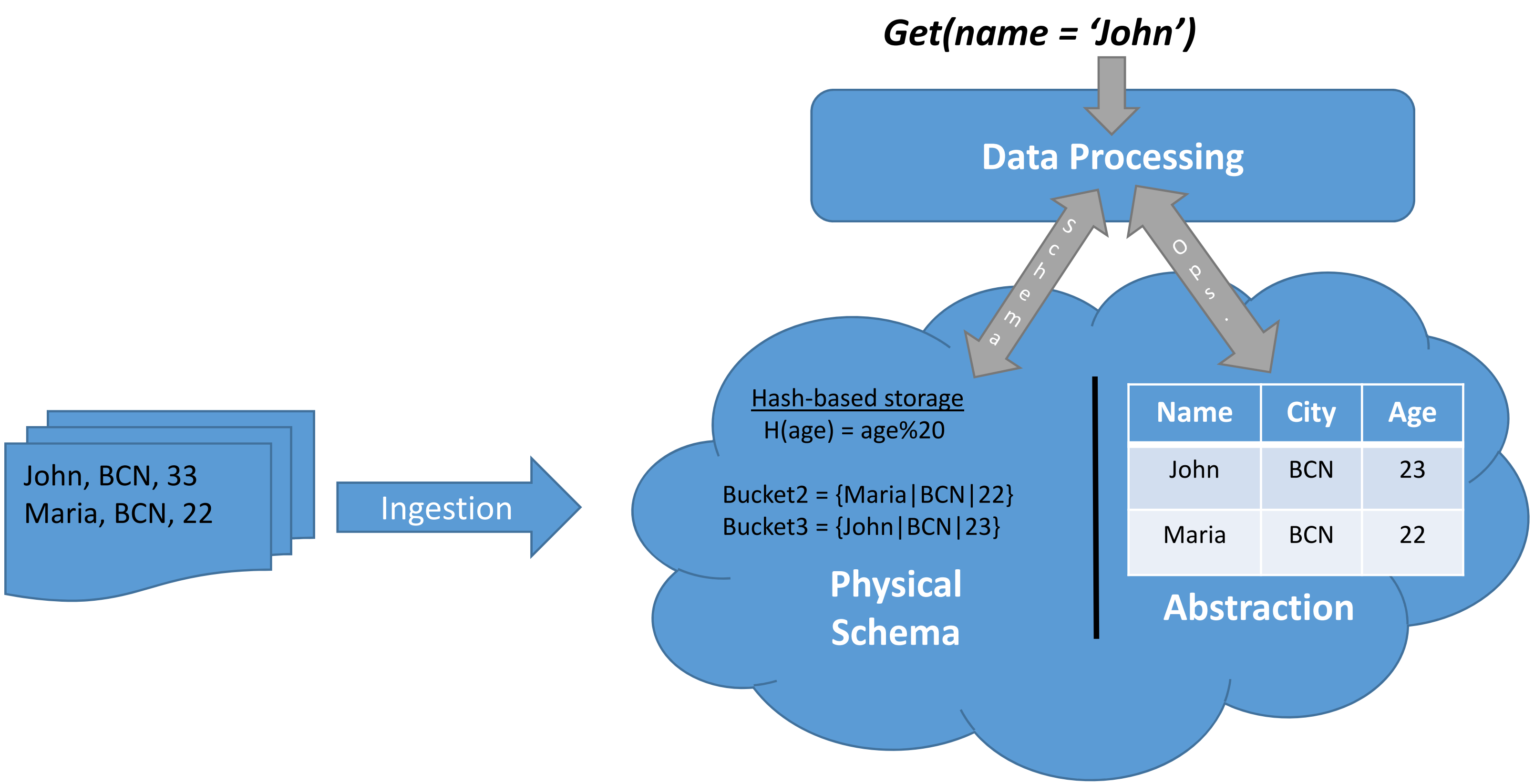
1. Explain the problema of a spaghetti architecture
2. Explain the need of the Lambda architecture
3. Explain the difference between the Kappa and Lambda architectures
4. Justify the need of a Data Lake
5. Identify the difficulties of a Data Lake
6. Explain the need of each component in the Bolster architecture
7. Map the components of Bolster to a RDBMS architecture

Application Objectives

1. Given a use case, define its software architecture

Problem definition

Data Management (I)



Data Management (II)

Data management refers to the features a DBMS must provide:

- Ingestion: means provided to insert /upload data
  - E.g., ORACLE SQL*Loader
- Storage: format/structures used to persist data
  - E.g., hash, B-tree, heap file
- Modelling: arrangement of data within the available structures
  - E.g., normalization, partitioning
- Processing: means provided to manipulate data
  - E.g., PL/SQL
- Querying/fetching: means provided to allow users to retrieve data
  - E.g., SQL, relational algebra

In Big Data settings, they are the same concepts but assuming NOSQL underneath

- Typically, a distributed system
- Possibly with an alternative data model to the Relational one
- Implementing ad-hoc architectural solutions

Big Data Architectures

- Question the main principles of traditional DB architectures
- Implement from scratch the whole stack
  - Ingestion, Storage, Modeling, Processing, and Querying
- Use new trendy technological features
  - Primary indexes to implement the global catalog
    - Distributed Tree
    - Dynamic Hashing
  - In-memory processing
  - Columnar block iteration: vertical fragmentation + fixed-size values + RLE compression
    - Heavily exploited by column-oriented databases
    - Good for read-only workloads
  - Sequential reads for large workloads
    - Take the most out of databases by boosting sequential reads
      - Enables pre-fetching
      - Option to maximize the effective read ratio (by a good DB design)
        - Key design

The Multi-Project Approach

- The DBMS tasks can be spread over different systems
  - Independent
  - Heterogeneous
- Hadoop as paradigmatic case:
  - Storage: HDFS + Hbase
  - Modeling: HCatalog
  - Ingestion: Sqoop
  - Processing: Spark
  - Querying: Spark SQL

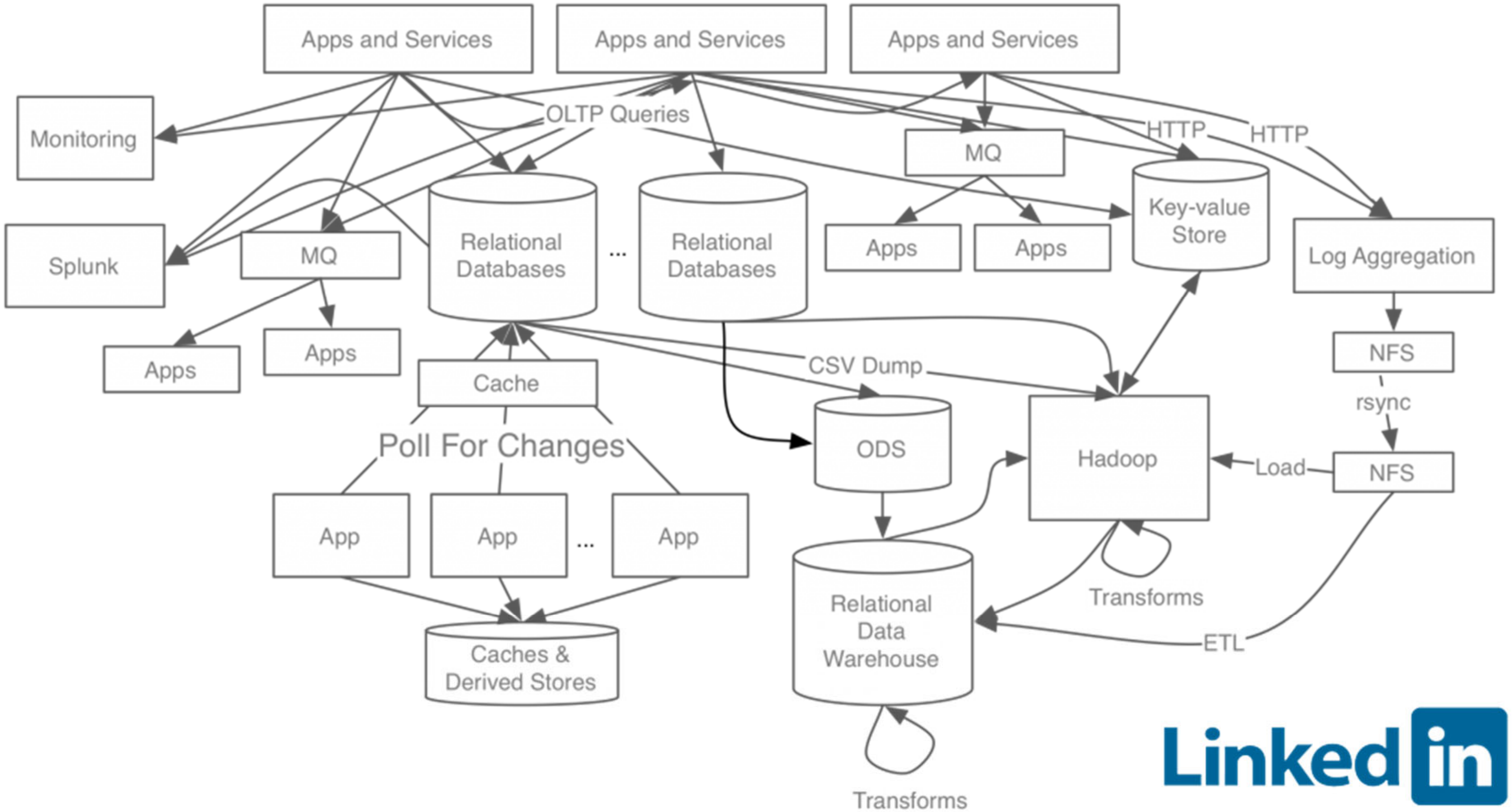
Big Data Landscape

V1 – Last updated 6/19/2018

© Matt Turck (@mattturck), Demilade Obayomi (@demi_obayomi), & FirstMark (@firstmarkcap) mattturck.com/bigdata2018

