THE UNIVERSITY OF QUEENSLAND
AUSTRALIA
CREATE CHANGE

Advanced Database Systems (INFS3200)

Lecture 11: Data Lakes for Big Data Management

Lecturer: Dr Zhi Chen

Topics

Relational Database vs. Data Warehouse

- What motivates data lake?

Data Lake: Conceptual Design

- Data Lake history
- Data Lake architecture overview
- Data Lake components

From Lake To Lakehouse – Modern Big Data Management Systems

Relational Database vs. Data Warehouse

Relational Database

- OLTP (OnLine Transactional Processing)
  - Transactional data
  - Structured data
  - Day to day data
- Schema on write
- The cost to store in database is high

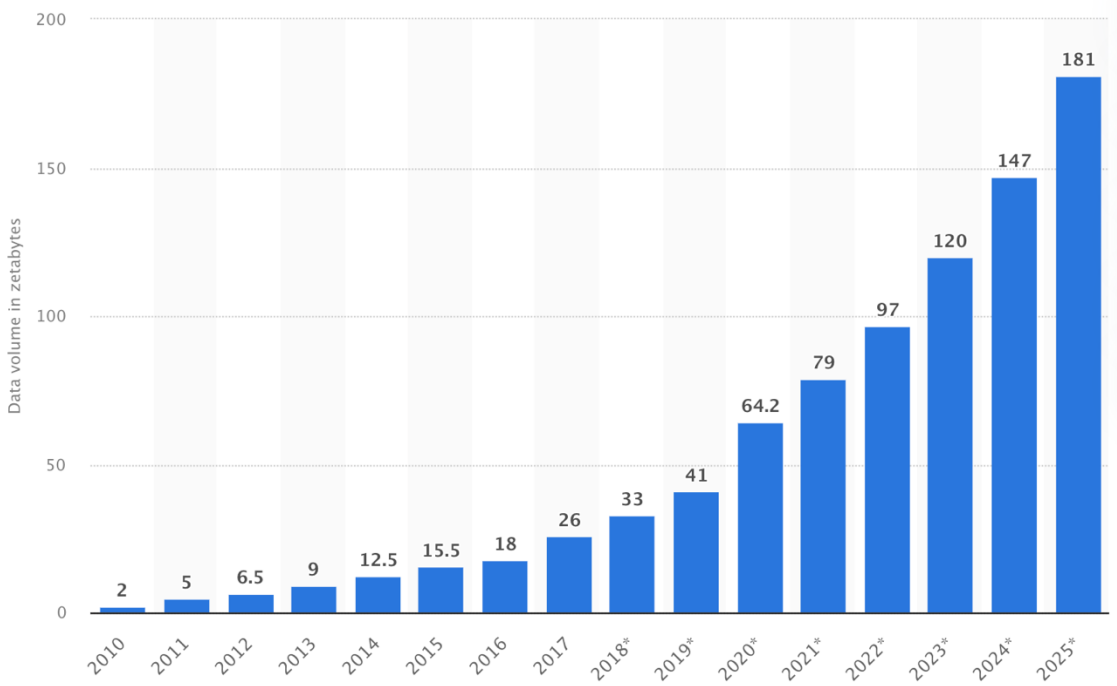
Data Warehouse

- OLAP (OnLine Analytical Processing)
  - Structured data
  - historical data from multiple sources
- ETL (Extract Transform Load)
  - Flexibility?
- Schema on write.
- Less cost than databases

Motivation

Big data era: exponential increase of data production

- Transactional / social media / Internet of Things (IoT) …
- High volume, high velocity, high variety, veracity issues
- semi-structured and unstructured data
  - The majority of big data is unstructured



