Big Data Architectures

Big Data Management

Knowledge objectives

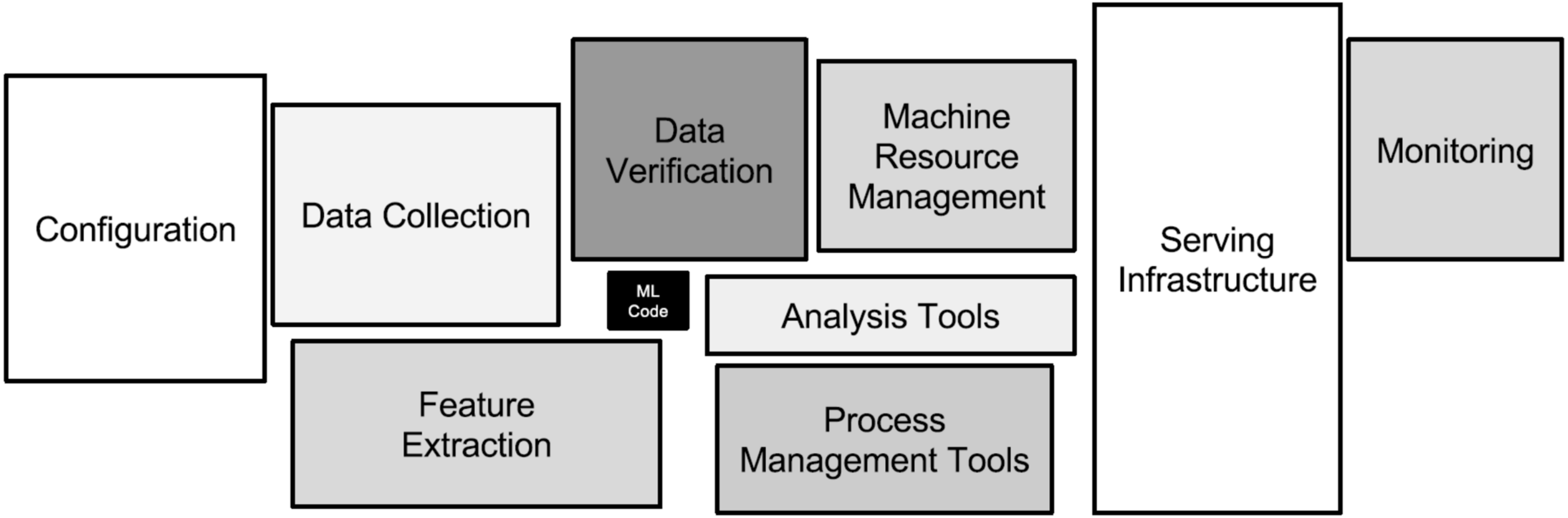
1. Explain the problema of a spaghetti architecture
2. Explain the need of the Lambda architecture
3. Explain the difference between the Kappa and Lambda architectures
4. Justify the need of a Data Lake
5. Identify the difficulties of a Data Lake
6. Explain the need of each component in the Bolster architecture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

Application Objectives

1. Given a use case, define its software architecture

Problem definition

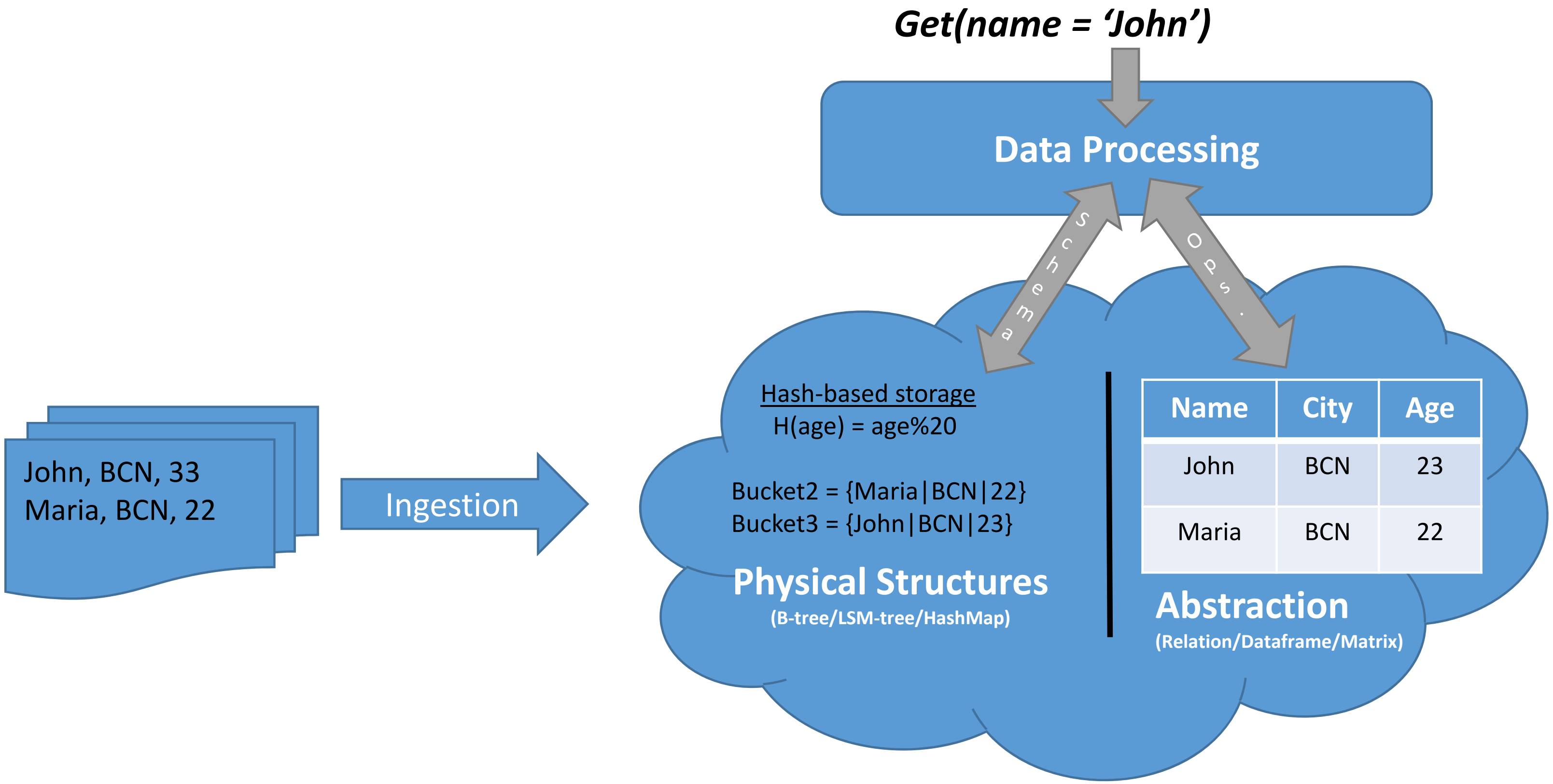
Vast and Complex surrounding infrastructure



- ML code is just a small box in the middle
  - … with a lot of plumbing around it

D. Sculley te al.

Data Management (I)



Data Management (II)

Data management refers to the functionalities 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

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

Big Data Architectures

- Question the main principles of traditional DB architectures
  - Data can grow beyond limits requiring scale out (a.k.a. Volume)
  - Data is not necessarily persisted (a.k.a. Velocity)
  - Data structure is neither known a priori, nor fixed (a.k.a. Variety and Variability)
- 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
- Implement from scratch the whole stack
  - Ingestion, Storage, Modeling, Processing, and Querying

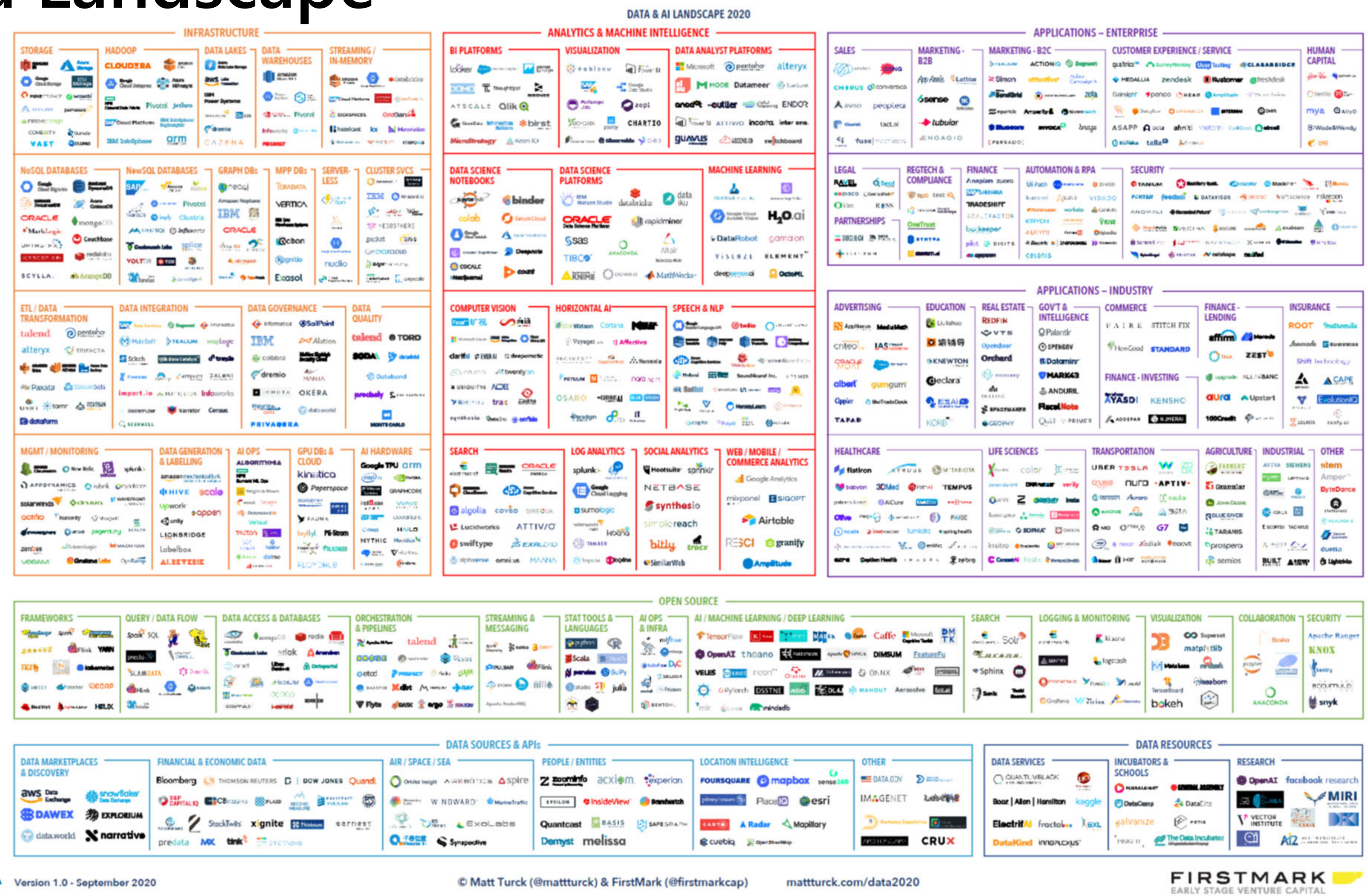
The Multi-Project Approach

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

Orchestrator
1. **Oozie is simple**
2. **Musketeer is a bit complex**
3. **Its role would be similar to that of a global execution manager.**