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

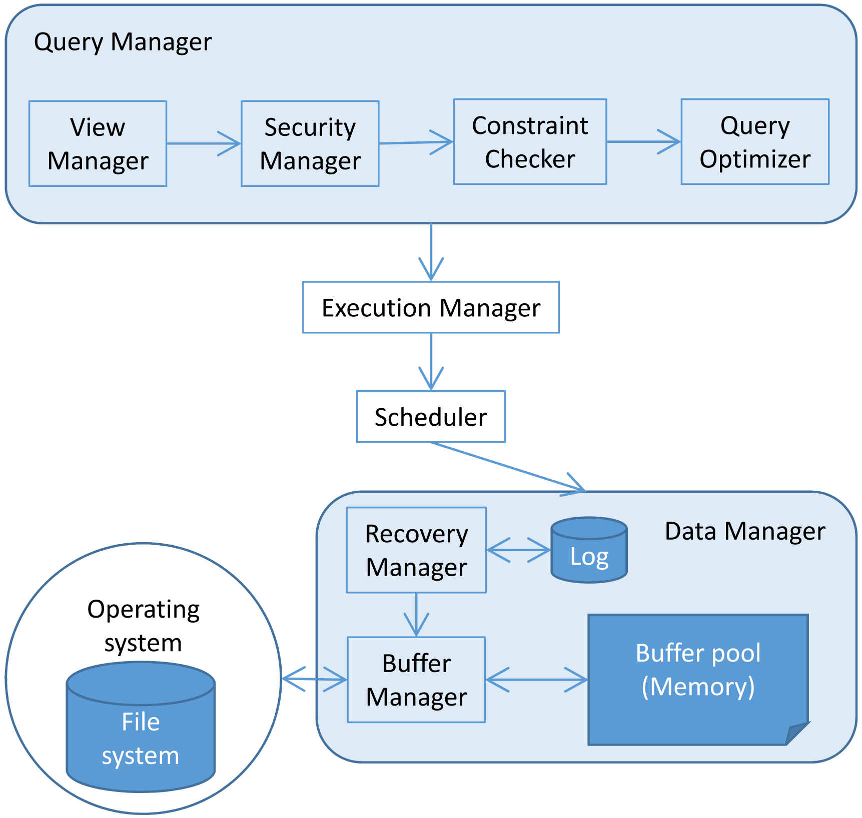
Spaghetti architecture



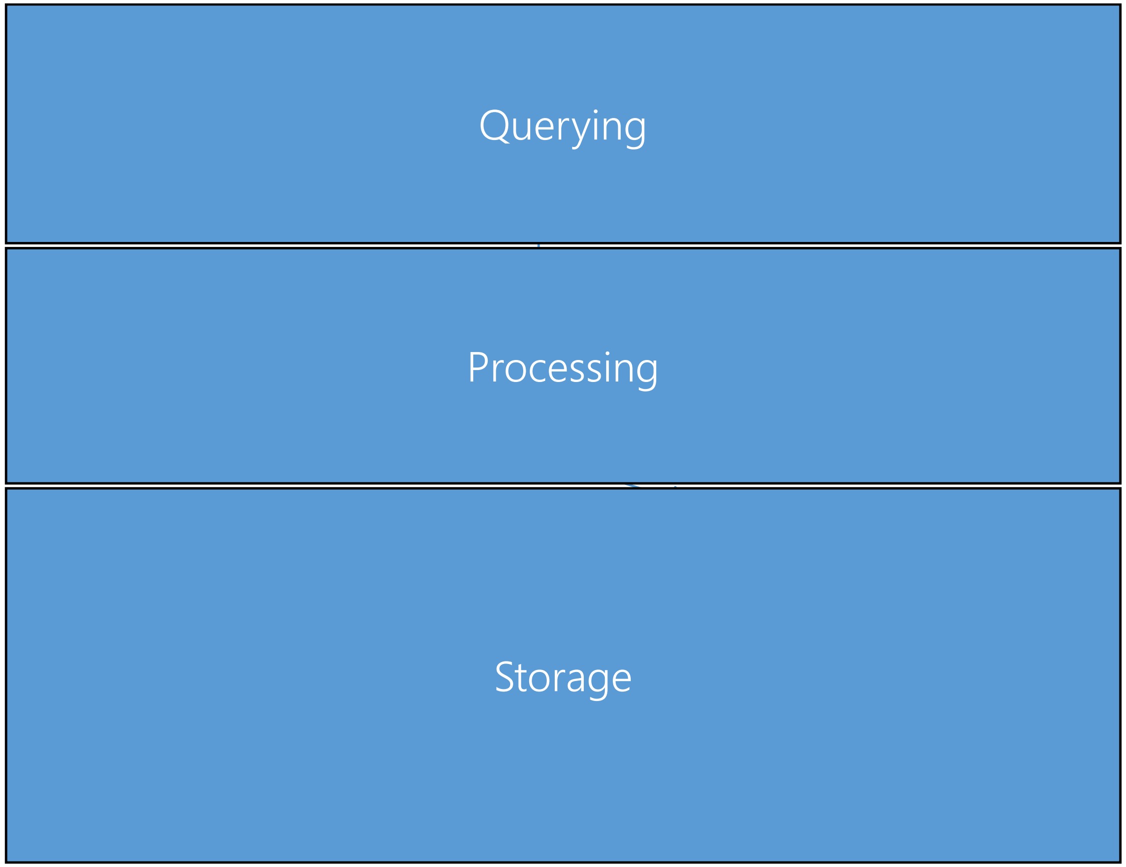
https://www.confluent.io/blog/event-streaming-platform-1