https://www.statista.com/statistics/871513/worldwide-data-created/

Motivation

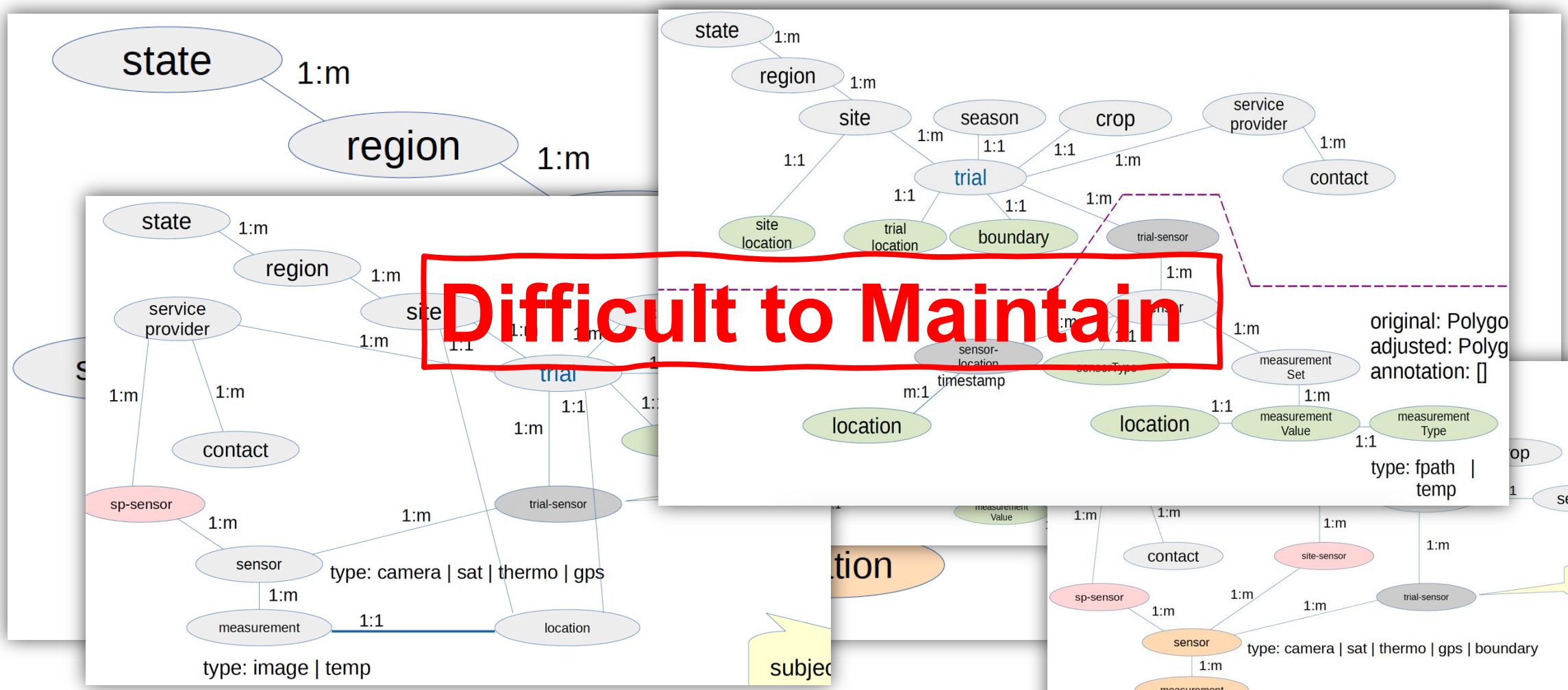
Big data era: exponential increase of data production

- Transactional / social media / Internet of Things (IoT) …
- High volume, high velocity, high variety, veracity issues
- semi-structured and unstructured data
  - The majority of big data is unstructured

These factors induce great challenges for data warehouses

- ETL, Schema-on-write, etc.

Exponential increase of data production



Topics

Relational Database vs. Data Warehouse

- What motivates data lake?

Data Lake: Conceptual Design

- Data Lake history
- Data Lake architecture overview
- Data Lake components

From Lake To Lakehouse – Modern Big Data Management Systems

Data Lake

A flexible, scalable data storage and management system.

Ingests and stores raw data from heterogeneous sources in their original format

- Schema-on-write vs. schema-on-read
- Provides maintenance, query processing,
- Powering data analytics with the help of rich metadata

Stores and manages data in many real-life use cases

- Internet of things (IoT) and smart city
- Flights data
- Manufacturing
- …

Topics

Relational Database vs. Data Warehouse

- What motivates data lake?

Data Lake: Conceptual Design

- Data Lake history
- Data Lake architecture overview
- Data Lake components

From Lake To Lakehouse – Modern Big Data Management Systems

Data Lake History (2010 - 2013)