Ingestion - Sqoop, Kafka, Flume
Storage - HDFS+HBASE, mongodb
Processing - Spark, Flink, Spark Streaming, hadoop
metadata - hcatalog, ontop, graphdb, poolparty, jeena
Querying - Postgres, SparkSQLL, Neo4j, redis, graphite
Analytics - QlikView, HIVE, SparkSQL, Big analytics, ml flow, mahout

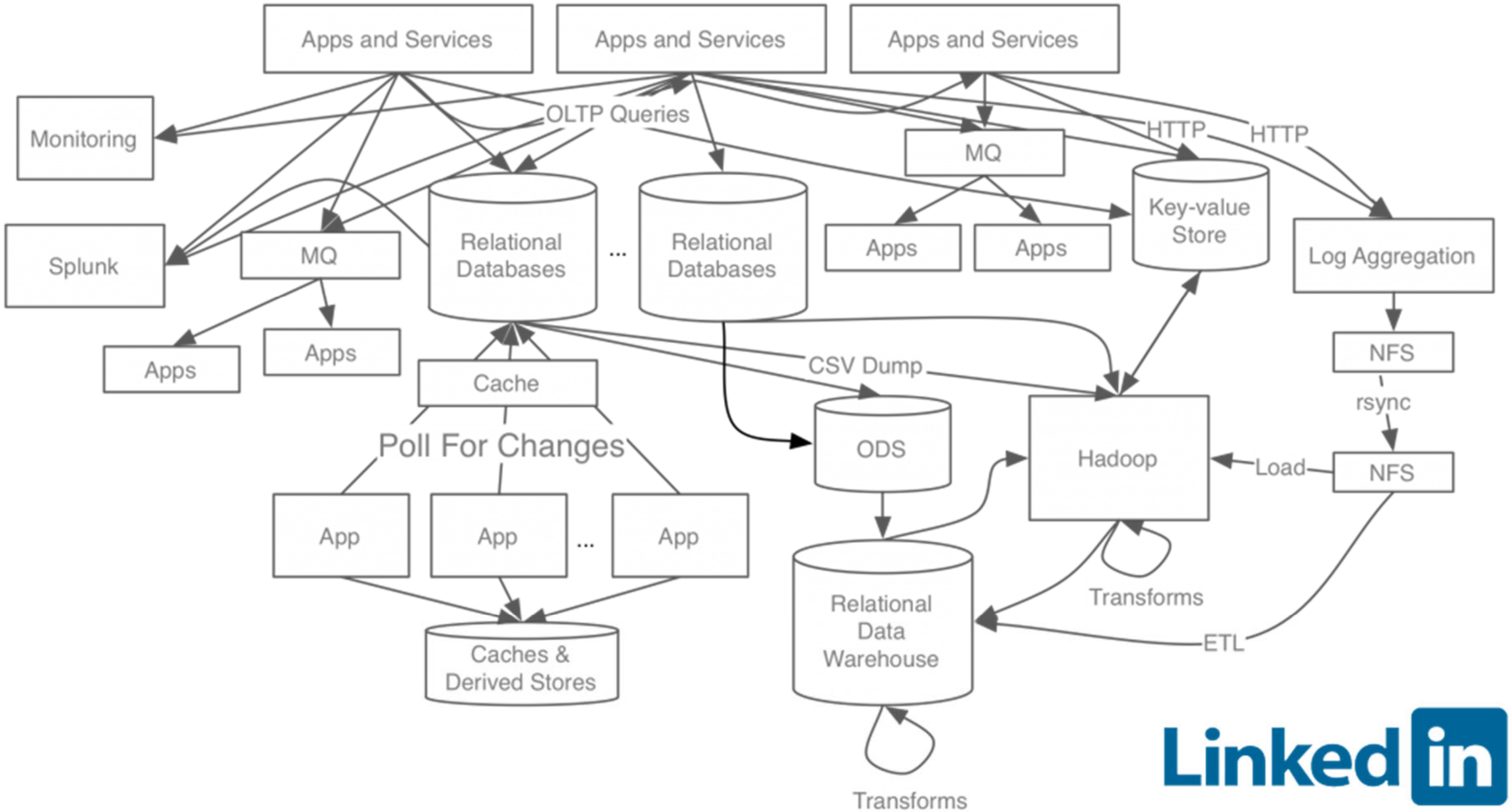
Big Data Landscape



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

www.essi.upc.edu/dtim

Spaghetti architecture



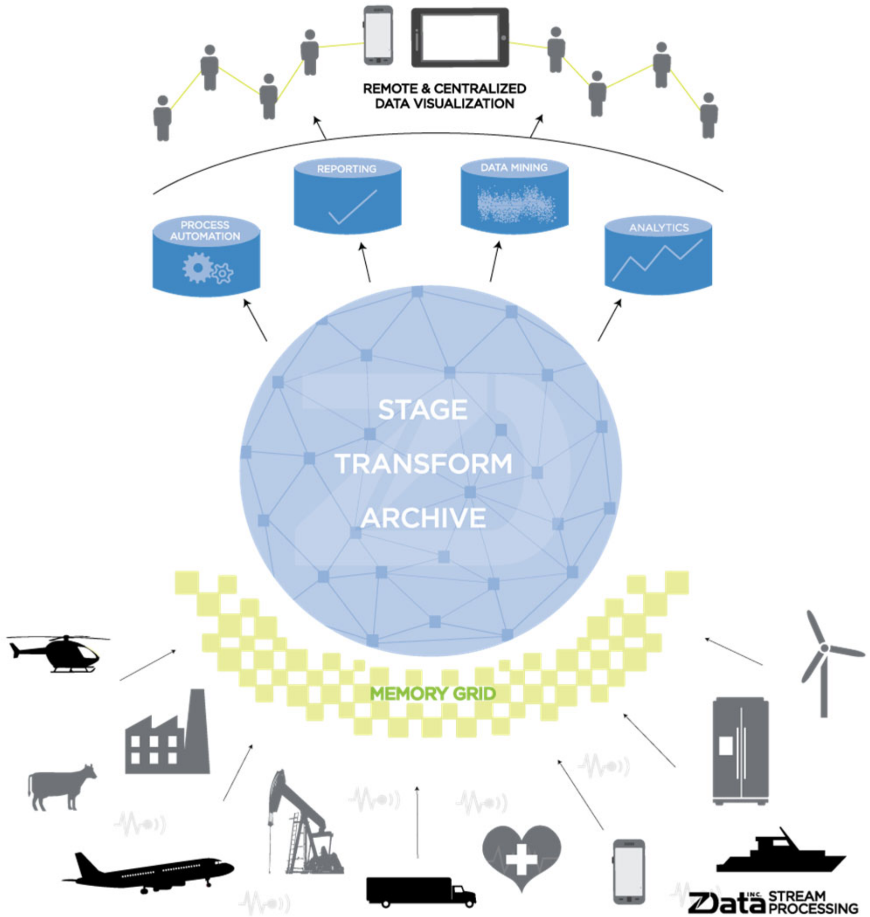
https://www.confluent.io/blog/event-streaming-platform-1