Database Management System view

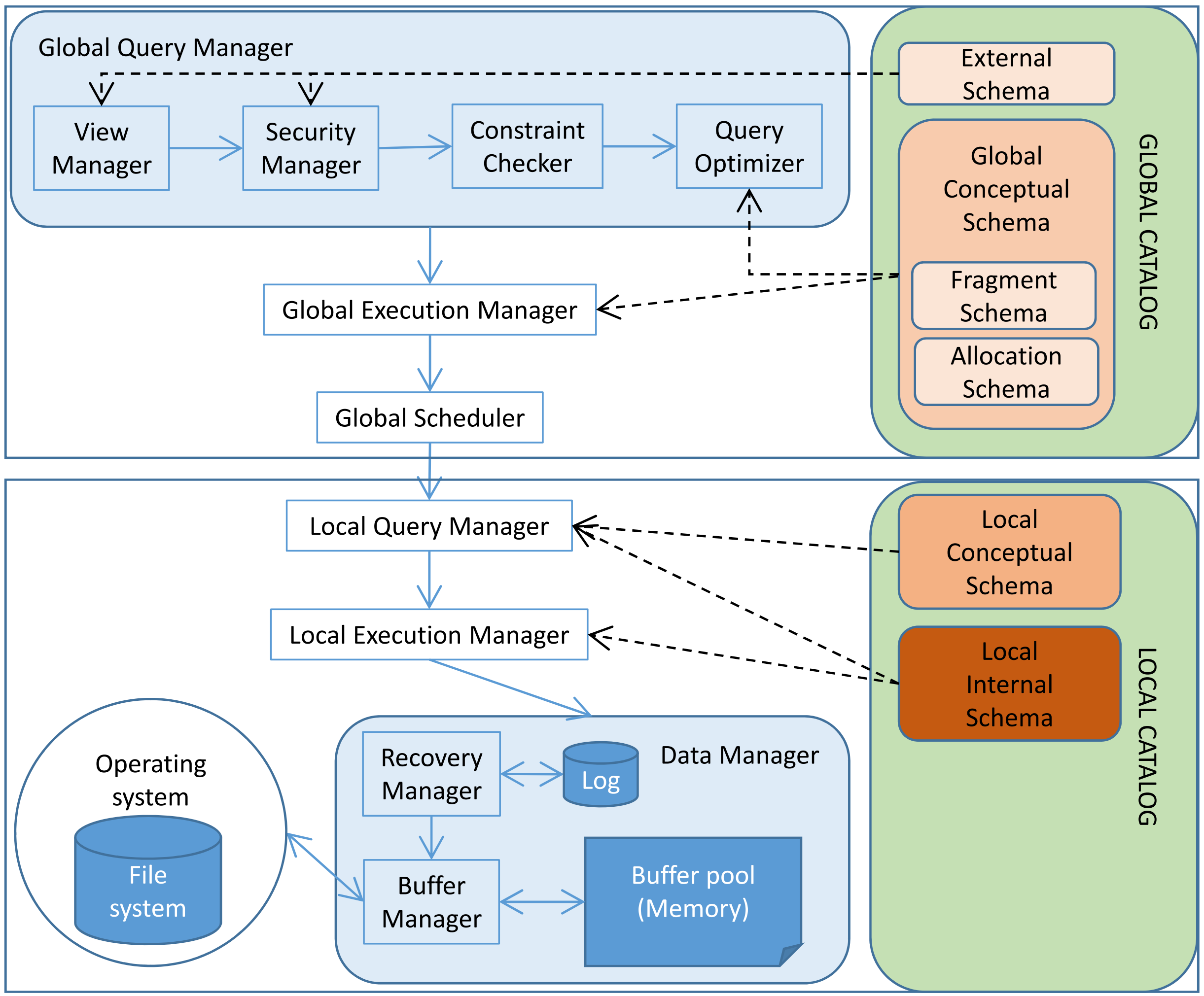
Centralized DBMS Architecture



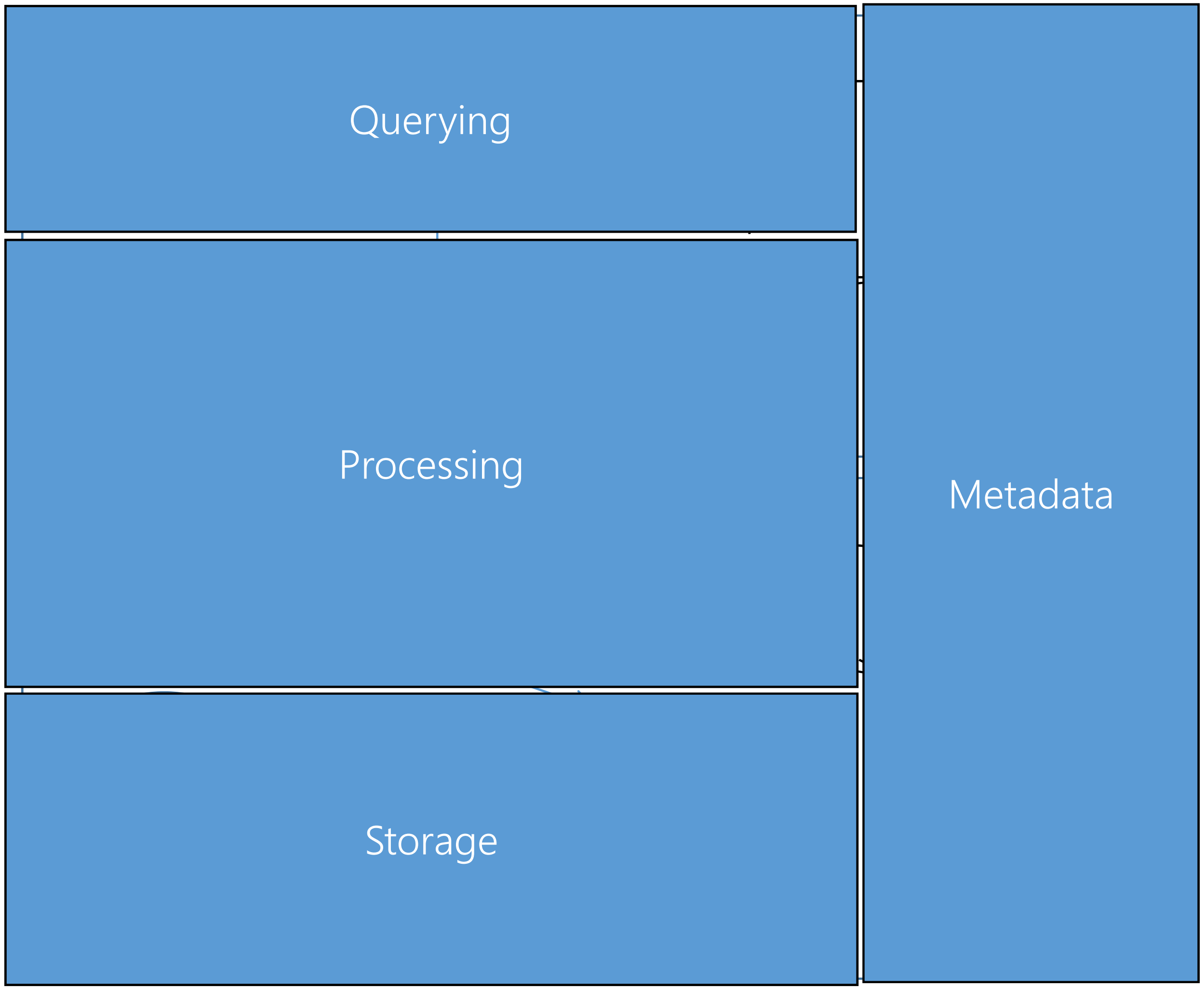
Centralized DBMS Architecture



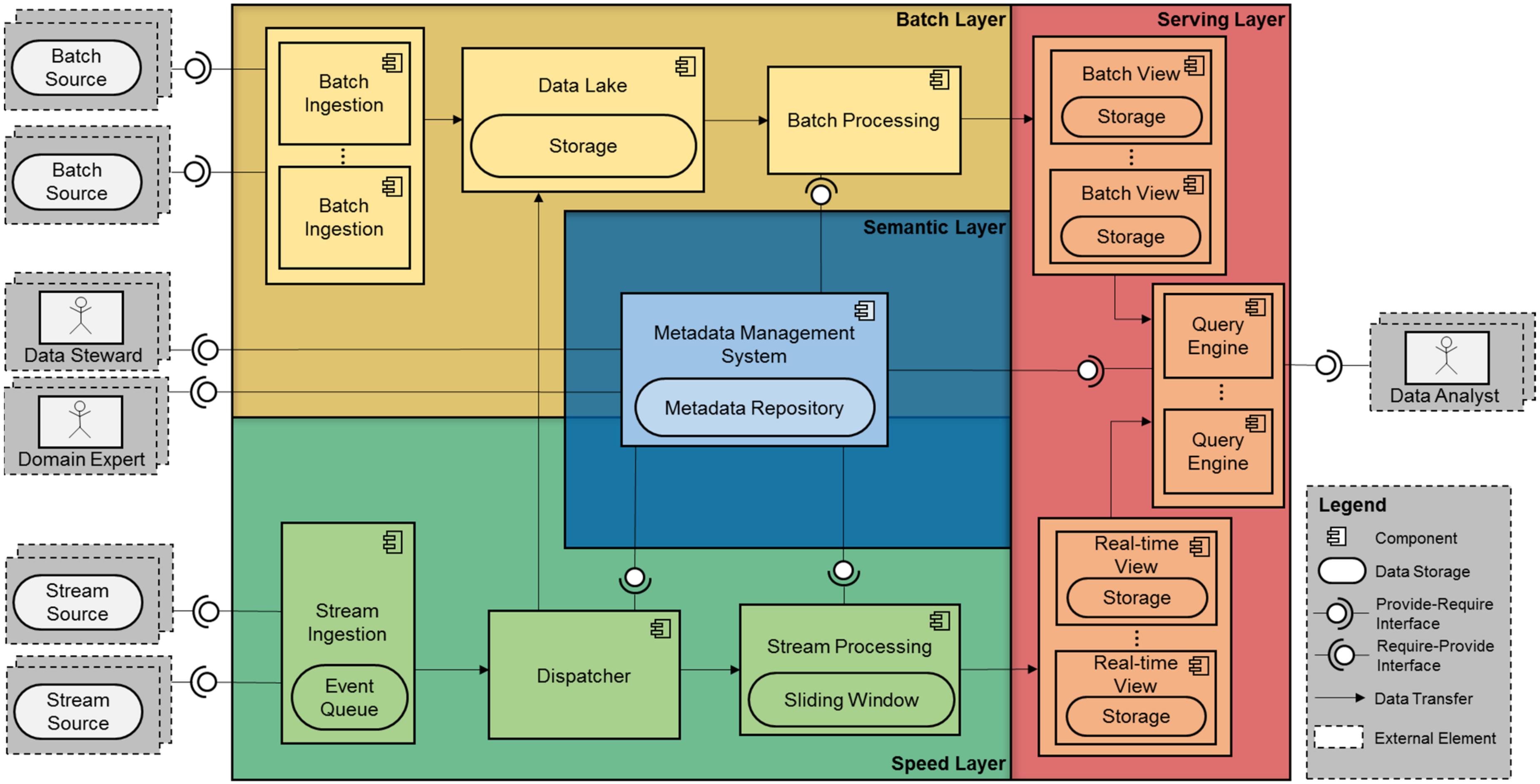
Distributed DBMS Architecture



Distributed DBMS Architecture

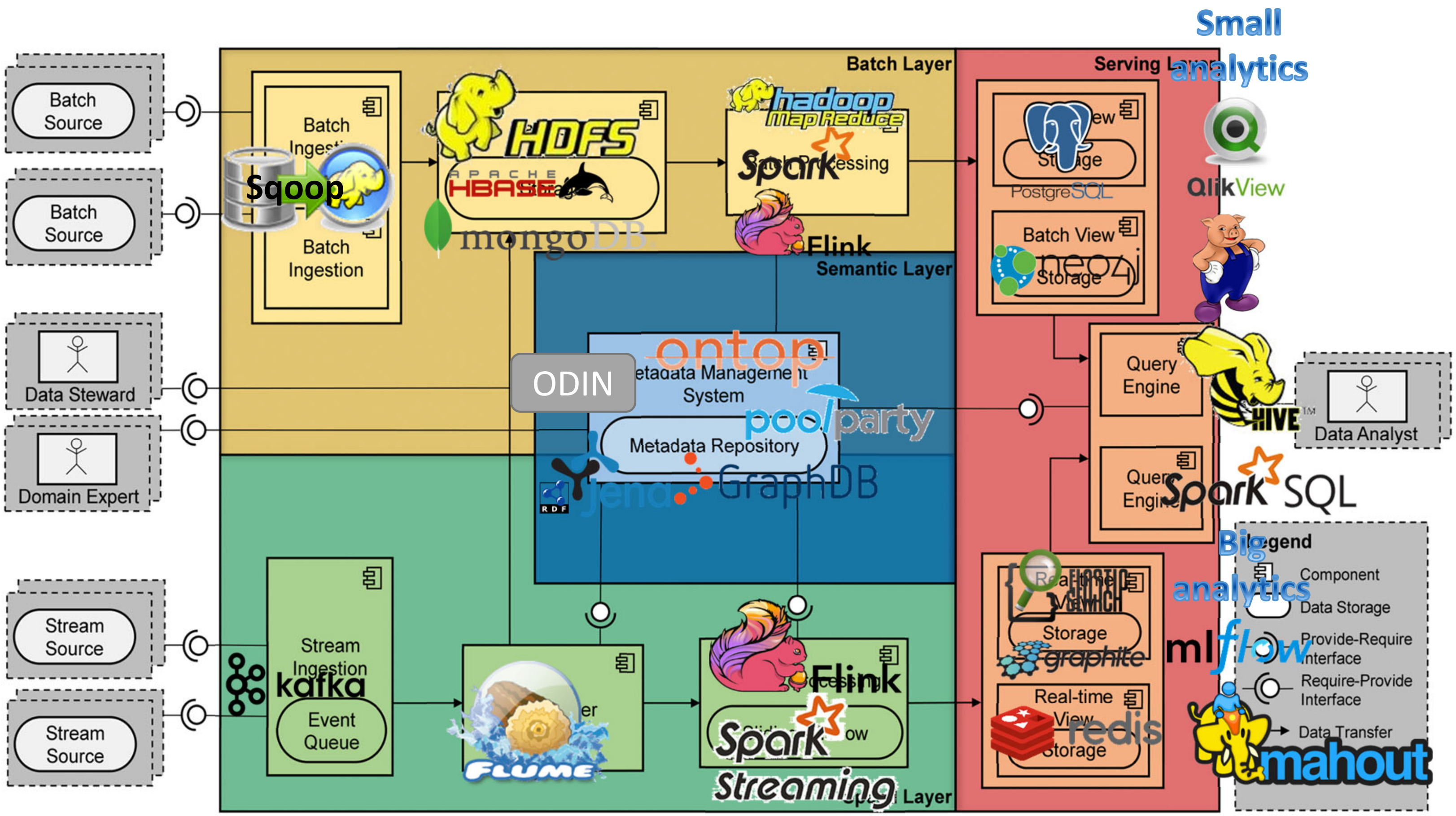


Bolster



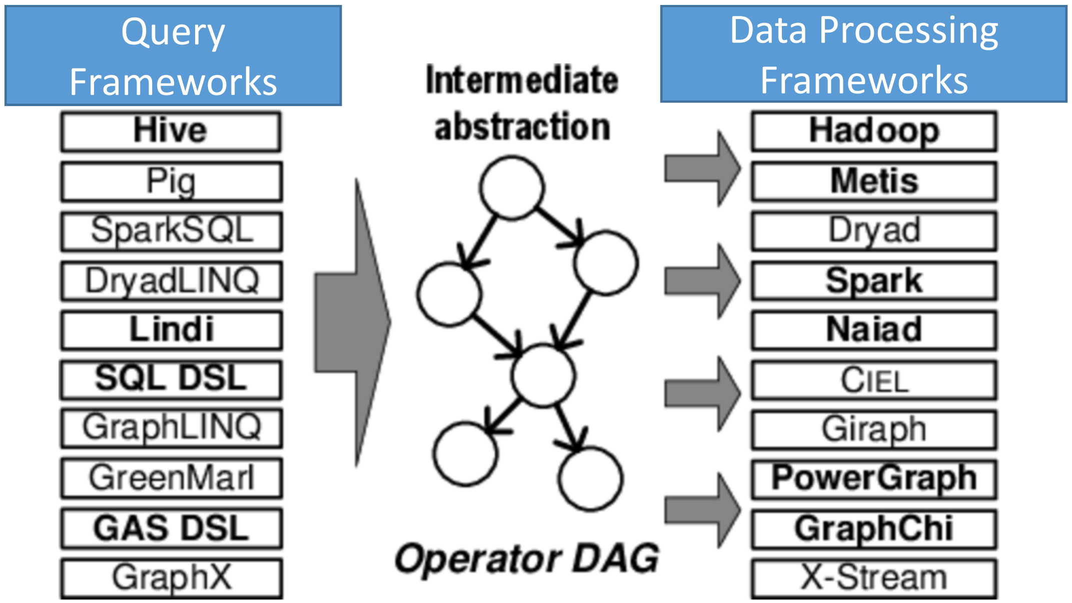
S. Nadal et al.