Handles raw data from **one** source and supports diverse user requirements

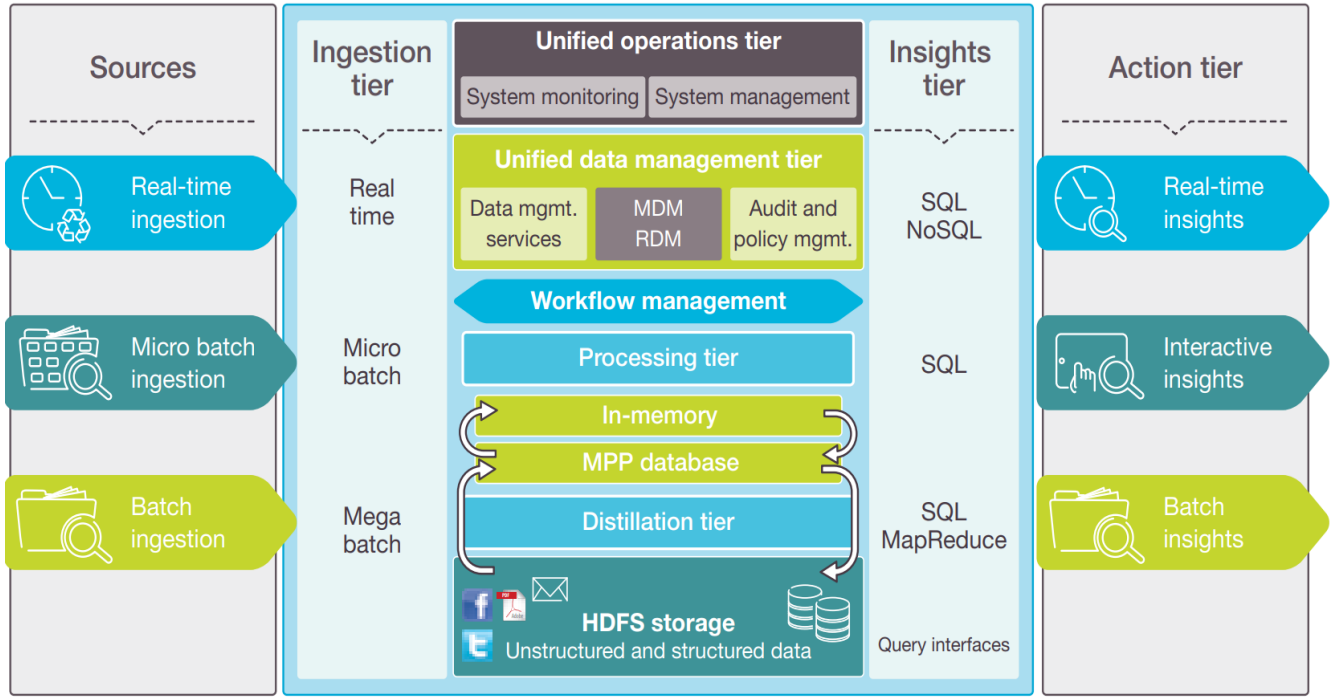
- Avoid or delay expensive standard ‘transform’
- Lake: A large body of water in a more natural state.

Data Lake History (2010 - 2013)

Raw data from multiple sources

Three abstract tiers:

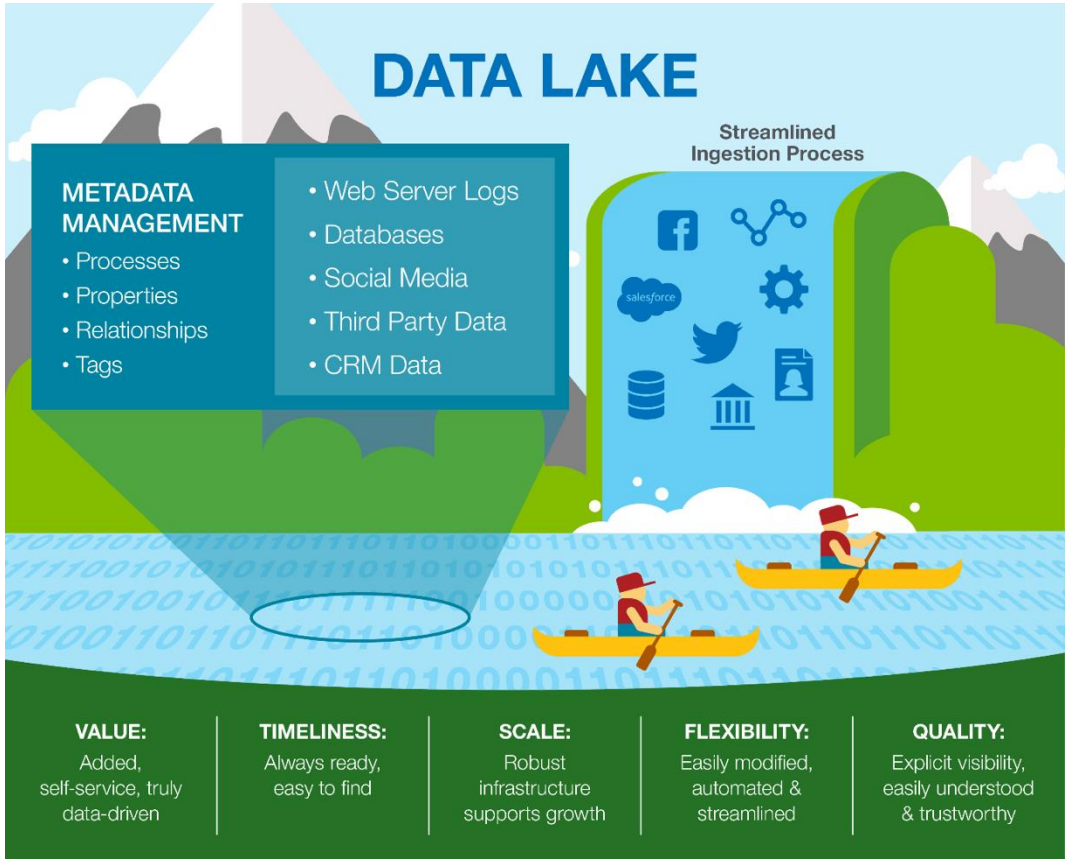
- Ingestion tier: takes data in real time
- Insight tier: analyses data
- Action tier: link insights with the existing applications