New Storage Architectural Pattern

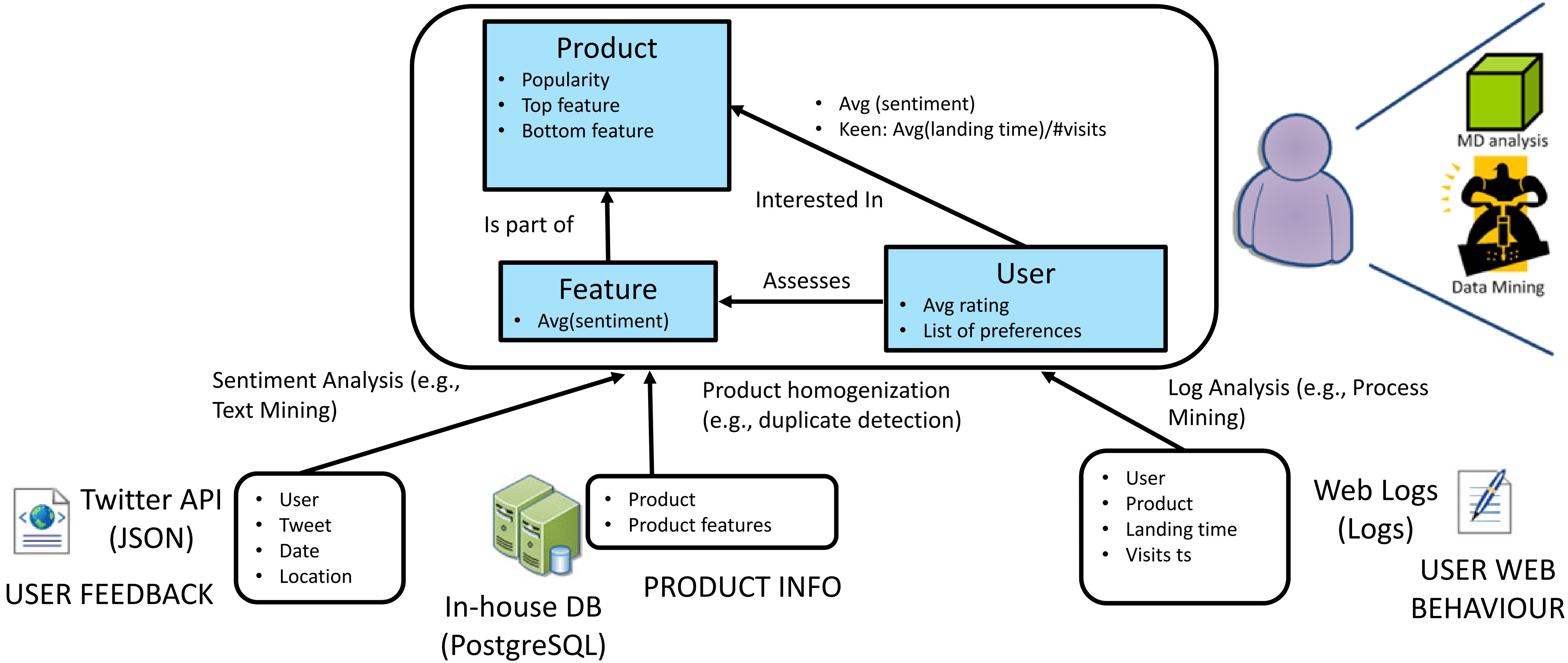
From data warehousing to data lakes

The Data Lake

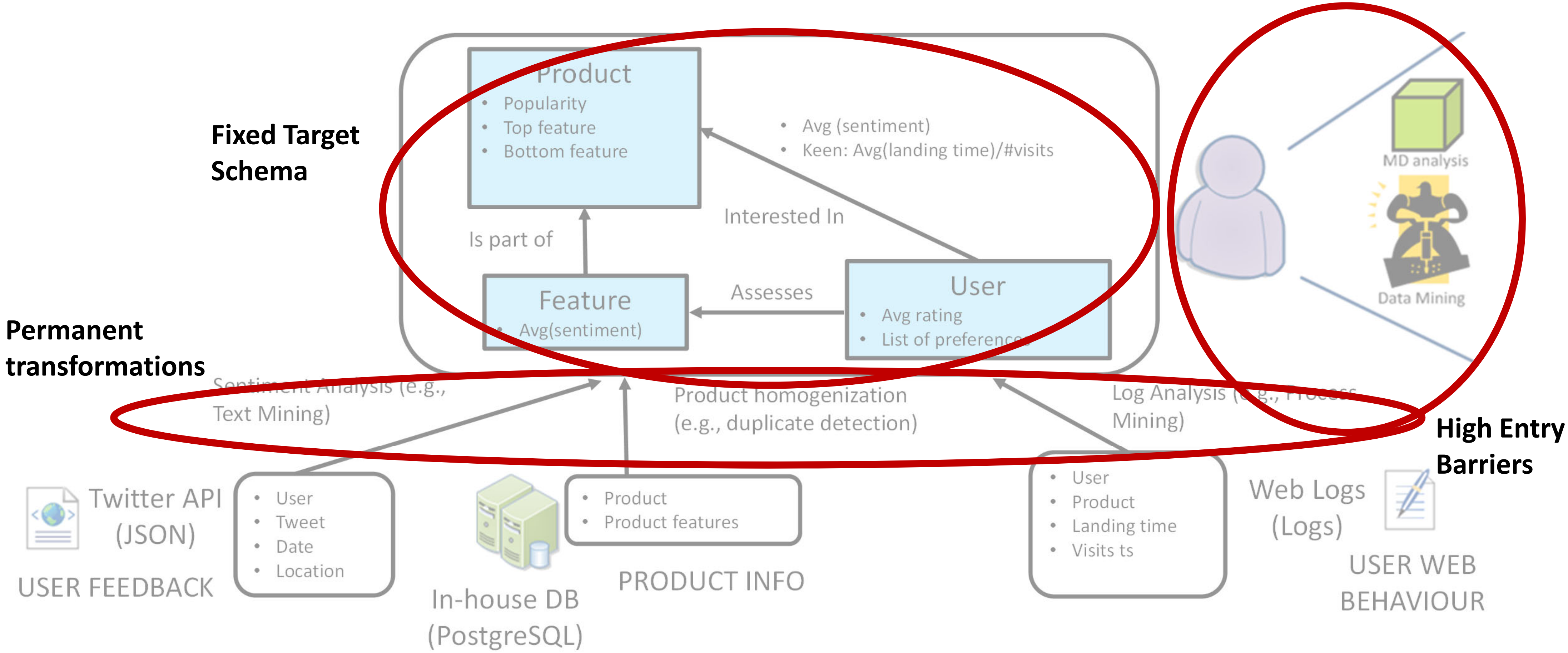
- Idea: Load-First, Model-Later
- Modelling at load time restricts the potential analysis that can be done later (Big Analytics)
- Characteristics:
  a) Store raw data
  b) Provide governing functionalities
  c) Create on-demand views to handle precise analysis needs



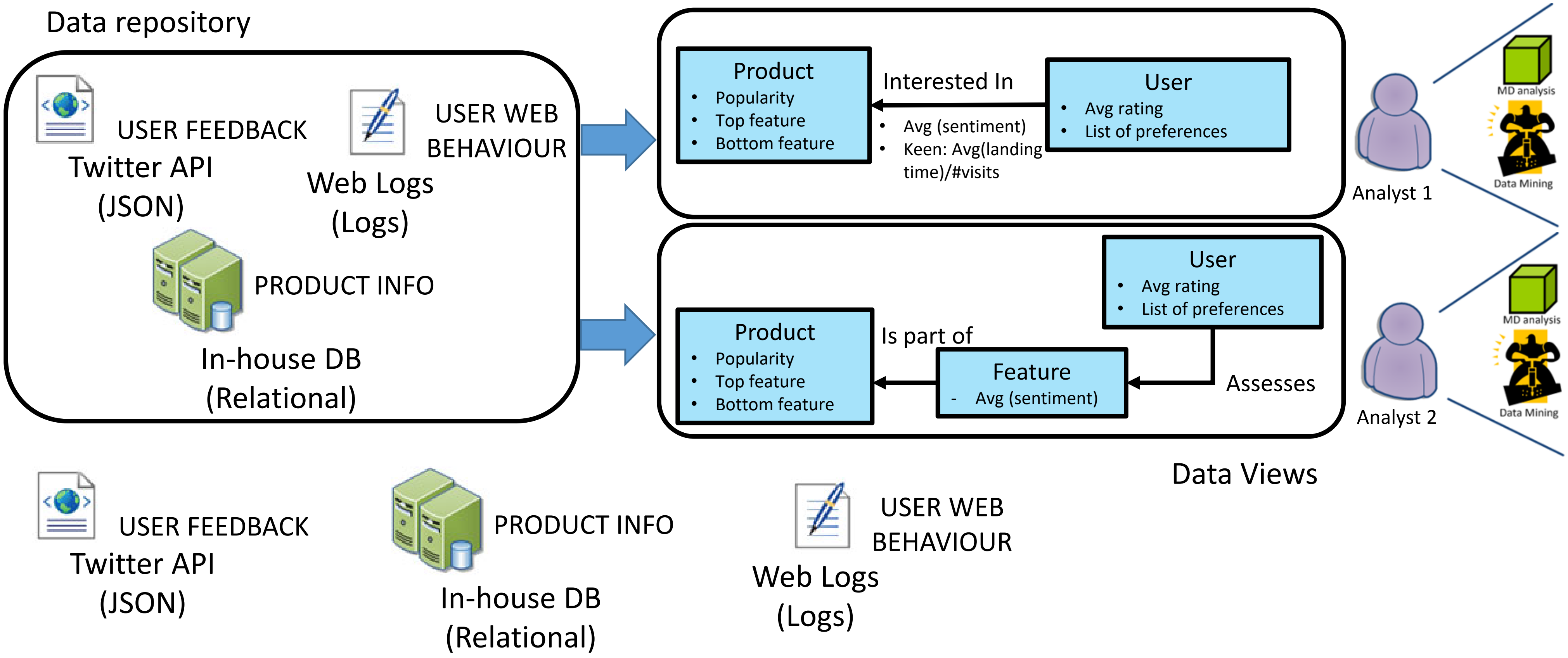
Model-First (Load-Later)



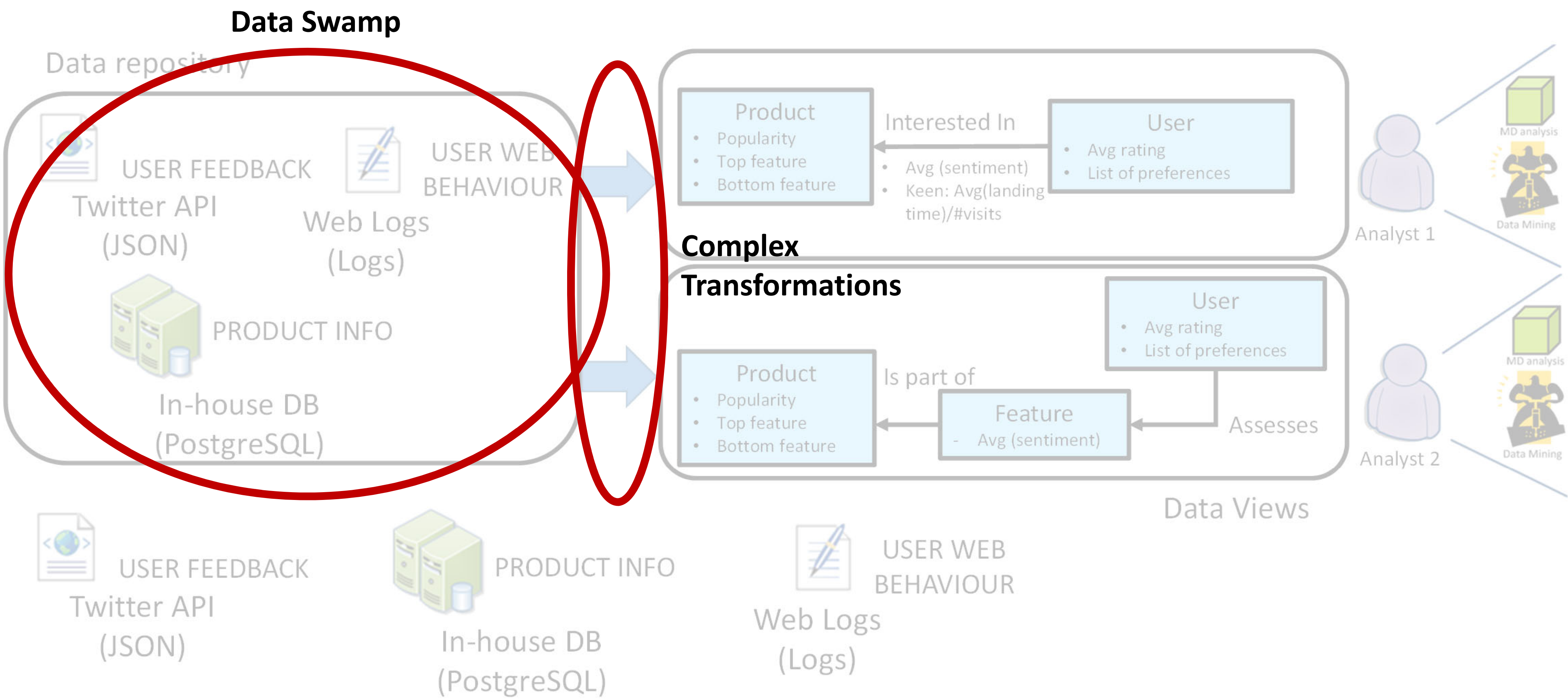
Drawbacks of Model-First (Load-Later)



Load-First (Model-Later)