Bolster Instantiation



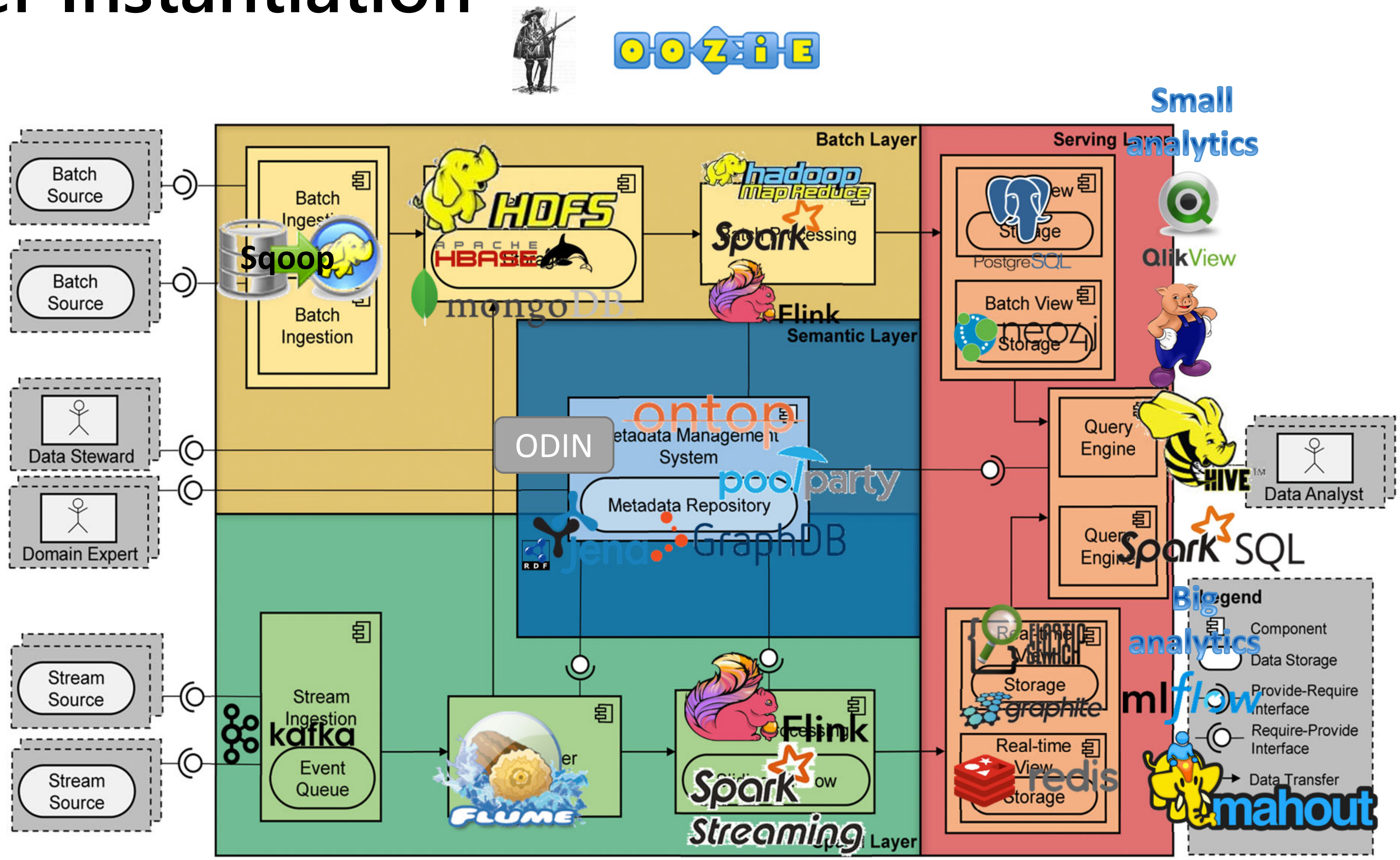
Workflow Orchestrators

- Current workflow orchestrators are rather poor: Oozie
- But there are attempts for smarter approaches: the ideas behind Musketeer deserves special attention

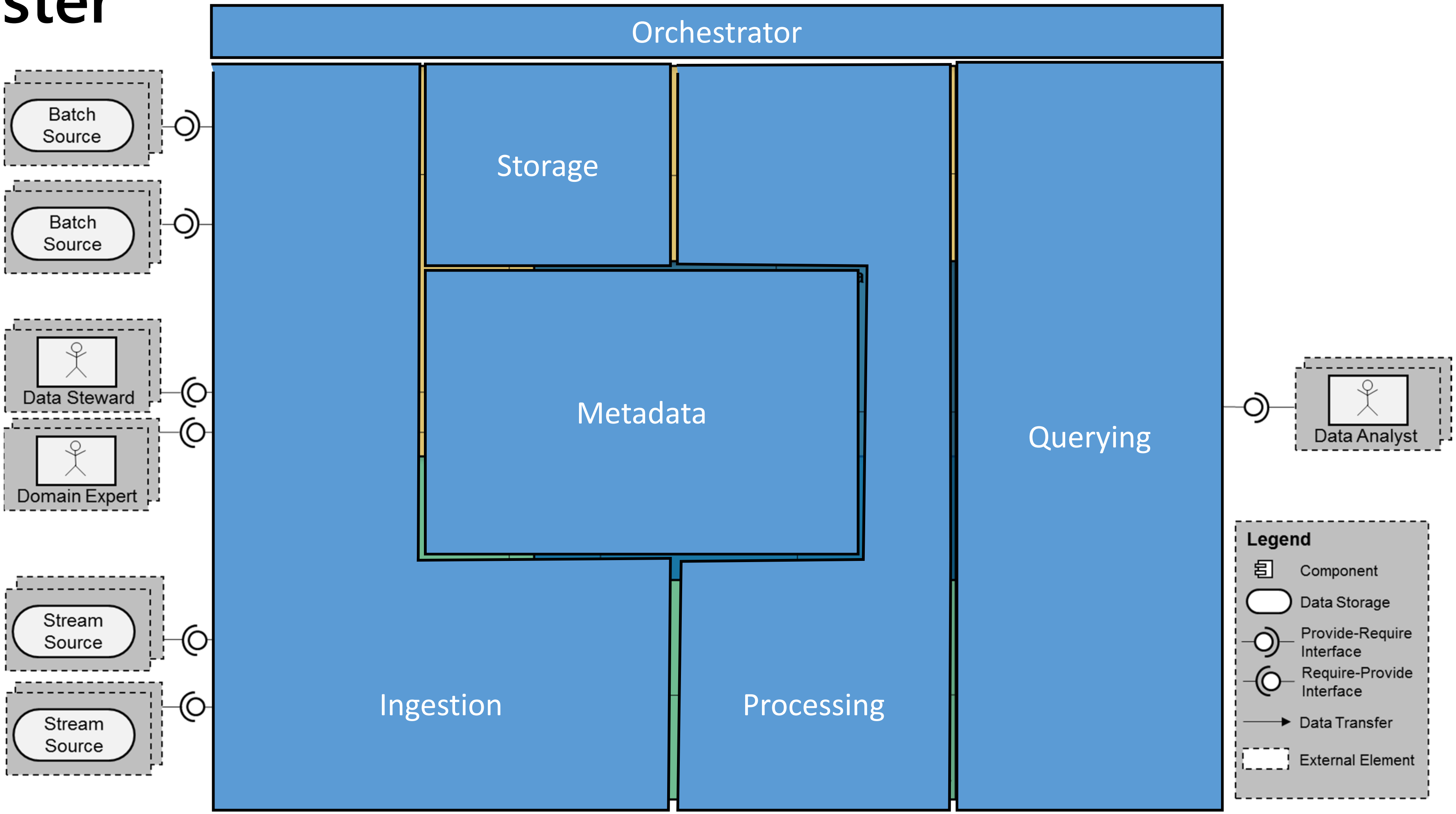


Shortly, does a similar job to global query optimizers of traditional distributed RDBMS

Bolster Instantiation



Bolster



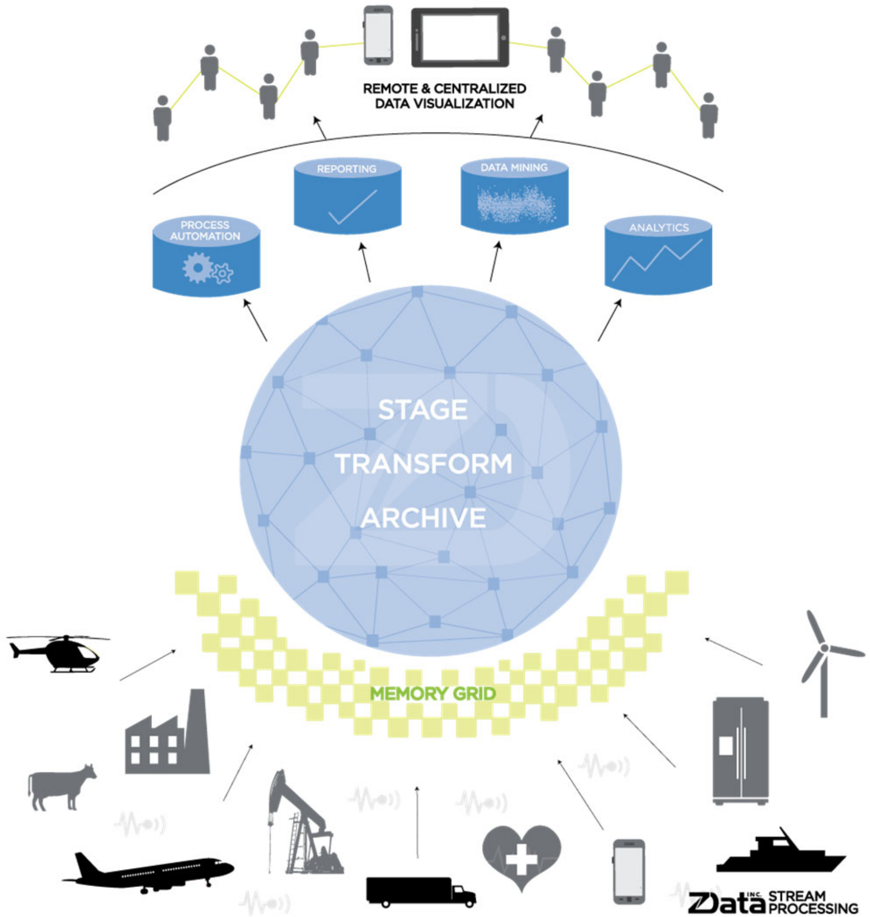
S. Nadal et al.