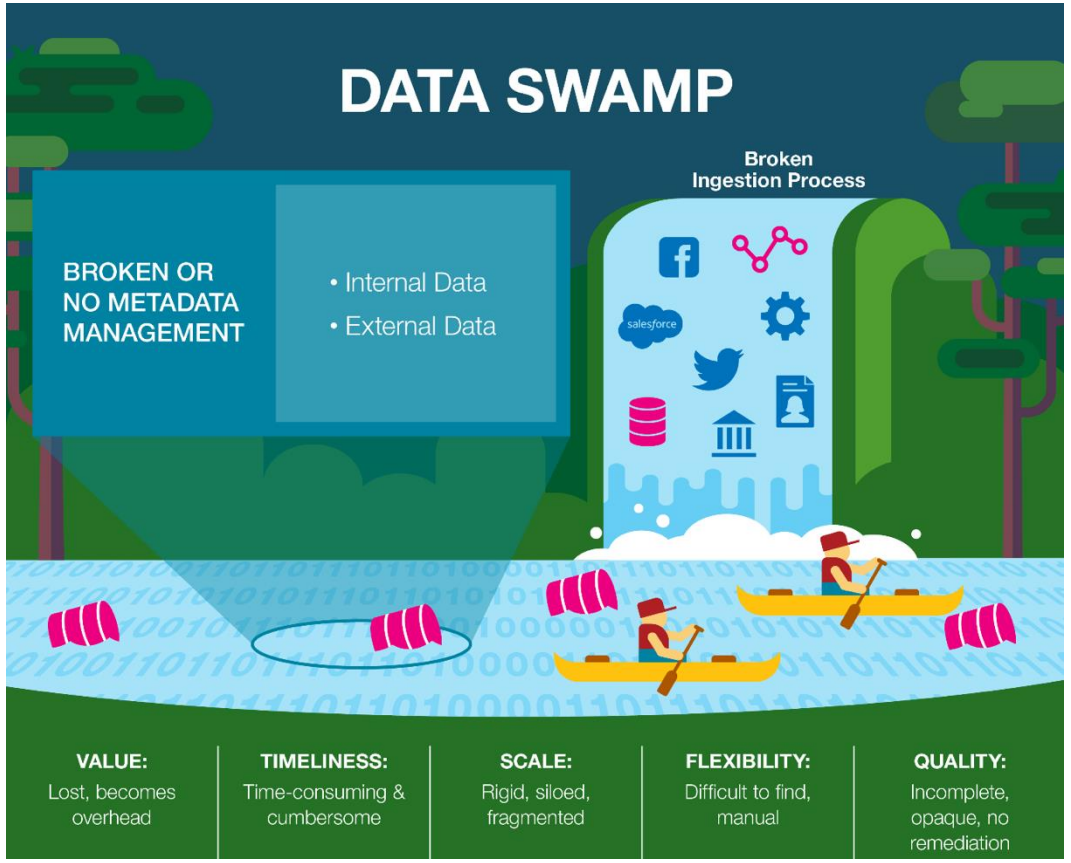


https://www.capgemini.com/wp-content/uploads/2017/07/pivotal-business-data-lake-technical_overview_web.pdf