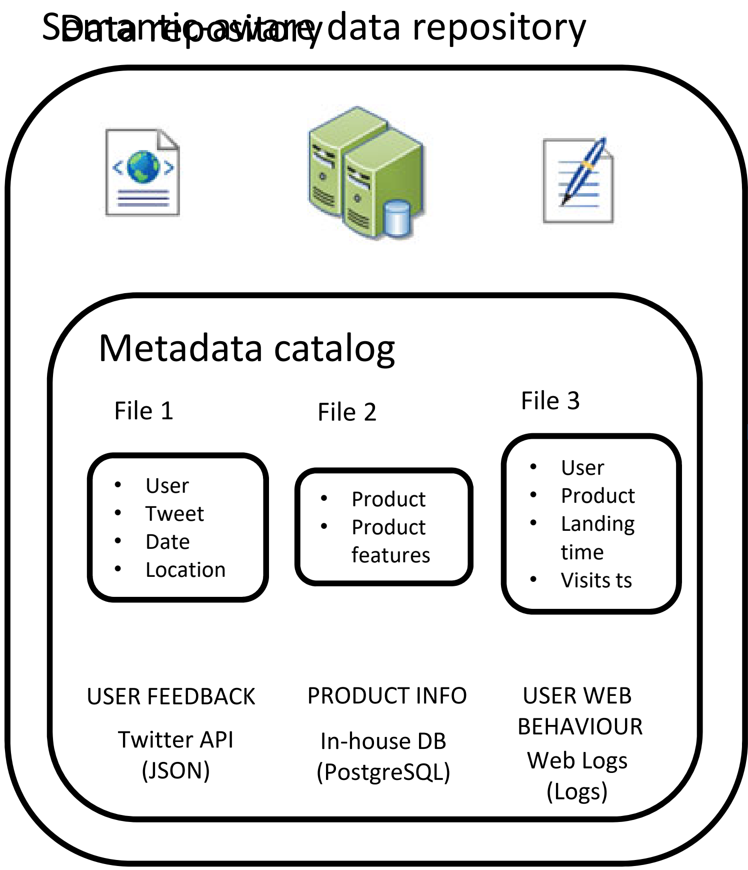


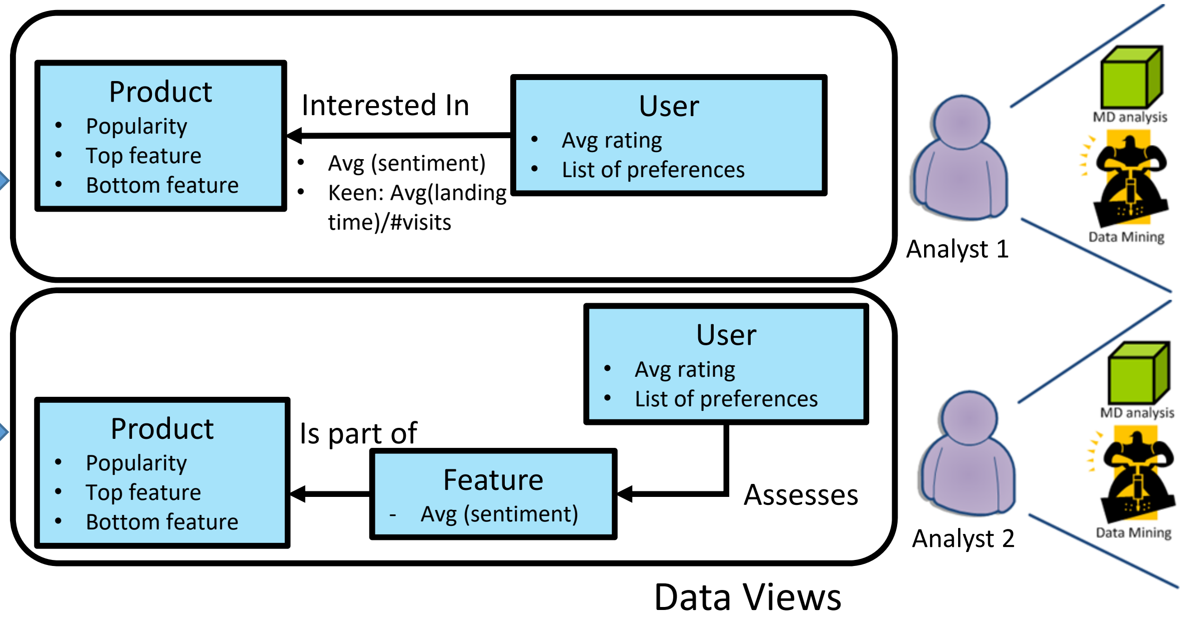
Drawbacks of Load-First (Model-Later)



Stonebraker (2014)

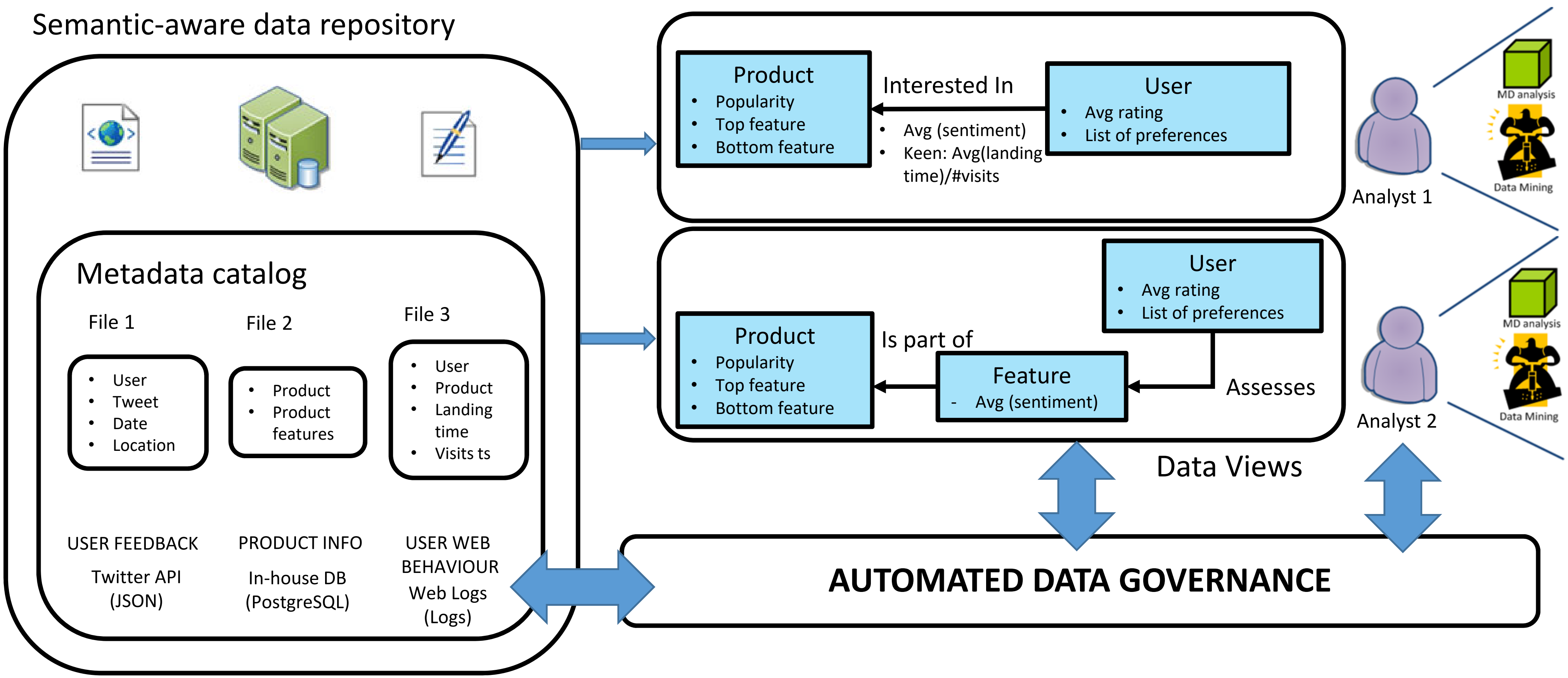
Towards semantic-awareness



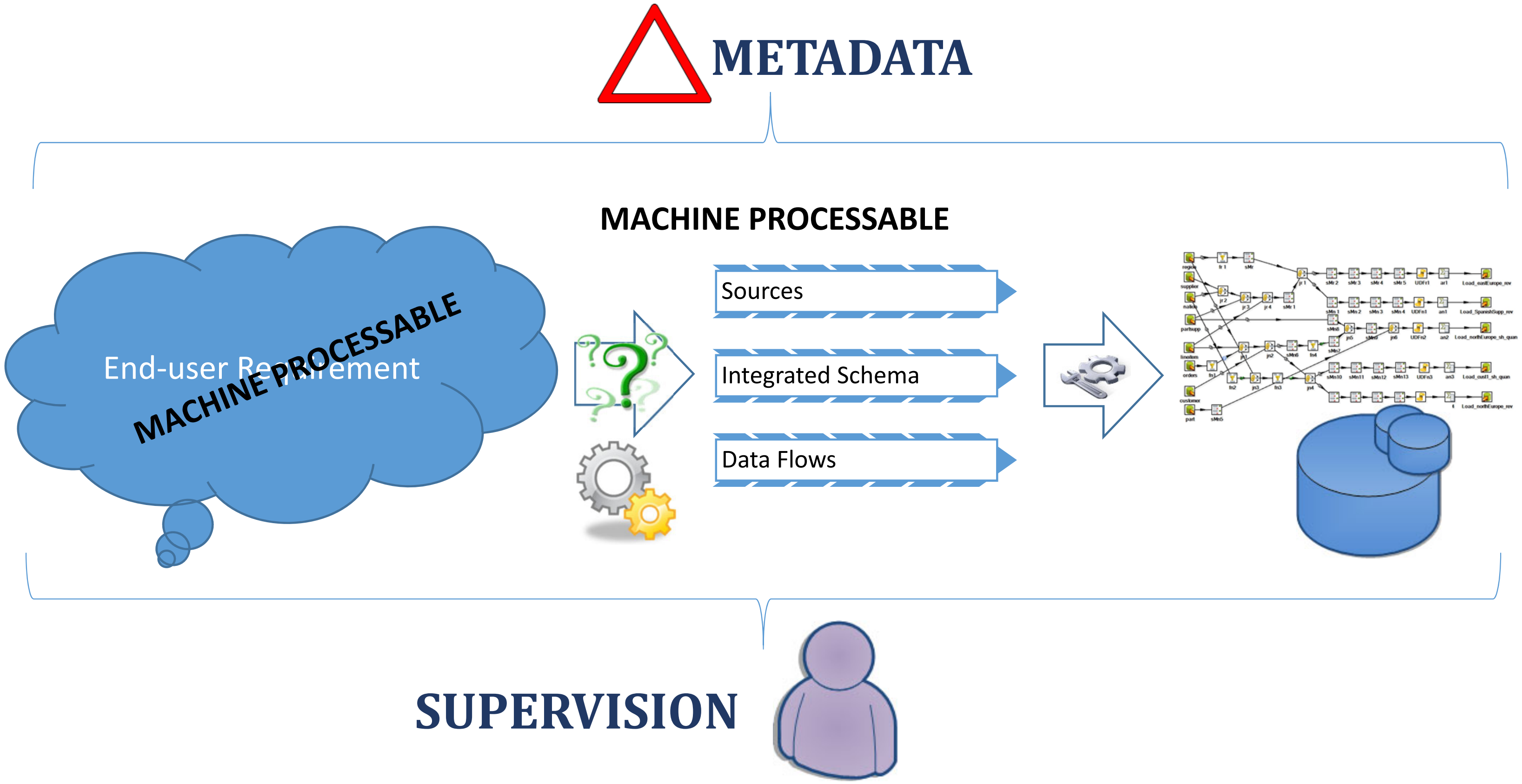


- Metadata provides semantics
  - Source schemata, mappings to views, parsing information, ...
- Automation of the integration processes