New Storage Architectural Pattern

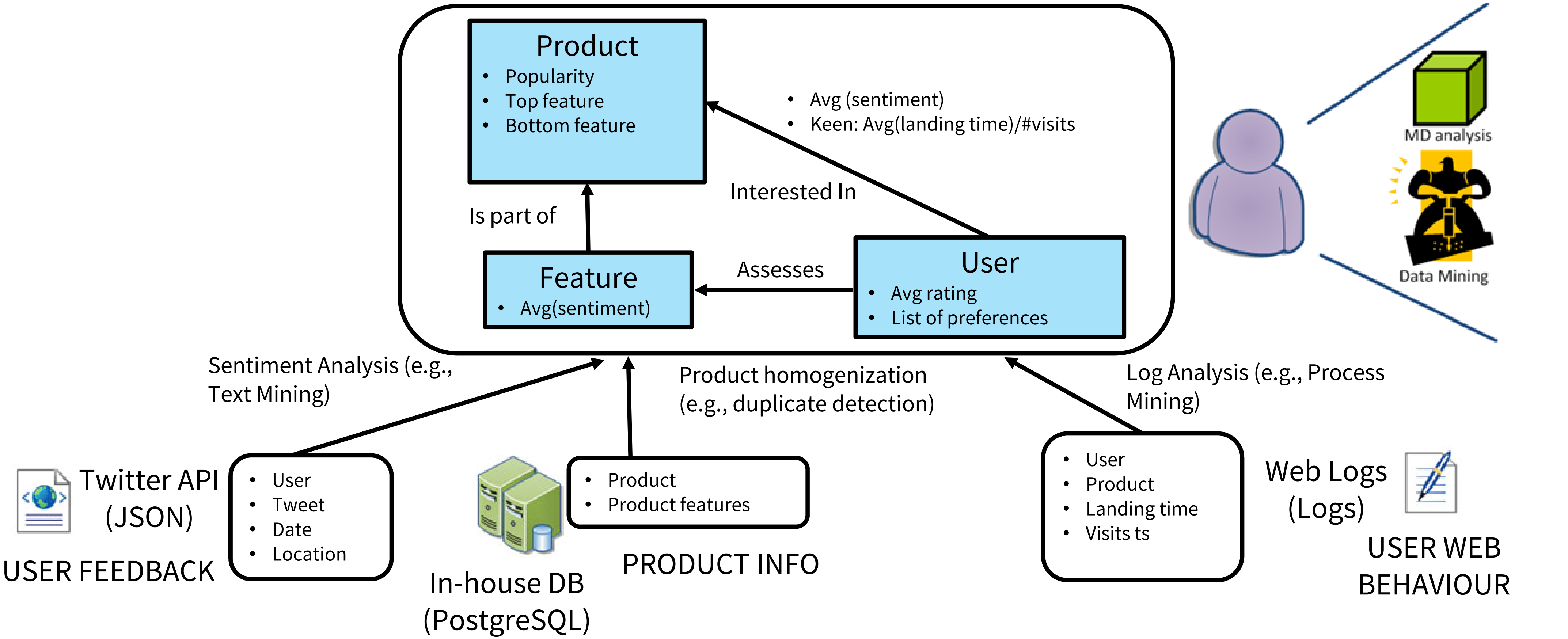
From data warehousing to data lakes

The Data Lake

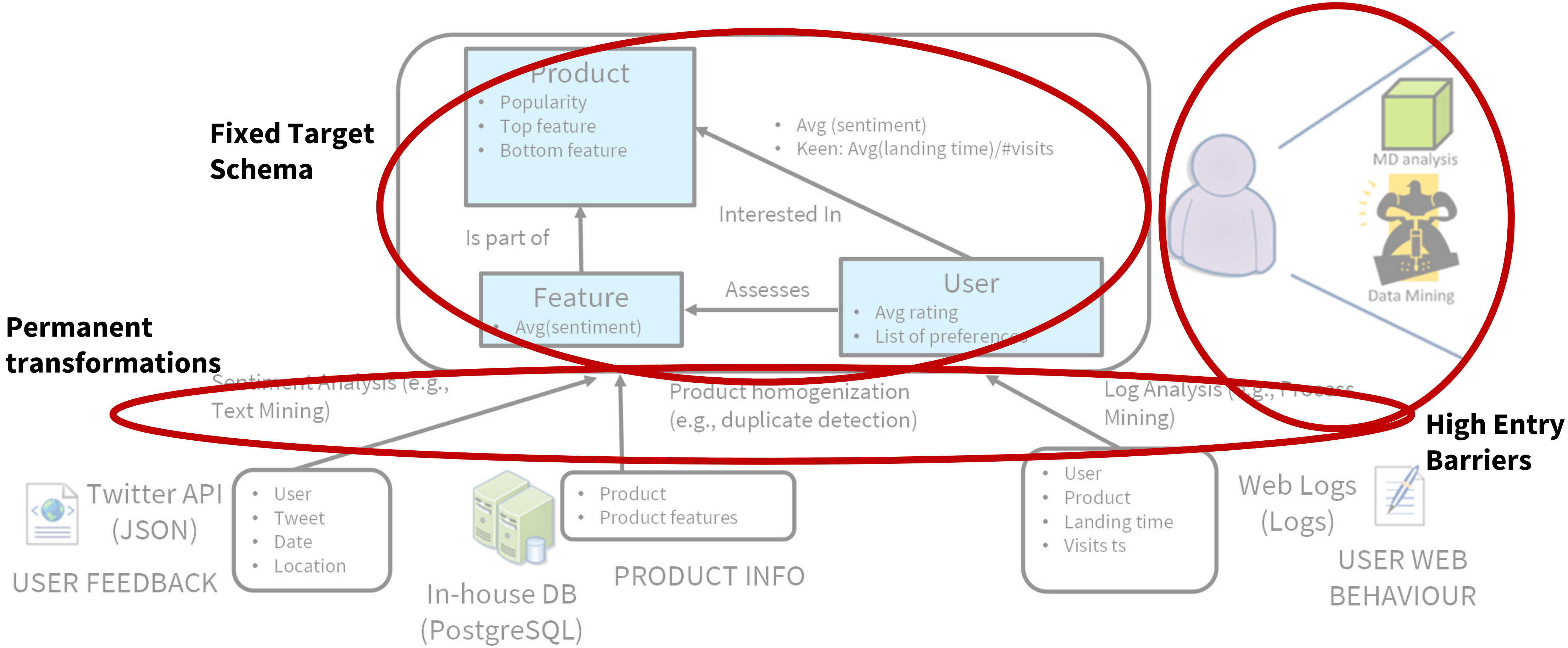
- IDEA: Load-first, Model-Later
- Modeling at load time restricts the potential analysis that can be done later (Big Analytics)
- Store raw data and create on-demand views to handle with precise analysis needs



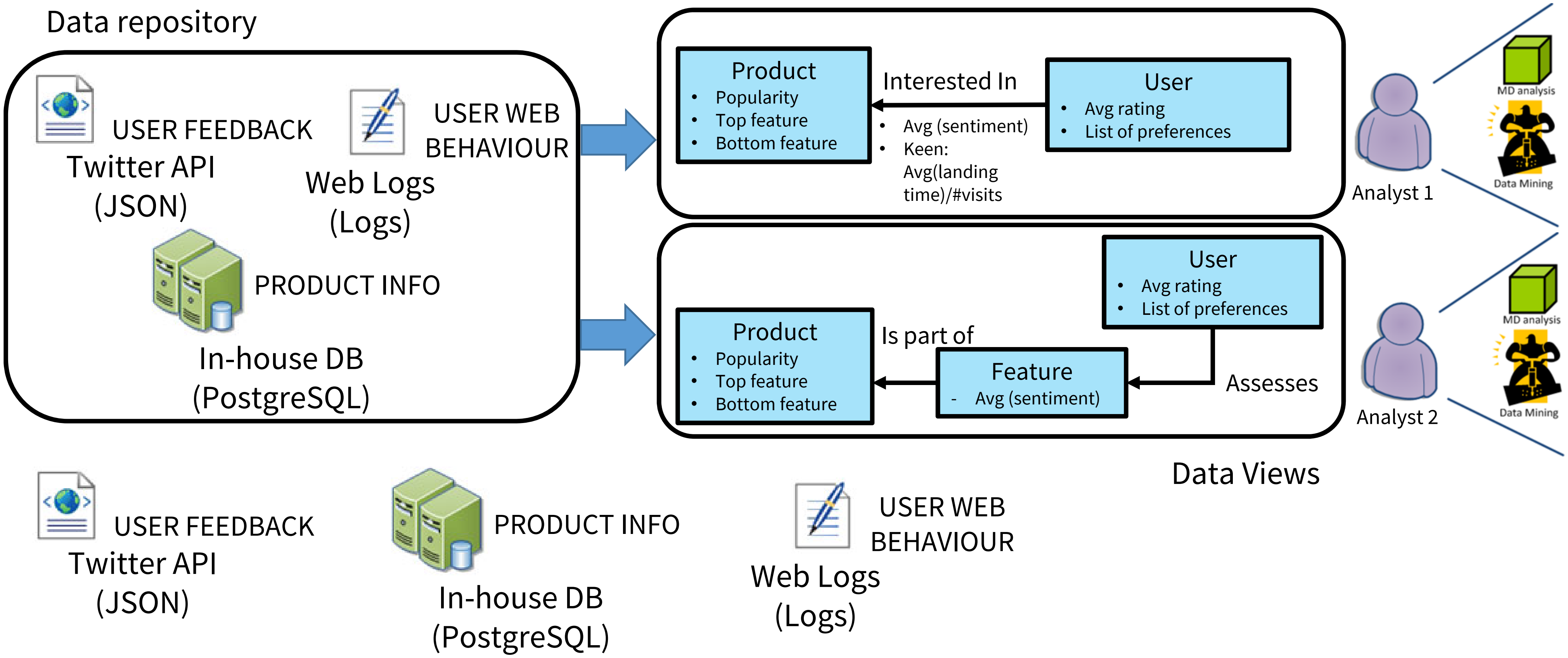
Model-First (Load-Later)