Data Lake History (2014 - 2015)

Criticisms: “Data lake” or “Data swamp”





Source: https://www.collibra.com/wp-content/uploads/DataLake-V2-HighRez-1.png

Data Lake History (2016 - present)

Realization of data lakes in industry

Data Lake Architecture

There are mainly two high-level data lake philosophies

Pond: partitions ingested data by their status and usage

- Raw data pond
- Analog data pond
- Application data pond
- Textual data pond
- Archival data pond

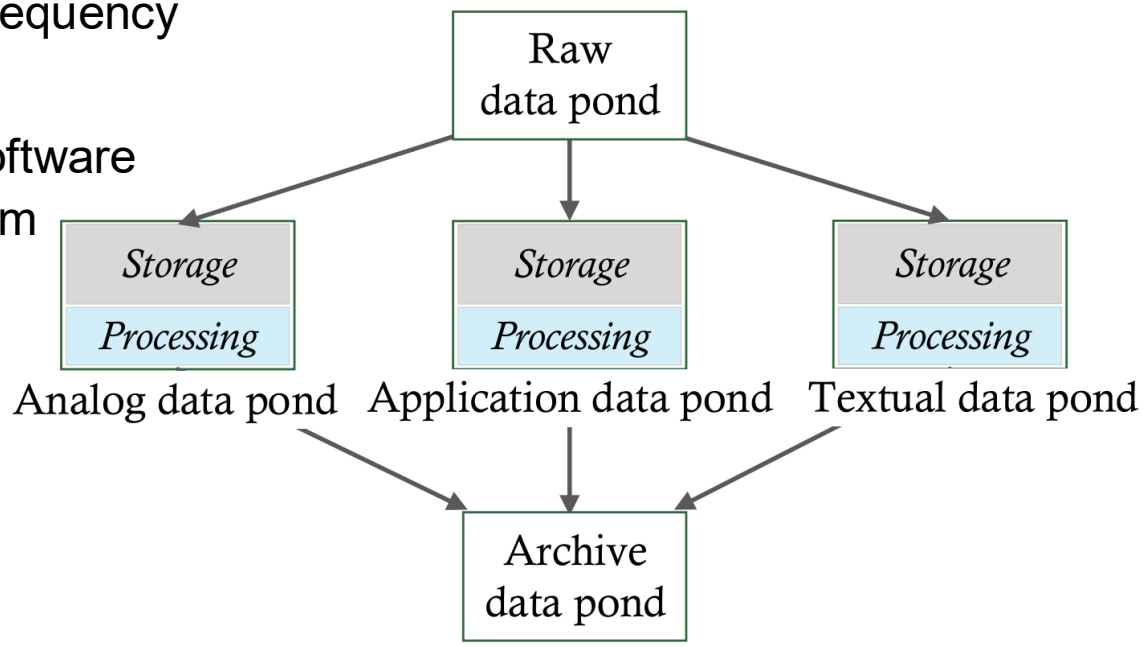
Zone: separates the life cycle of each dataset into different stages

Pond Architecture

Raw data pond deals with newly ingested, raw data.

Analog data pond are characterized by a very high frequency of measurements

Application data pond stores data that come from software applications and are thus generally structured data from relational DBMSs.



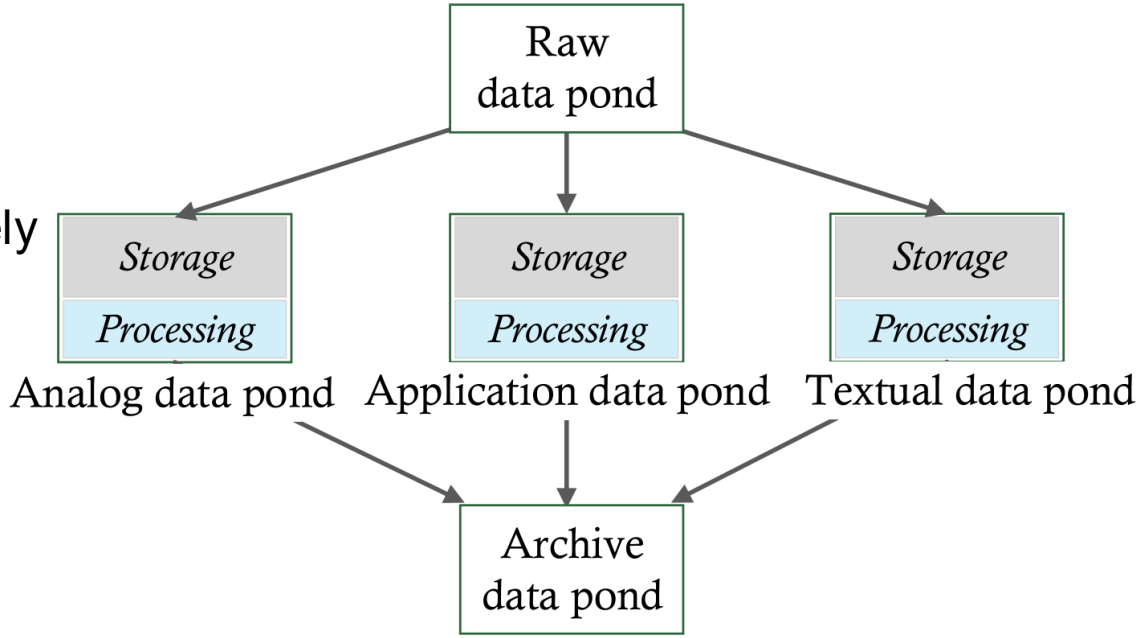
Raw data pond deals with newly ingested, raw data.
Analog data pond are characterized by a very high frequency of measurements
Application data pond stores data that come from software applications and are thus generally structured data from relational DBMSs.