From IT-Centered to User-Centered

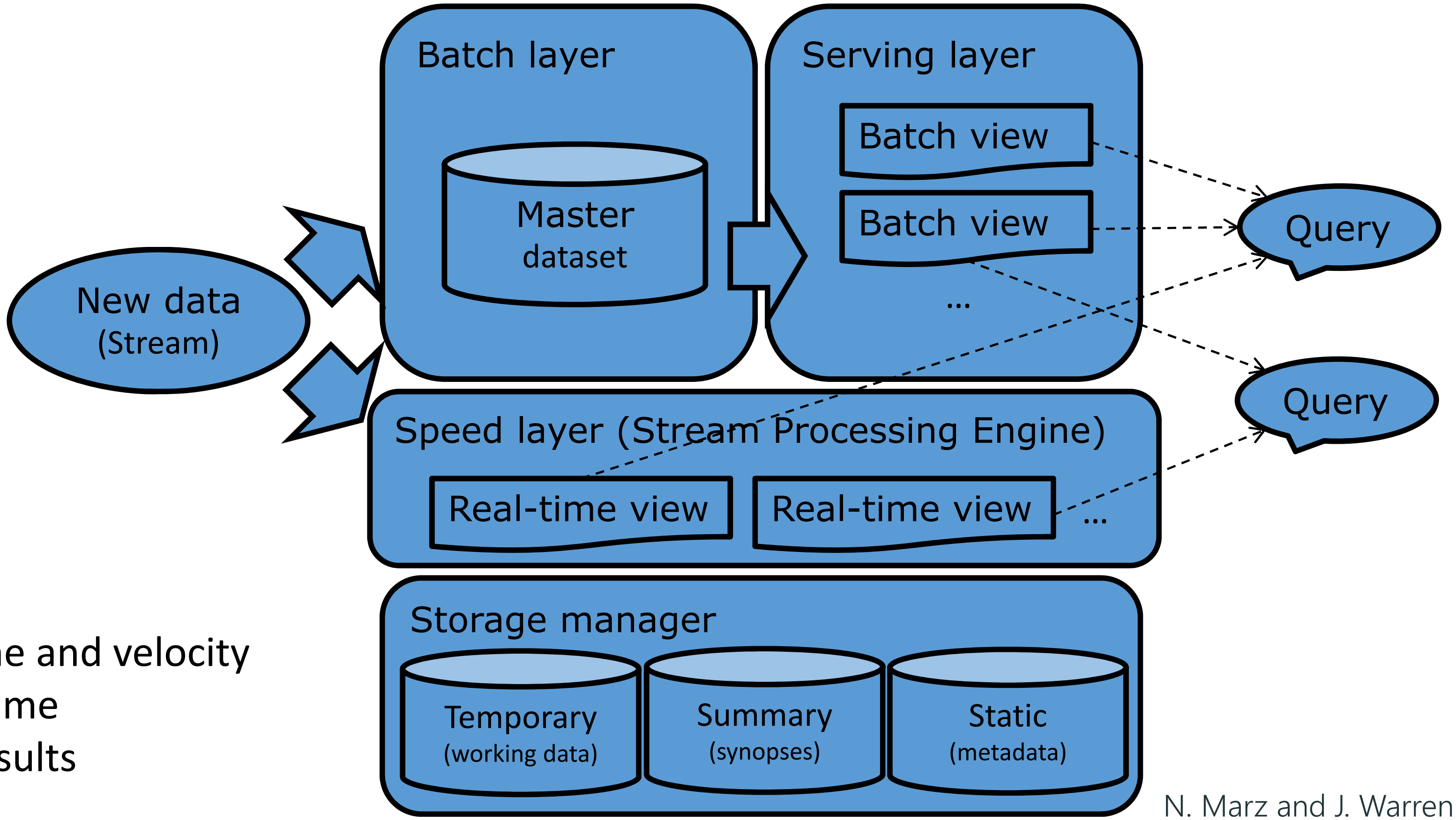


The Missing Link: Metadata



New Processing Architectural Patterns

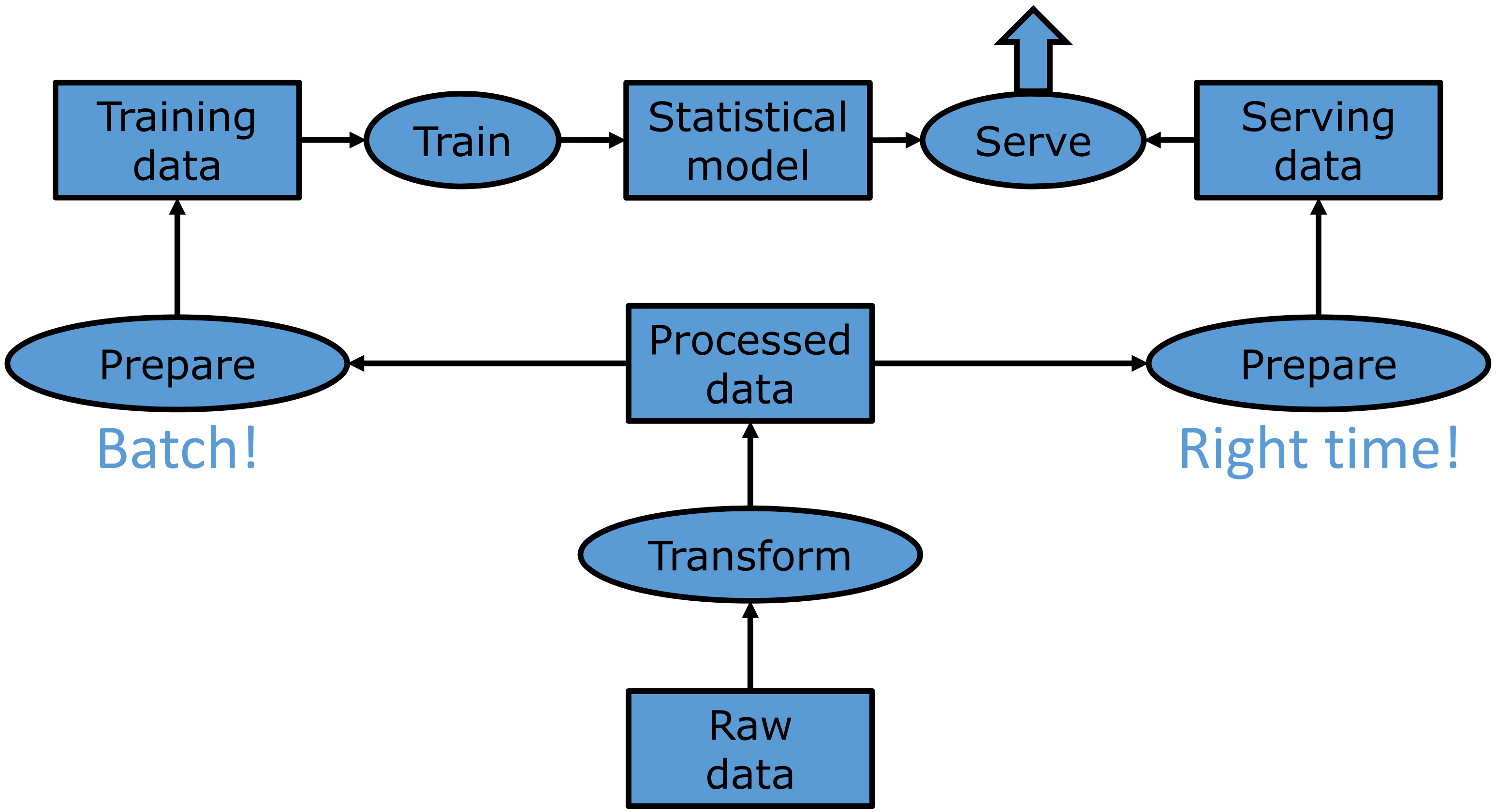
λ-Architecture



Idea: Accommodate volume and velocity
Batch processing vs. Real time
Precise vs. Approximate results