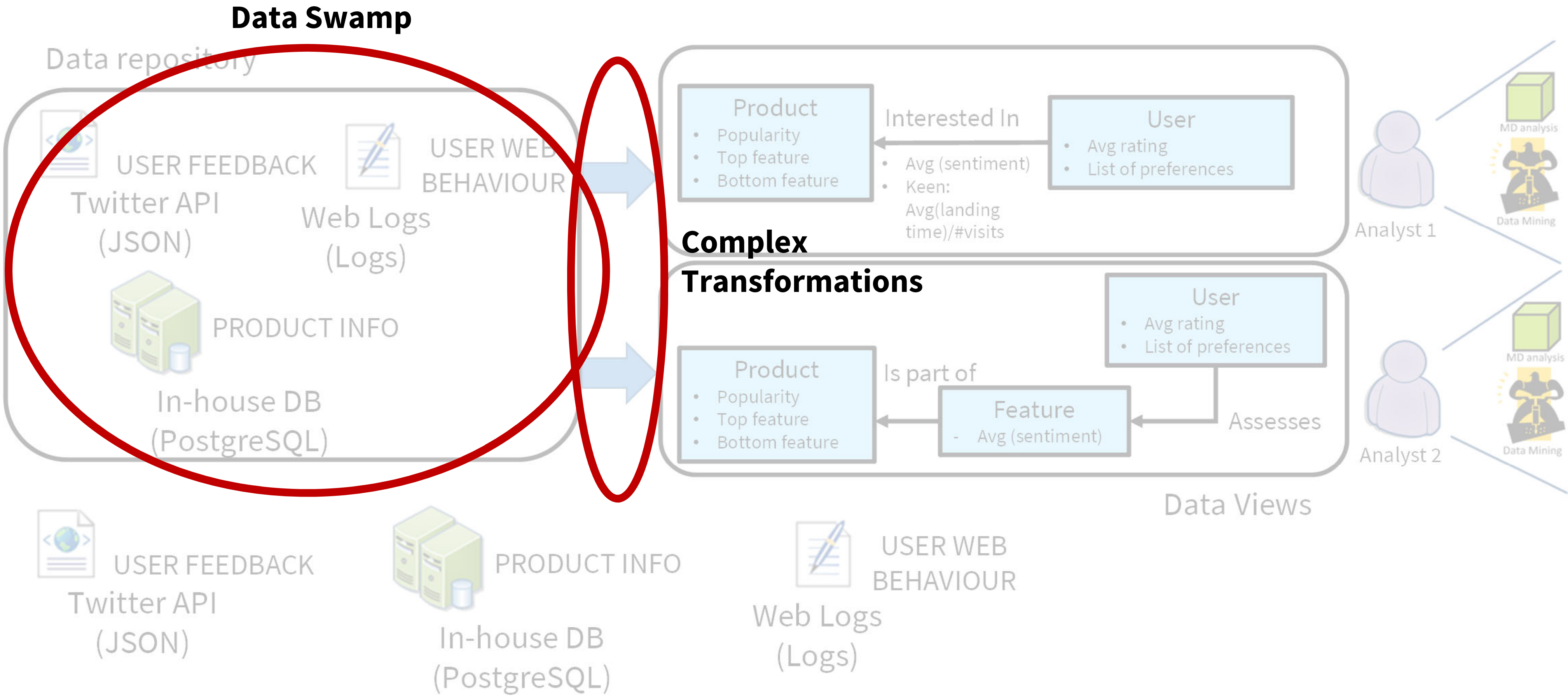
Drawbacks of Model-First (Load-Later)



Load-First (Model-Later)

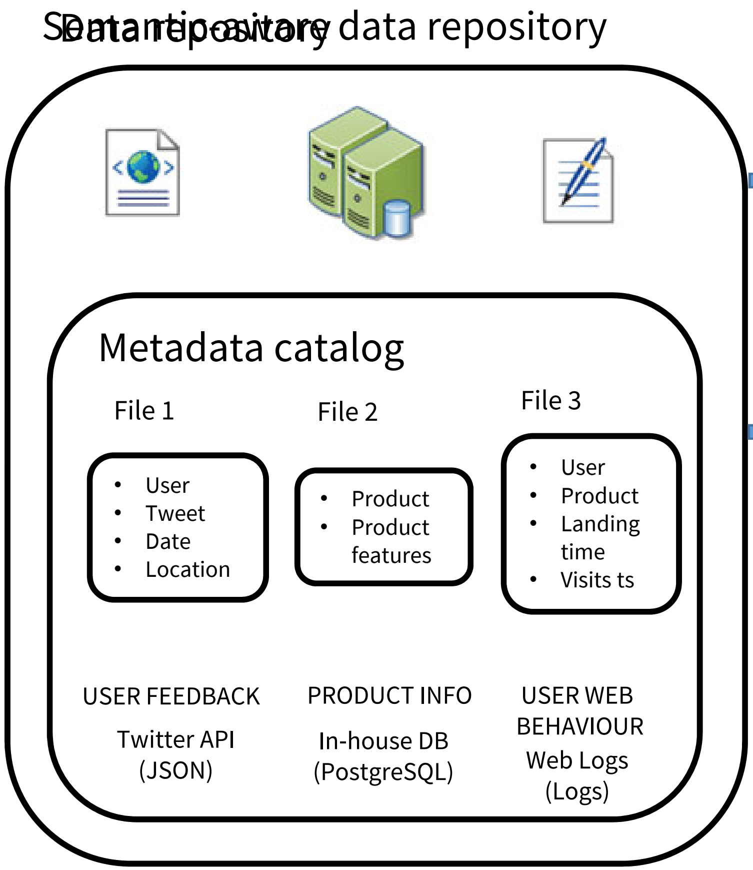


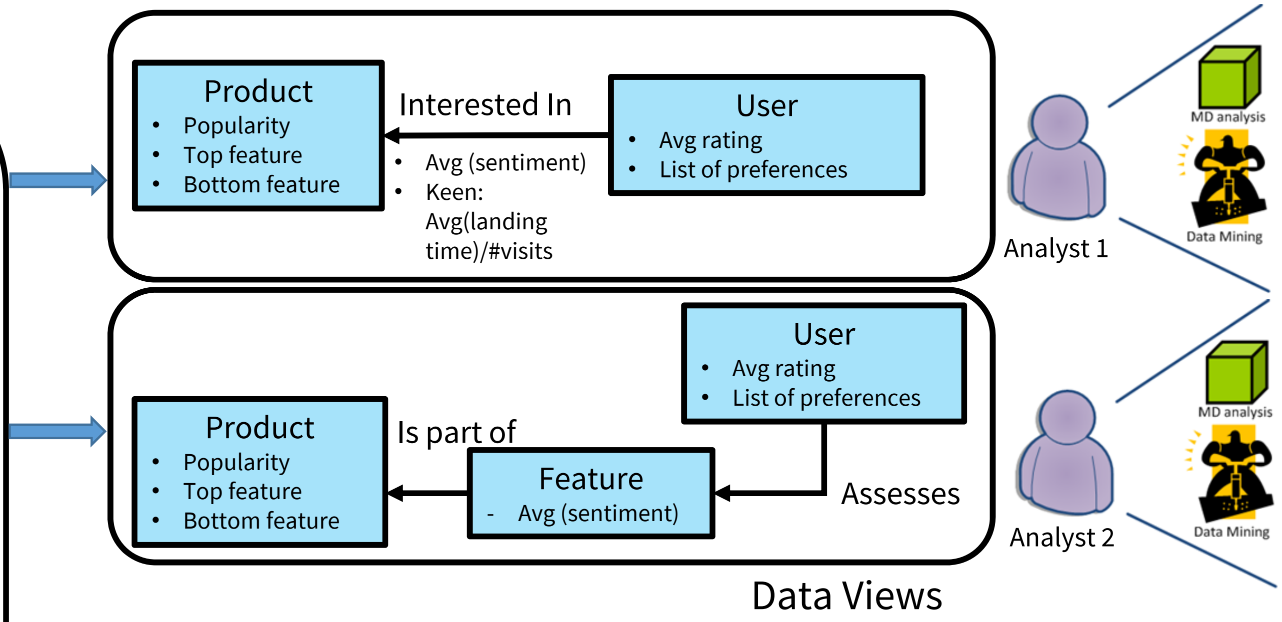
Drawbacks of Load-First (Model-Later)



Stonebraker (2014)