Pond Architecture

Textual data pond manages unstructured, textual data.

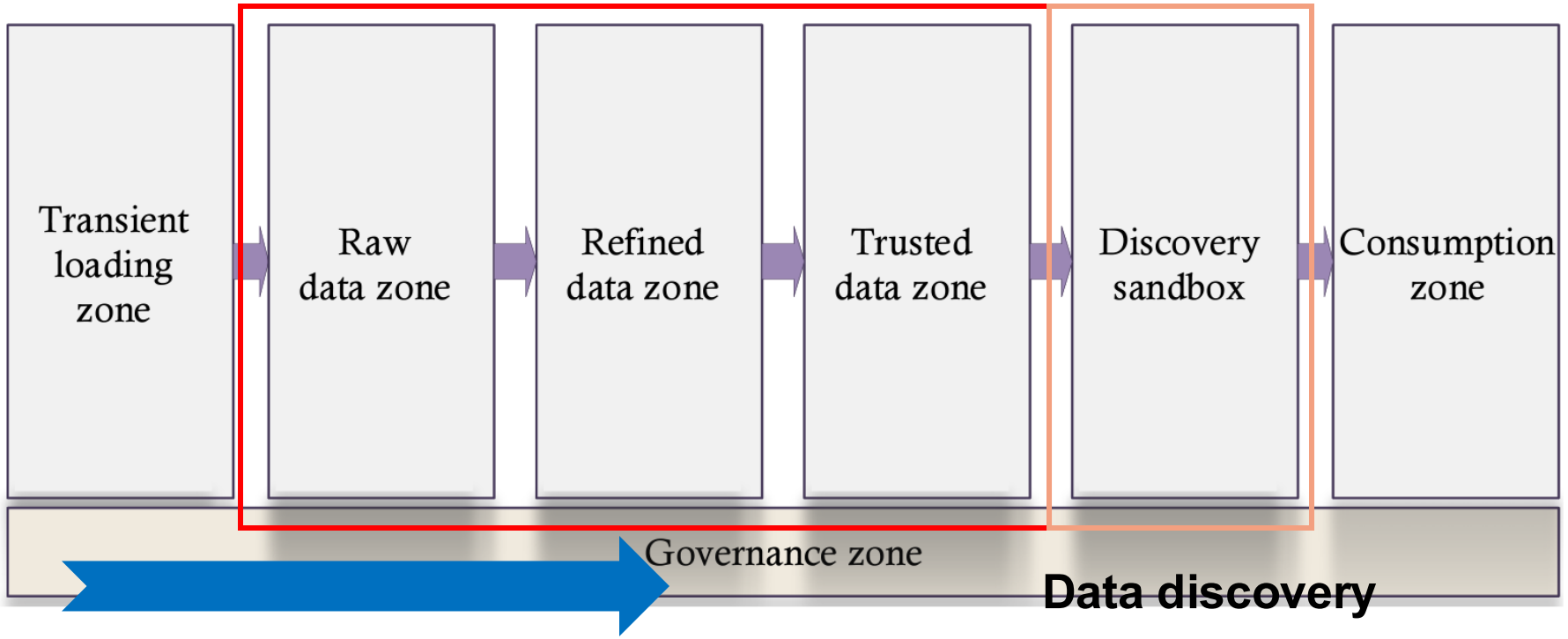
- It features a textual disambiguation process to ease textual data analysis.