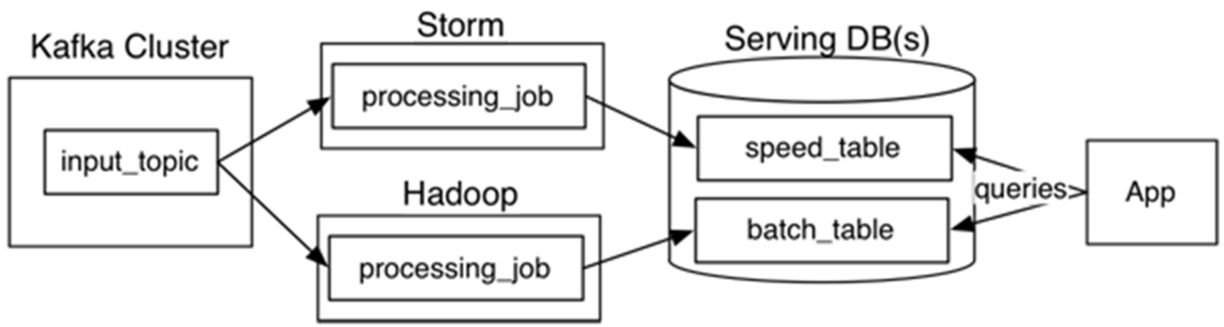
N. Marz and J. Warren

Data-centered architecture



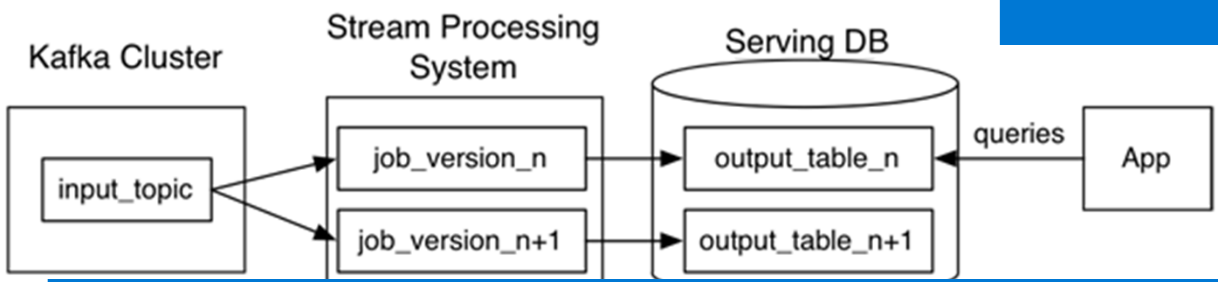
κ-architecture

λ



- Data is considered to be a never-ending stream

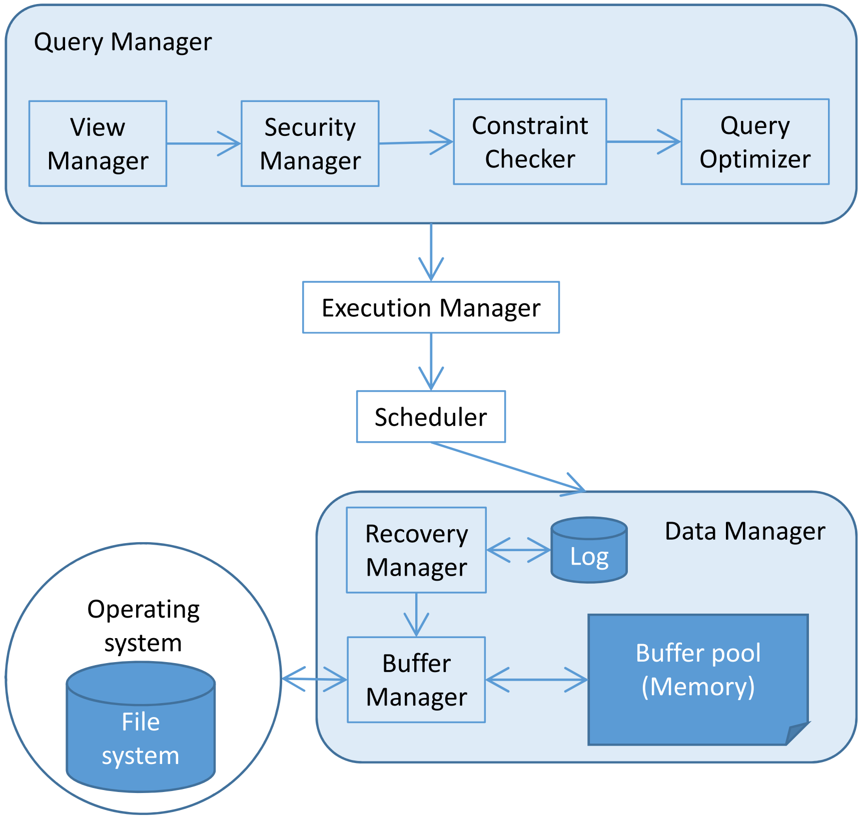
κ



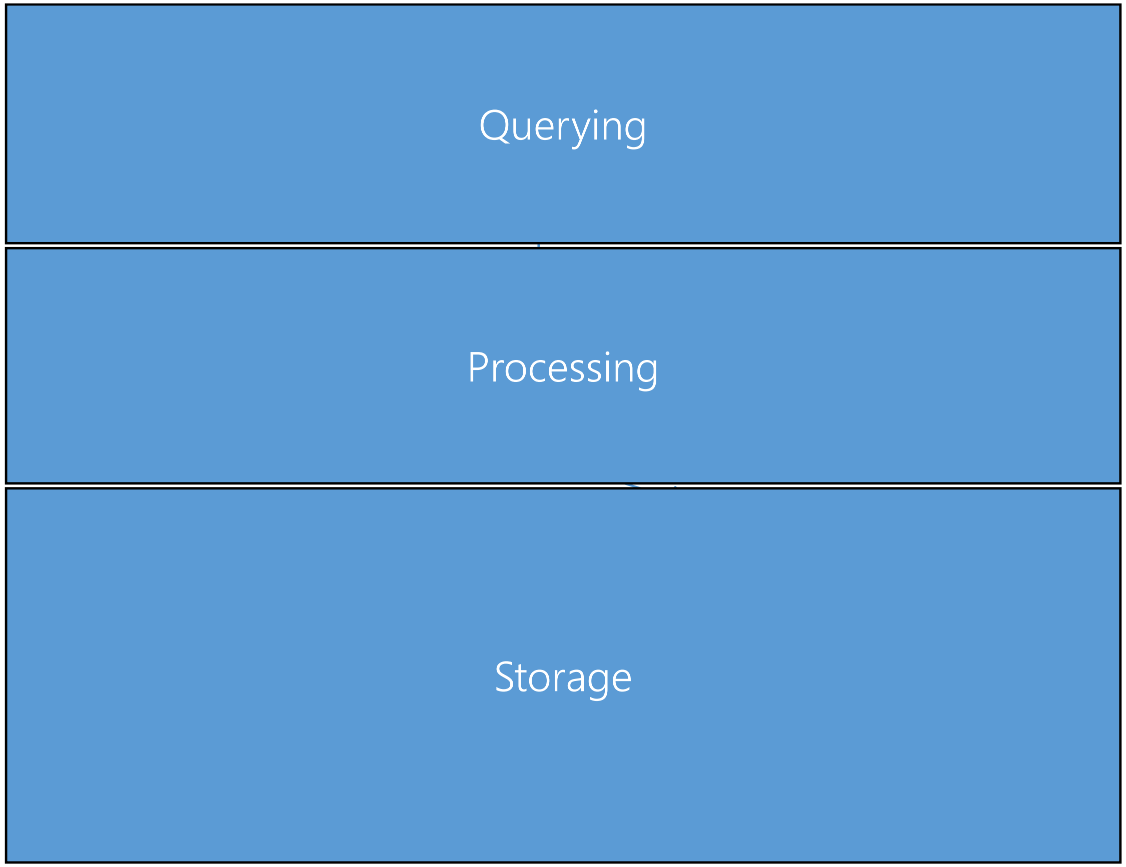
In a k-architecture, data is stored as an append [redacted]