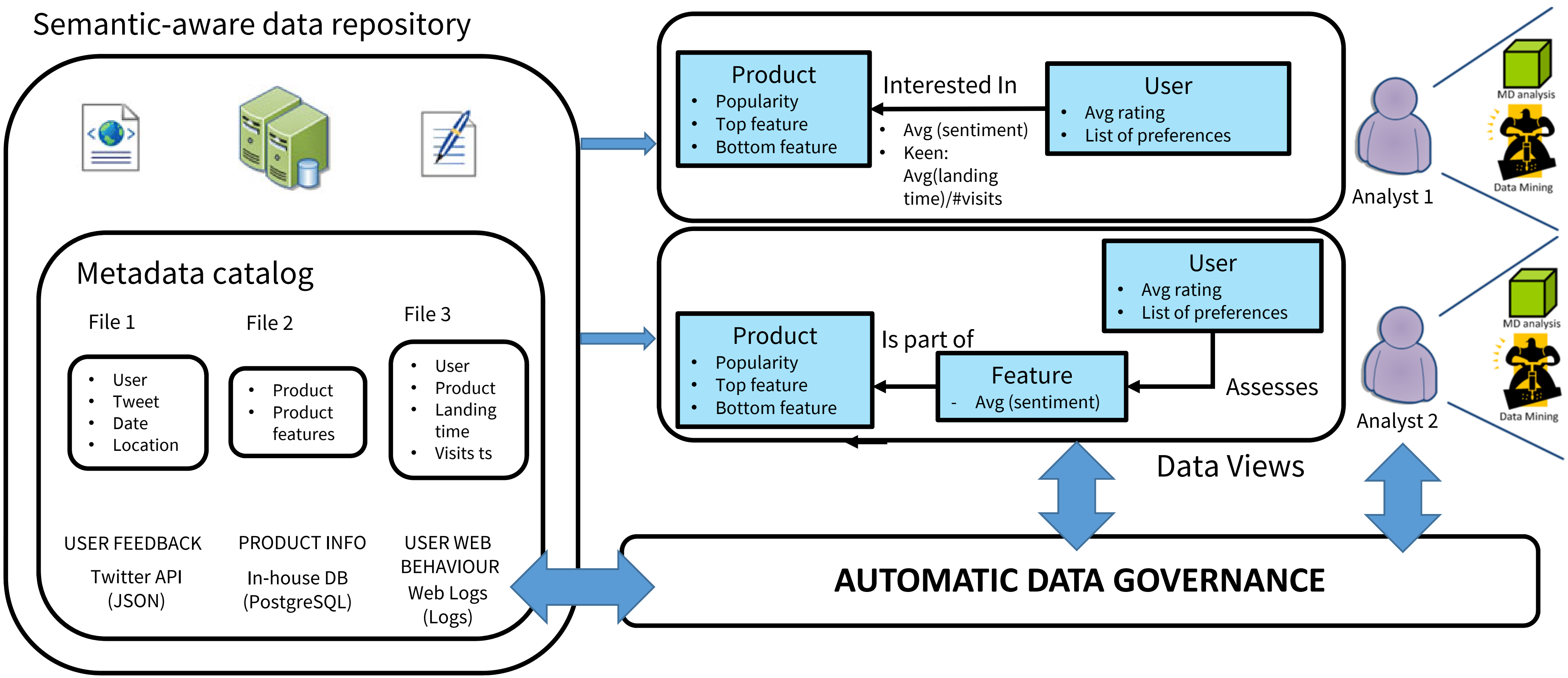
Towards semantic-awareness



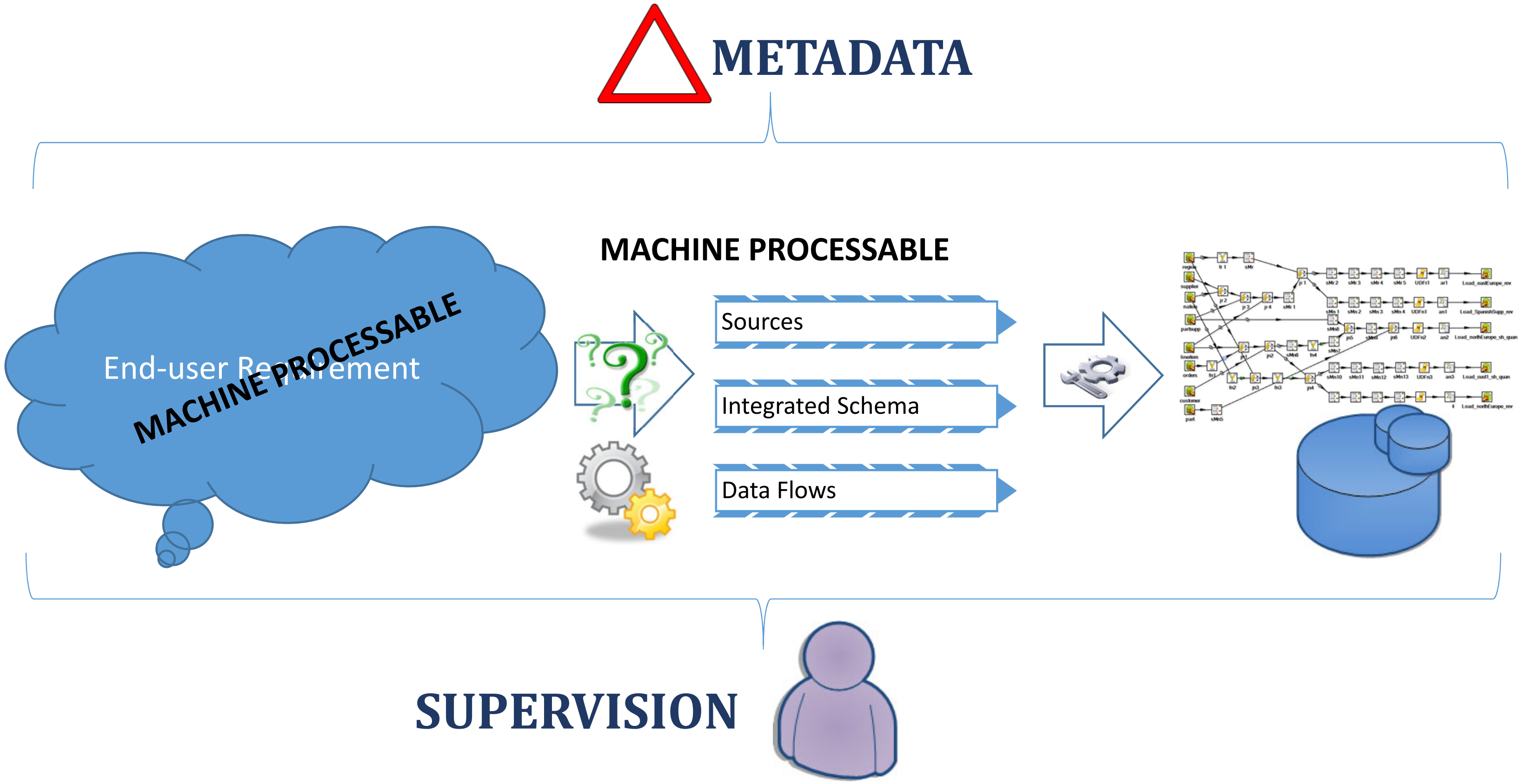


- Metadata provides semantics
  - Source schemata, mappings to views, parsing information, …
- Automation of the integration processes