Archival data pond saves the data that are not actively used but might still be needed in the future.



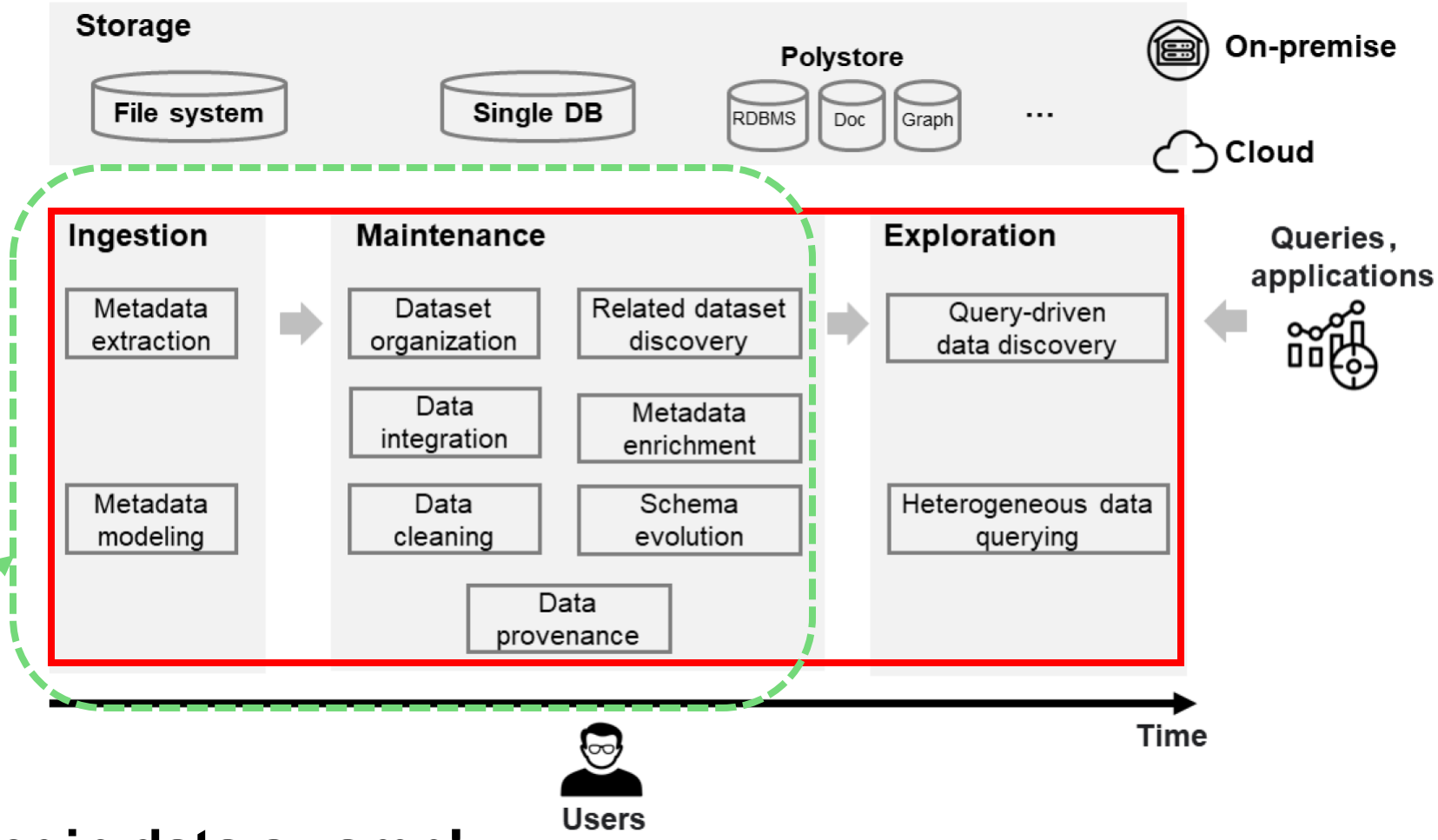
Zone Architecture

Zone architectures assign data to a zone according to their degree of refinement.



Data Lake Architecture

- Storage
- Ingestion
- Maintenance
- Exploration



Keeps data lake from begin data swamp!

Data Storage in Data Lakes

File-based storage systems

Hadoop Distributed File System (HDFS)

- Support wide range of file formats: text, binary, etc.
- Data compressions
- Column storage and row-based storage

Azure data lake store

- A hierarchical, multi-tier file-based storage system.
- Optimized for large unstructured object data
- Also supports Hadoop eco-system.