Database Management System view

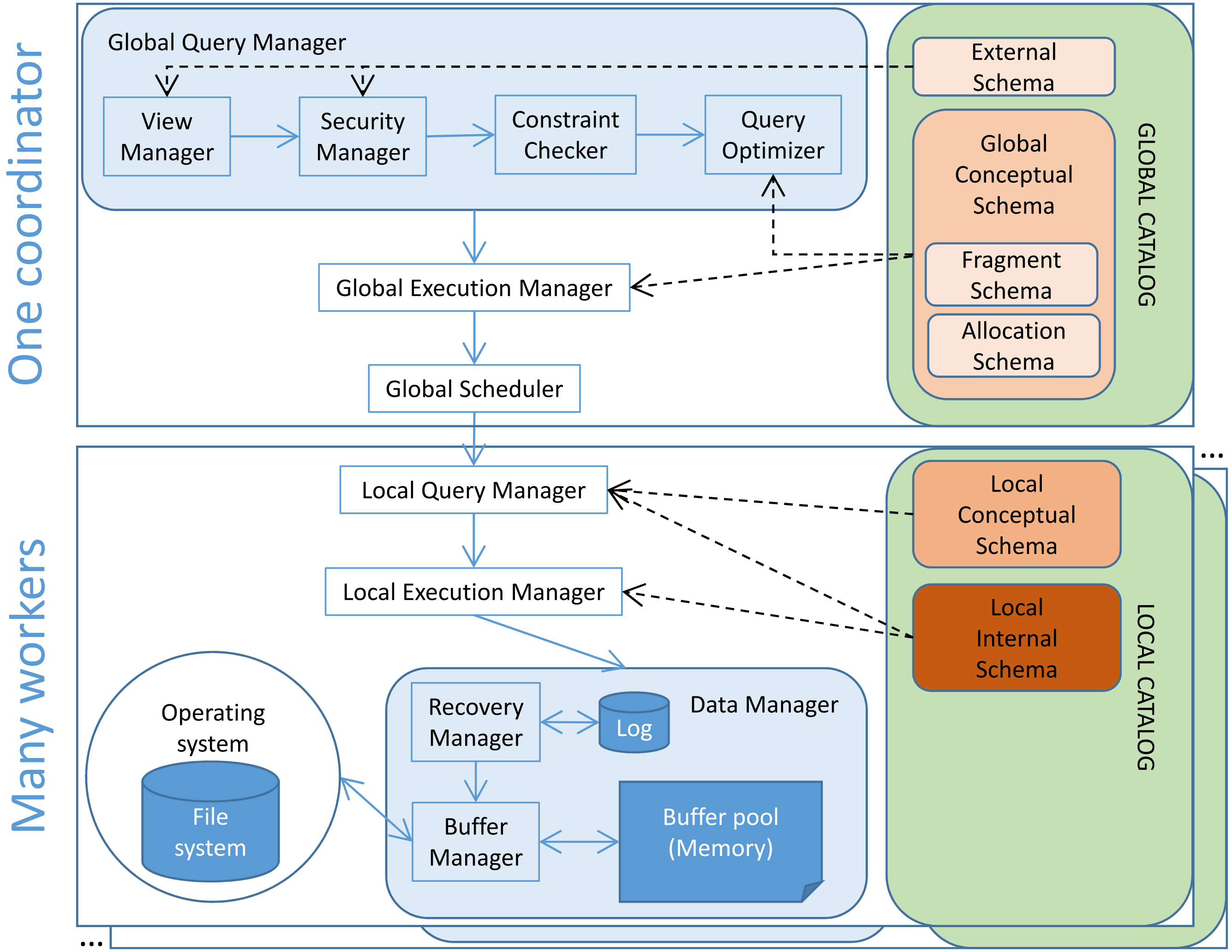
Centralized DBMS Architecture



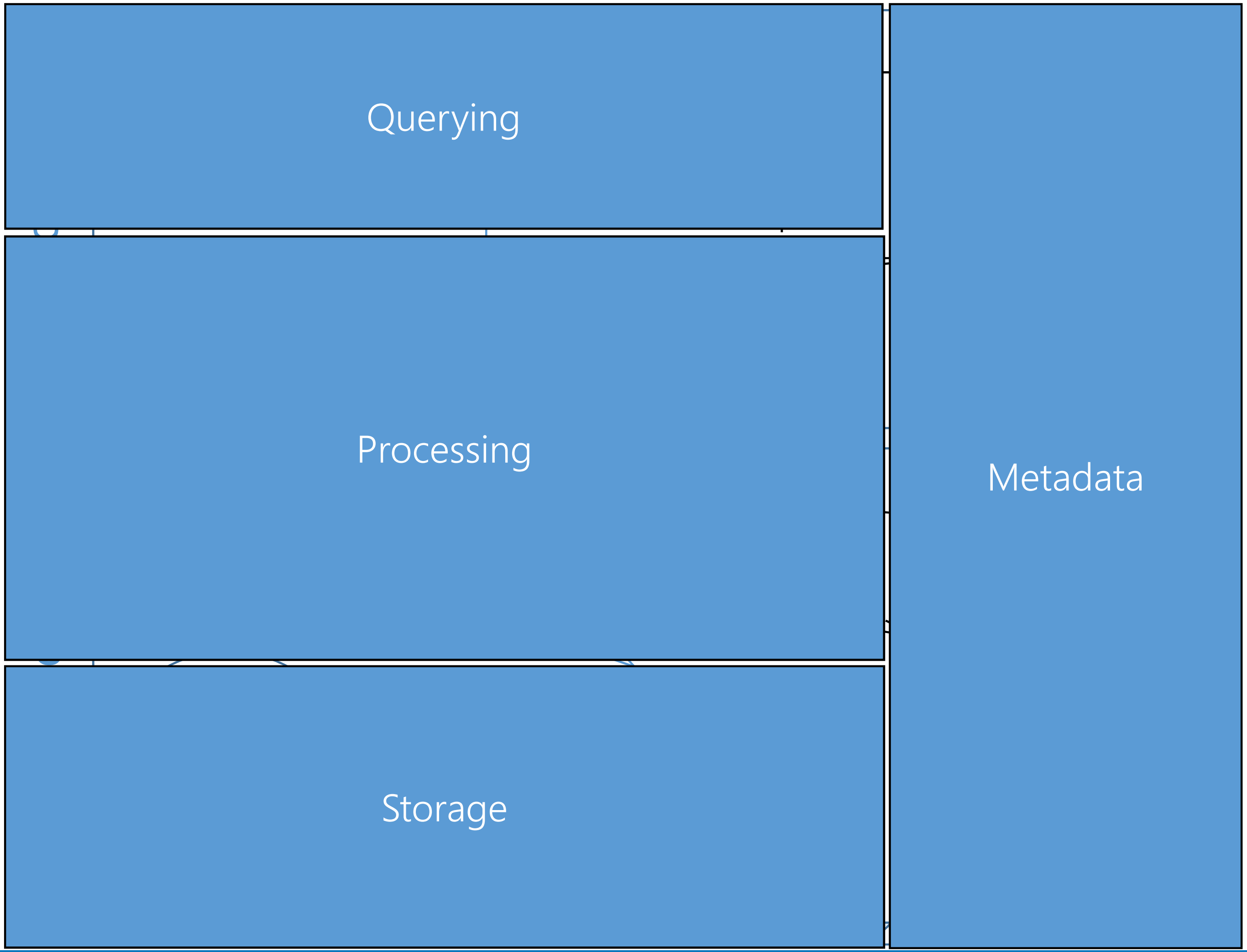
Centralized DBMS Architecture



Distributed DBMS Architecture

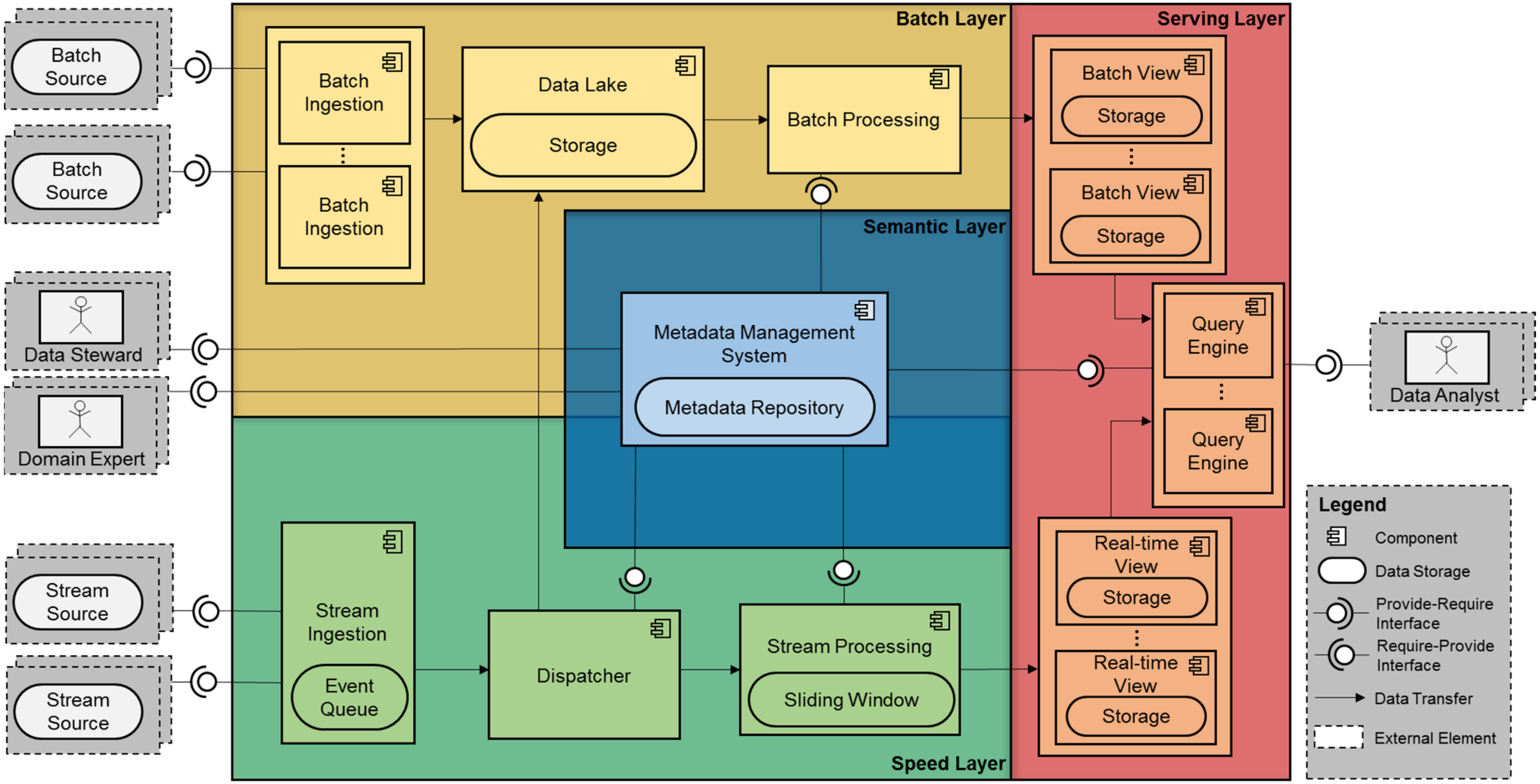


Distributed DBMS Architecture



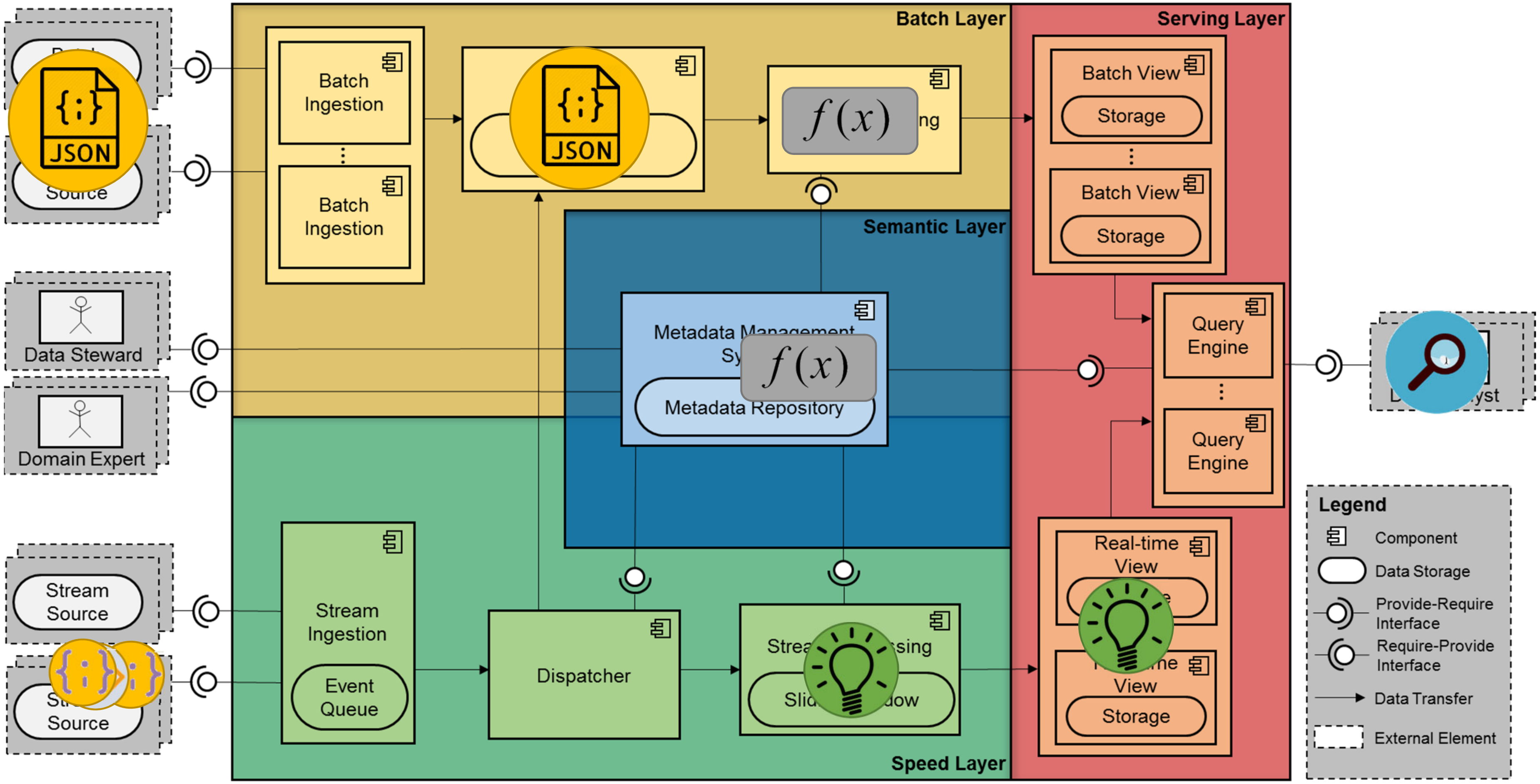
Bolster

In batch processing, Data Lake is for storage



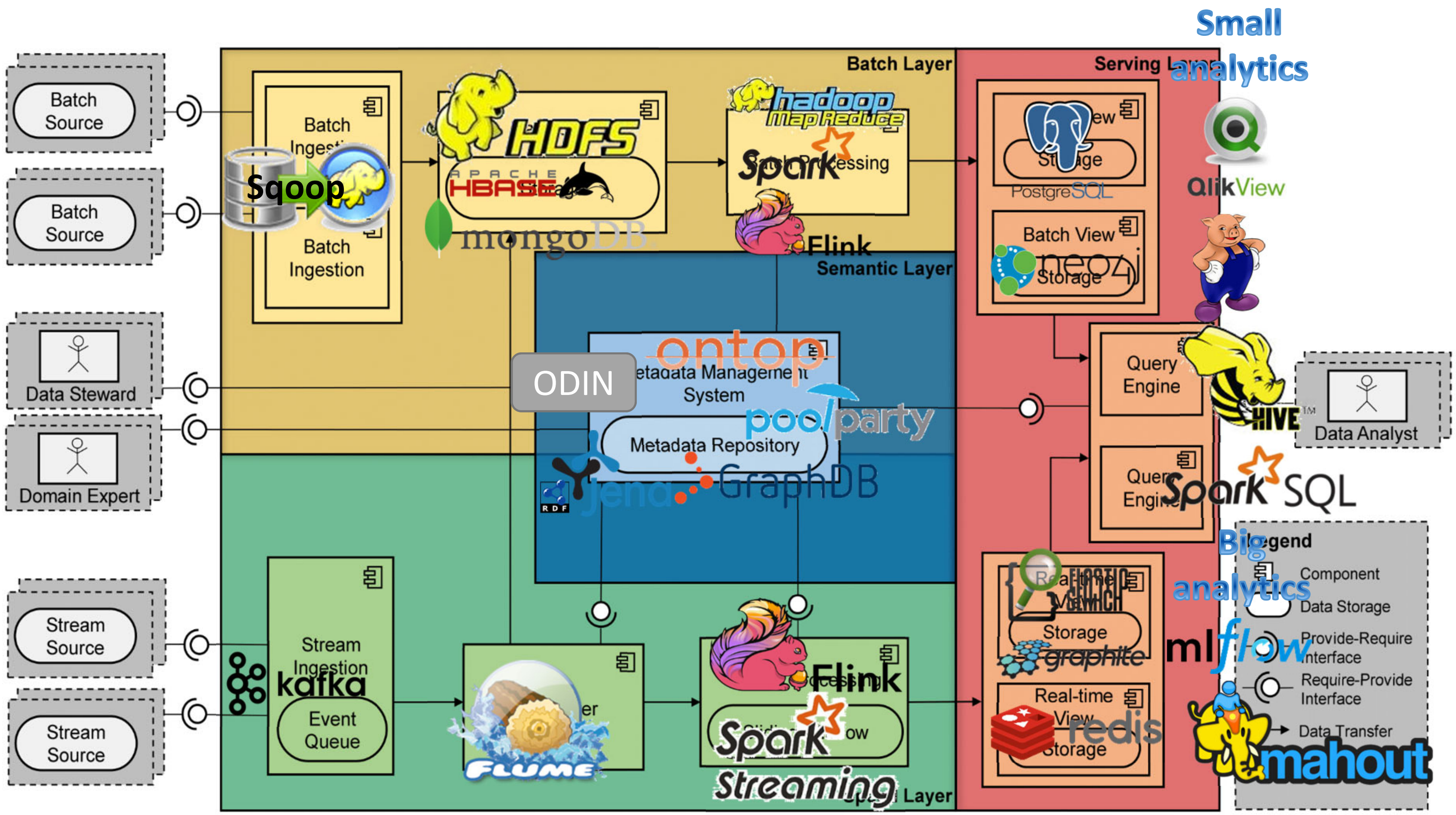
S. Nadal et al.