From IT-Centered to User-Centered

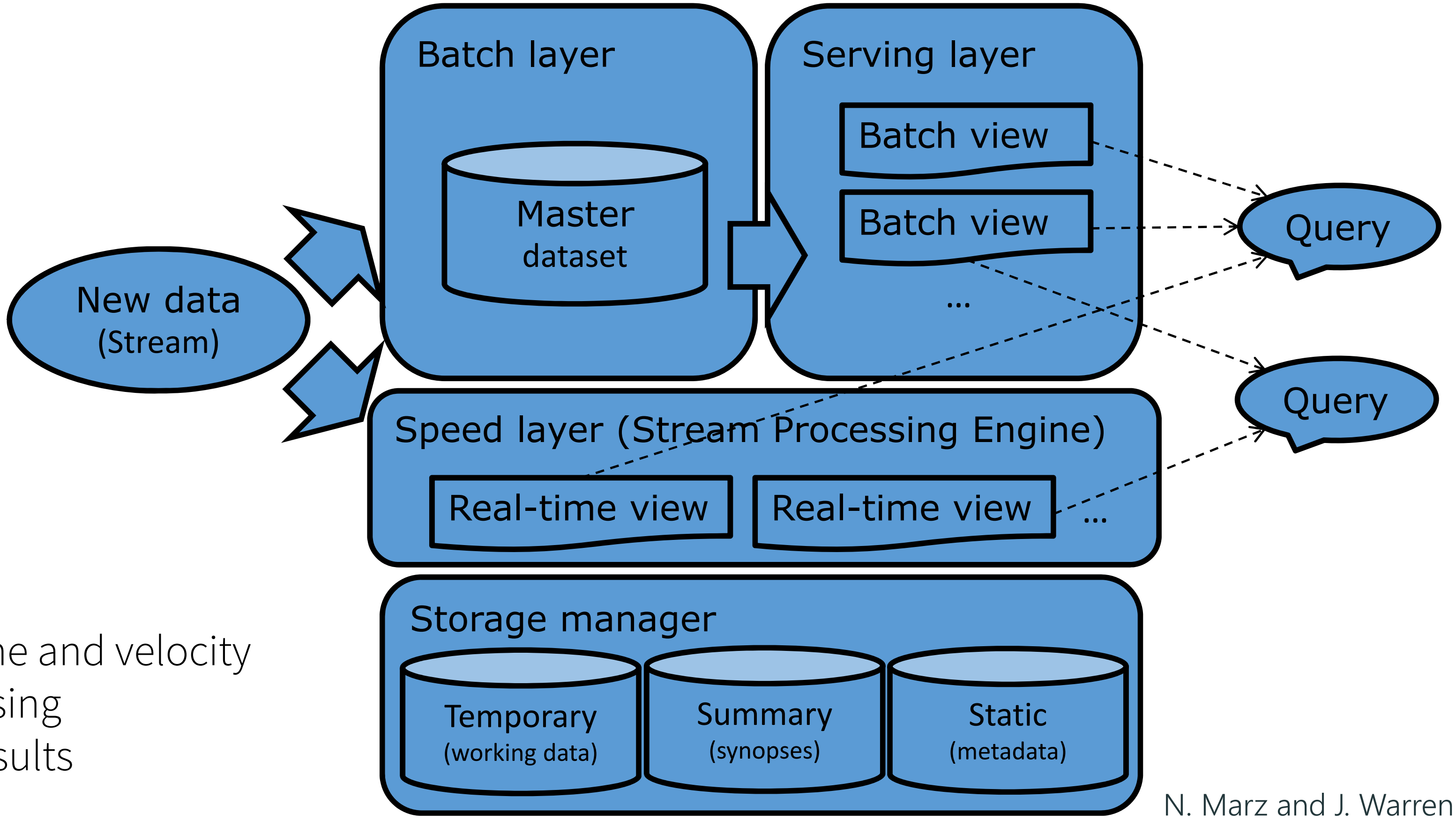


The Missing Link: Metadata



New Processing Architectural Patterns

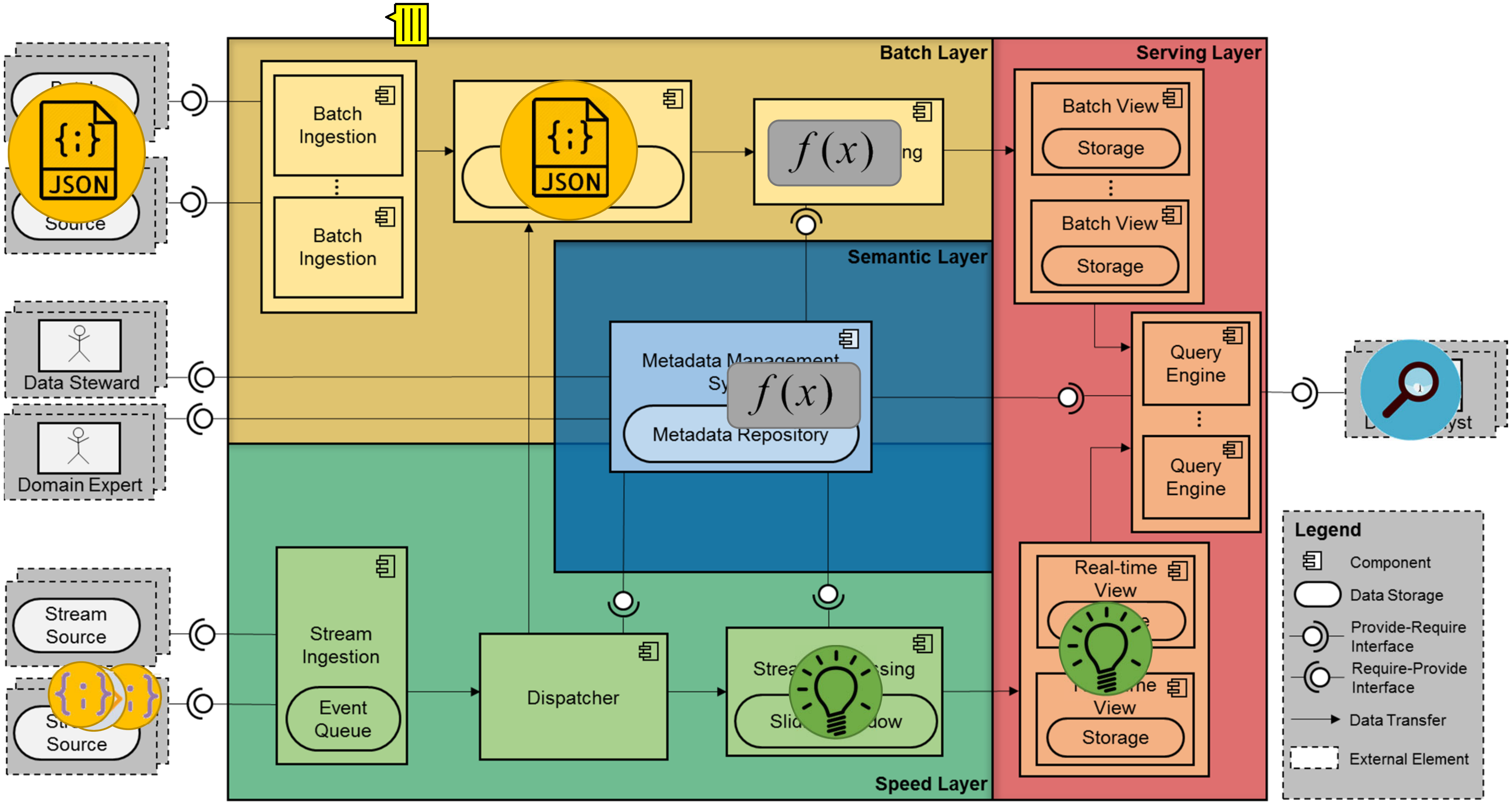
λ-Architecture



IDEA: Accommodate volume and velocity
Real time Vs. Batch processing
Precise Vs. Approximate results

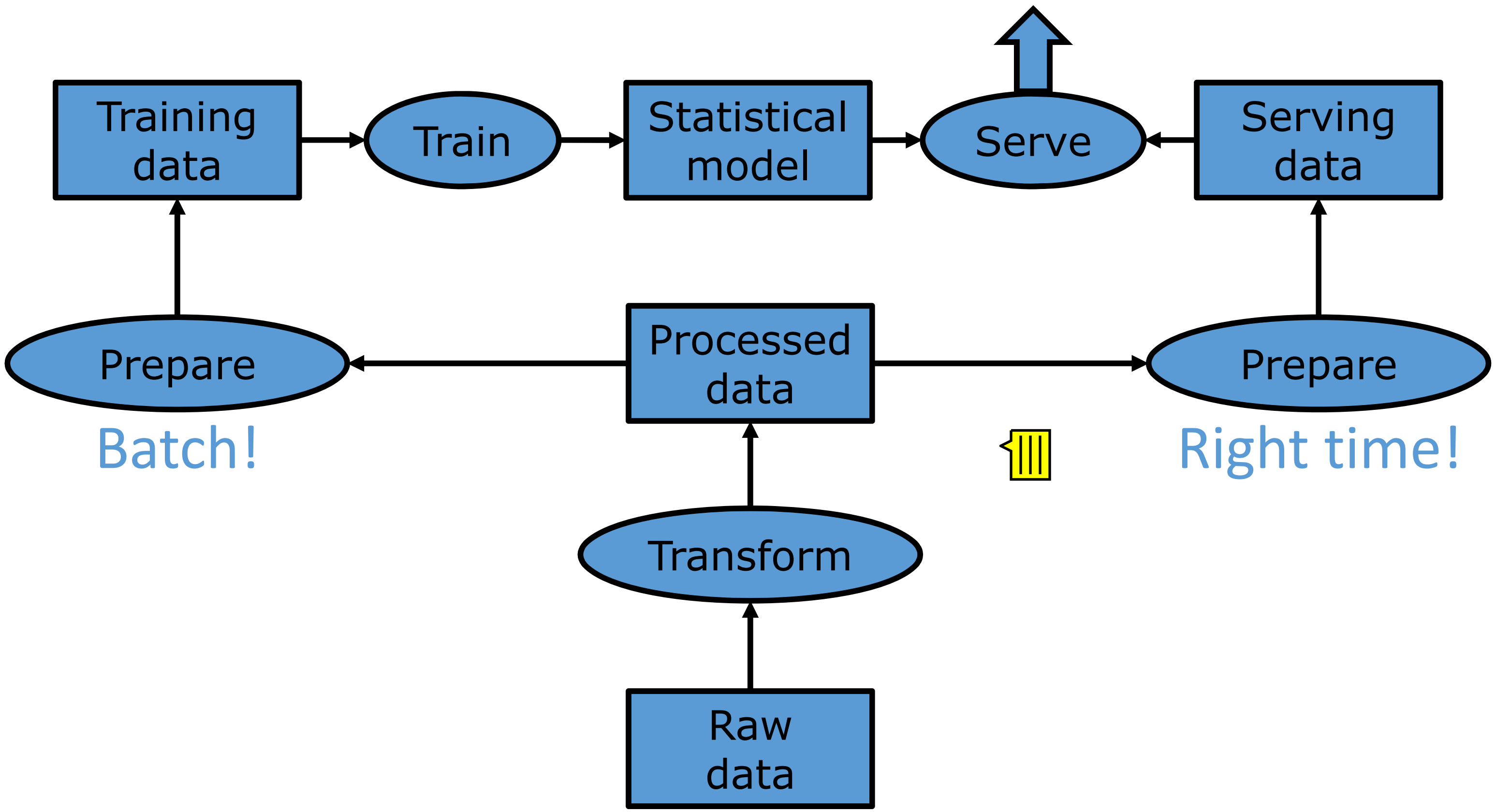
N. Marz and J. Warren

Bolster



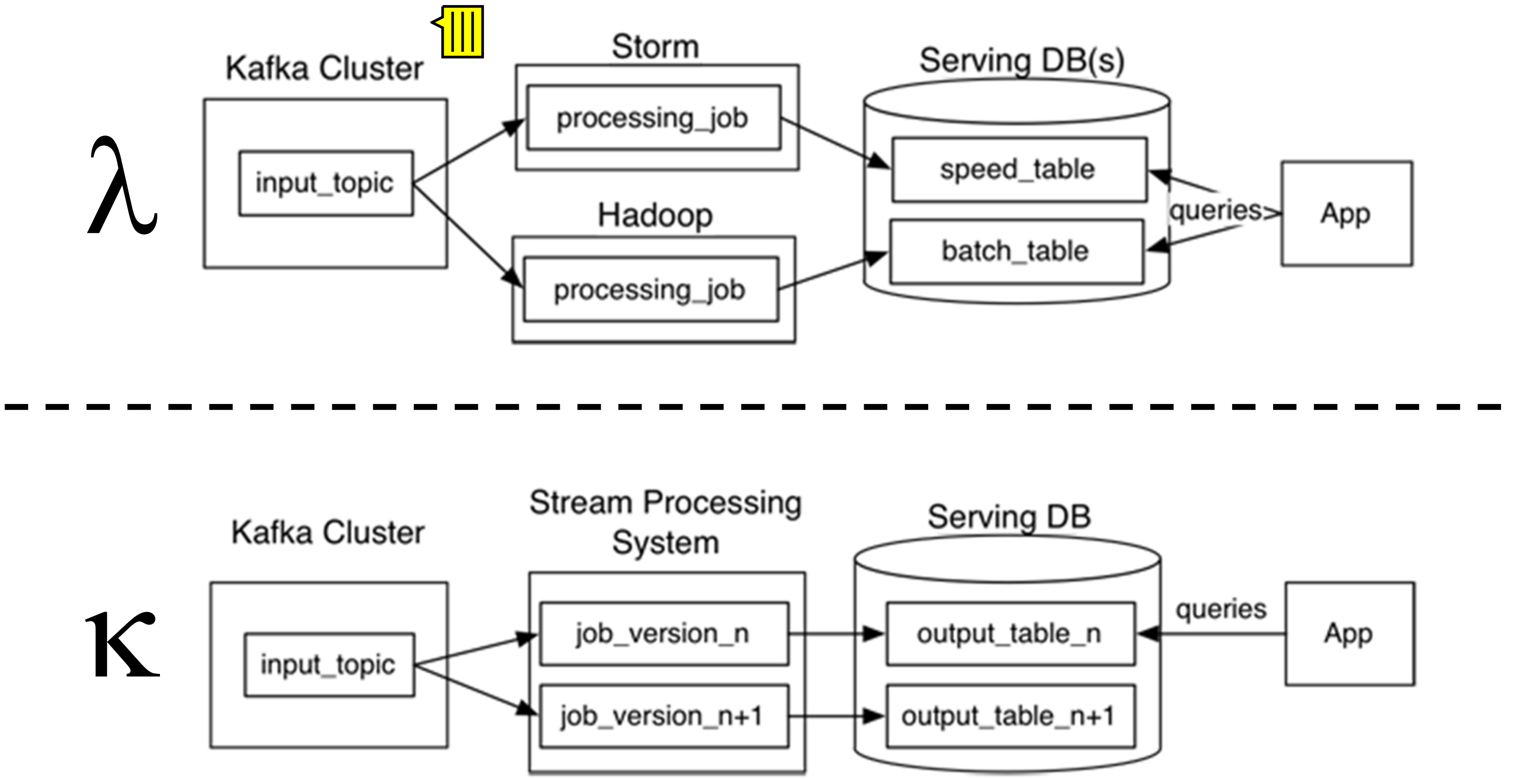
Batch Layer
Serving Layer
Semantic Layer
Speed Layer
Batch Ingestion
Batch Ingestion
ng
Batch View
Storage
Batch View
Storage
Data Steward
Domain Expert
Metadata Management
Metadata Repository
Query Engine
Query Engine
Stream Source
Source
Stream Ingestion
Event Queue
Dispatcher
Real-time View
View
Storage
Legend
Component
Data Storage
Provide-Require Interface
Require-Provide Interface
Data Transfer
External Element

Data-centered architecture



κ-architecture

- Data is considered to be a never-ending stream



Jay Kreps and O'Reilley

Closing

Summary

- New architectural solutions
  - Lambda
  - Kappa
  - Polyglot systems
- Data Lakes
  - The need of metadata
- Reference architectures
  - Bolster
  - Quarry

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
- 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
- P. Jovanovic et al. *Quarry: A User-centered Big Data Integration Platform*. Information Systems Frontier, 2021

Resources

- http://hadoop.apache.org
- http://www.cloudera.com
- http://hortonworks.com