Bolster



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

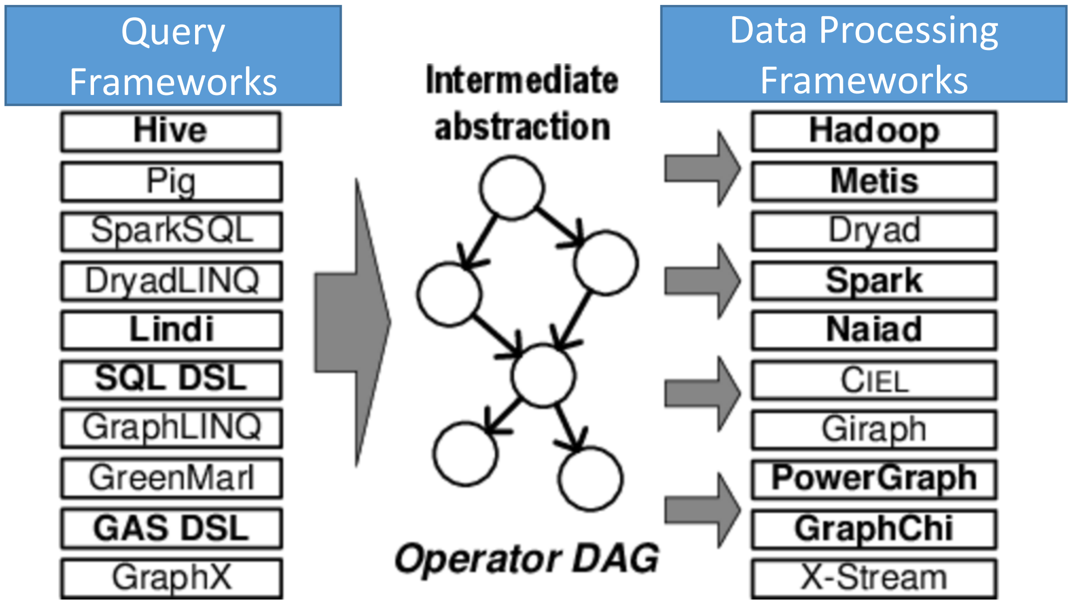
Bolster Instantiation



Workflow Orchestrators

Oozie is simple
Musketeer is more complex

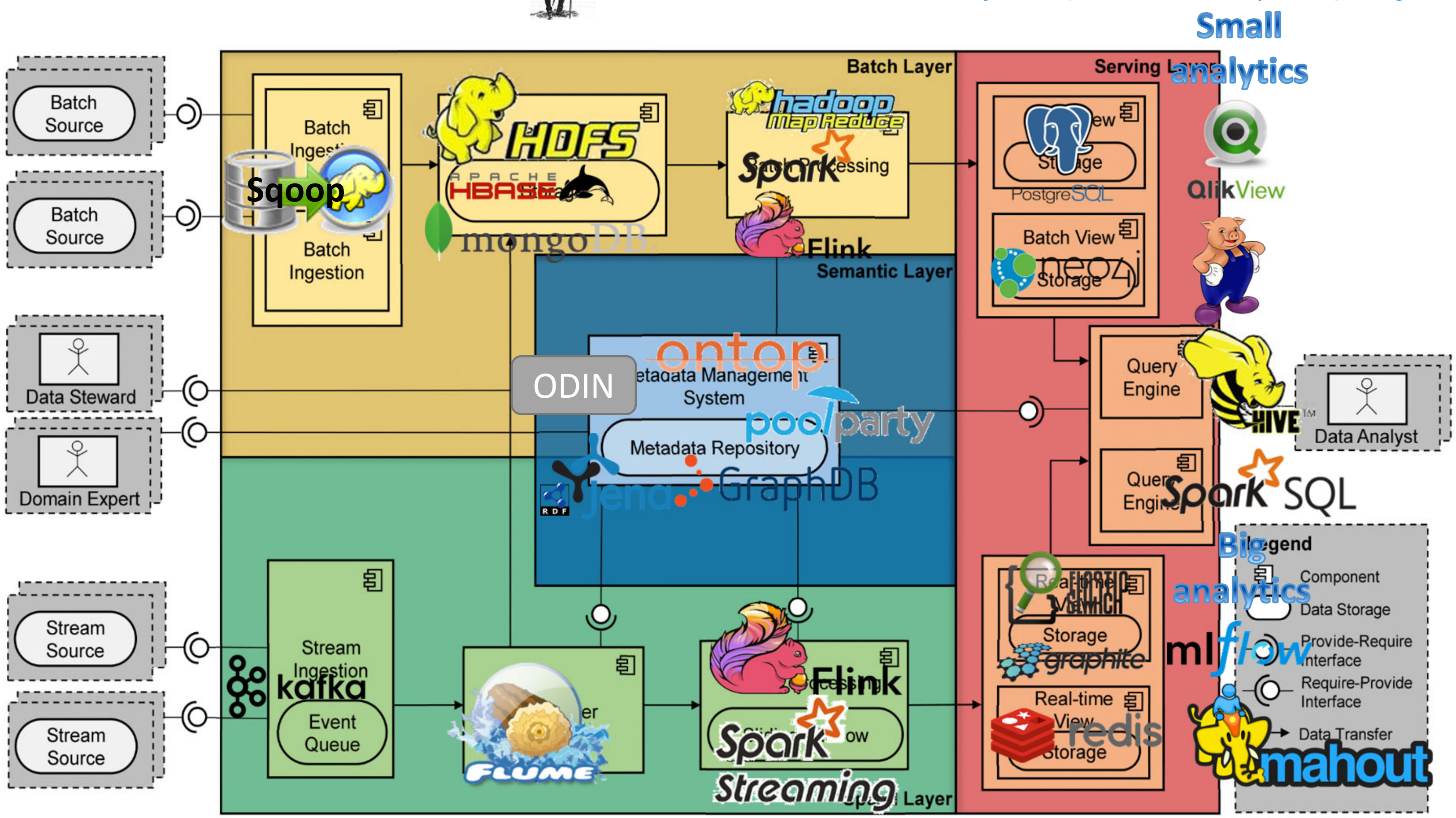
- Current workflow orchestrators are rather poor: Oozie
- But there are attempts for smarter approaches: the ideas behind Musketeer deserve special attention



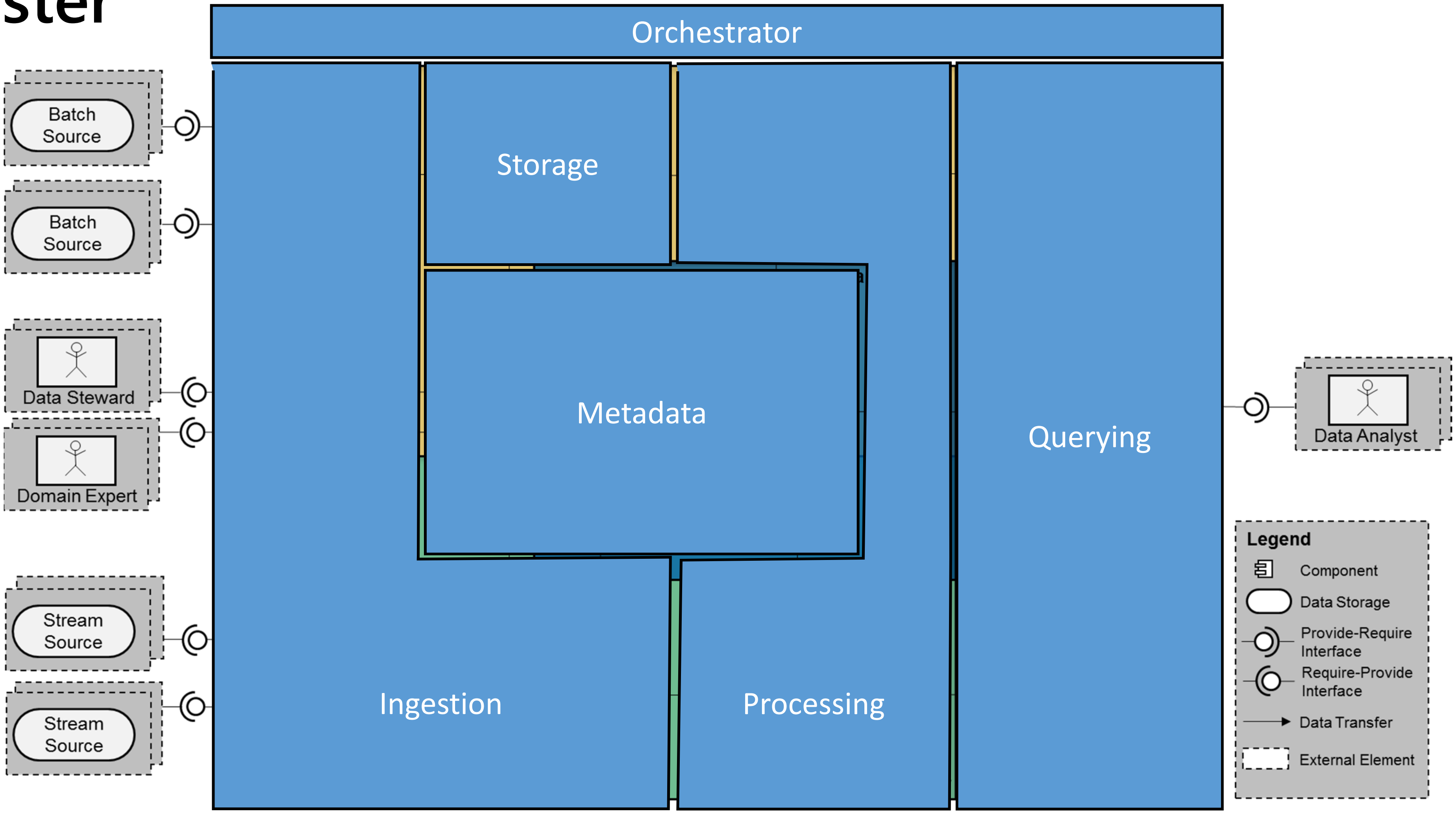
Does a similar job to global execution manager of traditional distributed RDBMS

Bolster Instantiation

Ingestion - Sqoop, Kafka, Flume
Storage - HDFS+HBASE, mongodb
Processing - Spark, Flink, Spark Streaming, hadoop
metadata - hcatalog, ontop, graphdb, poolparty, jeena
Querying - Postgres, SparkSQLL, Neo4j, redis, graphite
Analytics - QlikView, HIVE, SparkSQL, Big analytics, ml flow, mahout



Bolster



S. Nadal et al.

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
- 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

Resources

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