Data Storage in Data Lakes

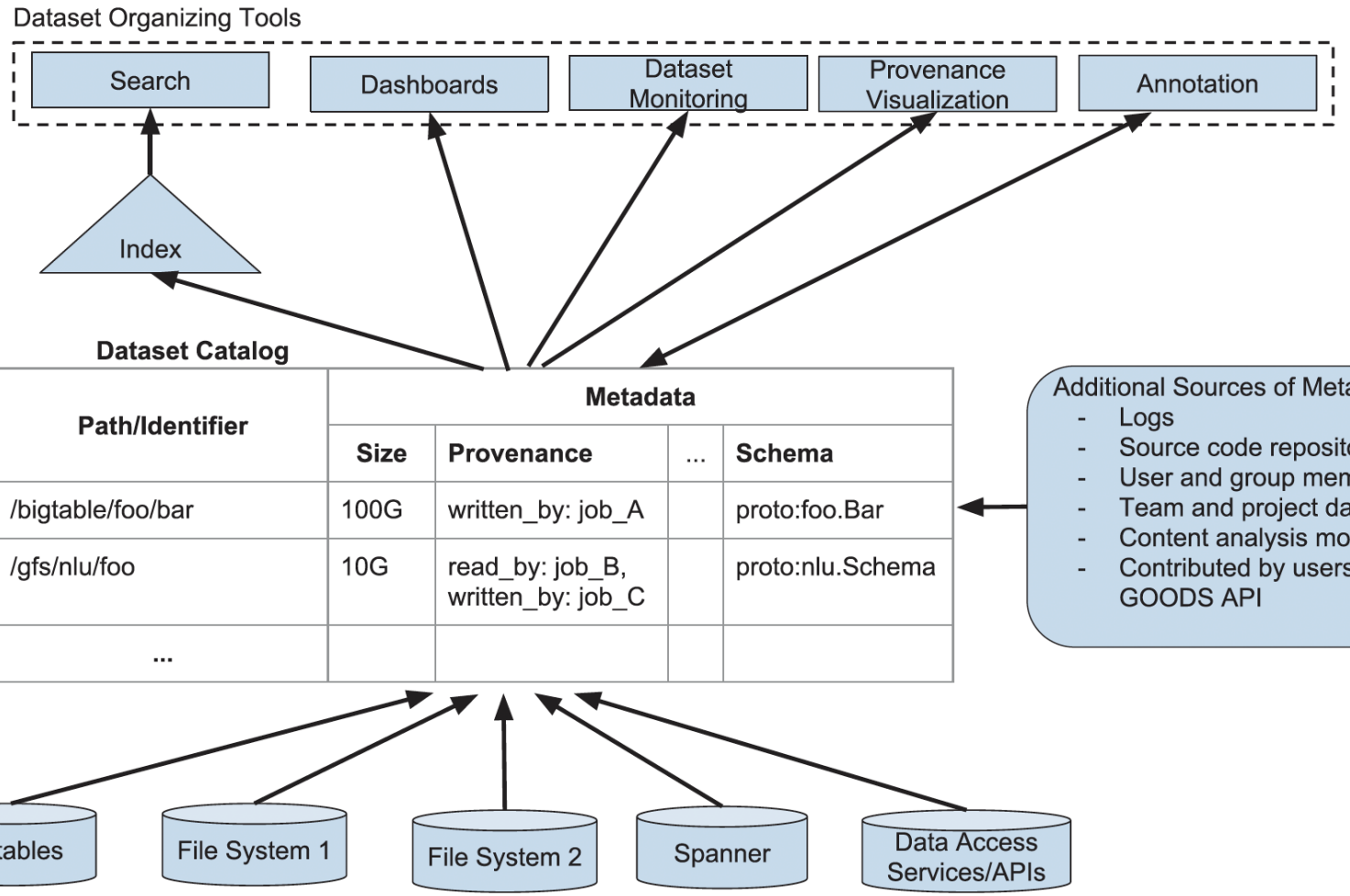
Single data store systems aims at specific types of data and employ a single database system at their storage tier

- E.g., Personal data lake: specialized for storing user data

Polystore systems: integrated access to a hybrid of multiple data stores for heterogeneous data

- E.g., Google Dataset Search (GOODS)
- A unified view of the data vs. no inherent structure or enforced schema.

Google Dataset Search (GOODS)



Unified view

Heterogeneous sources

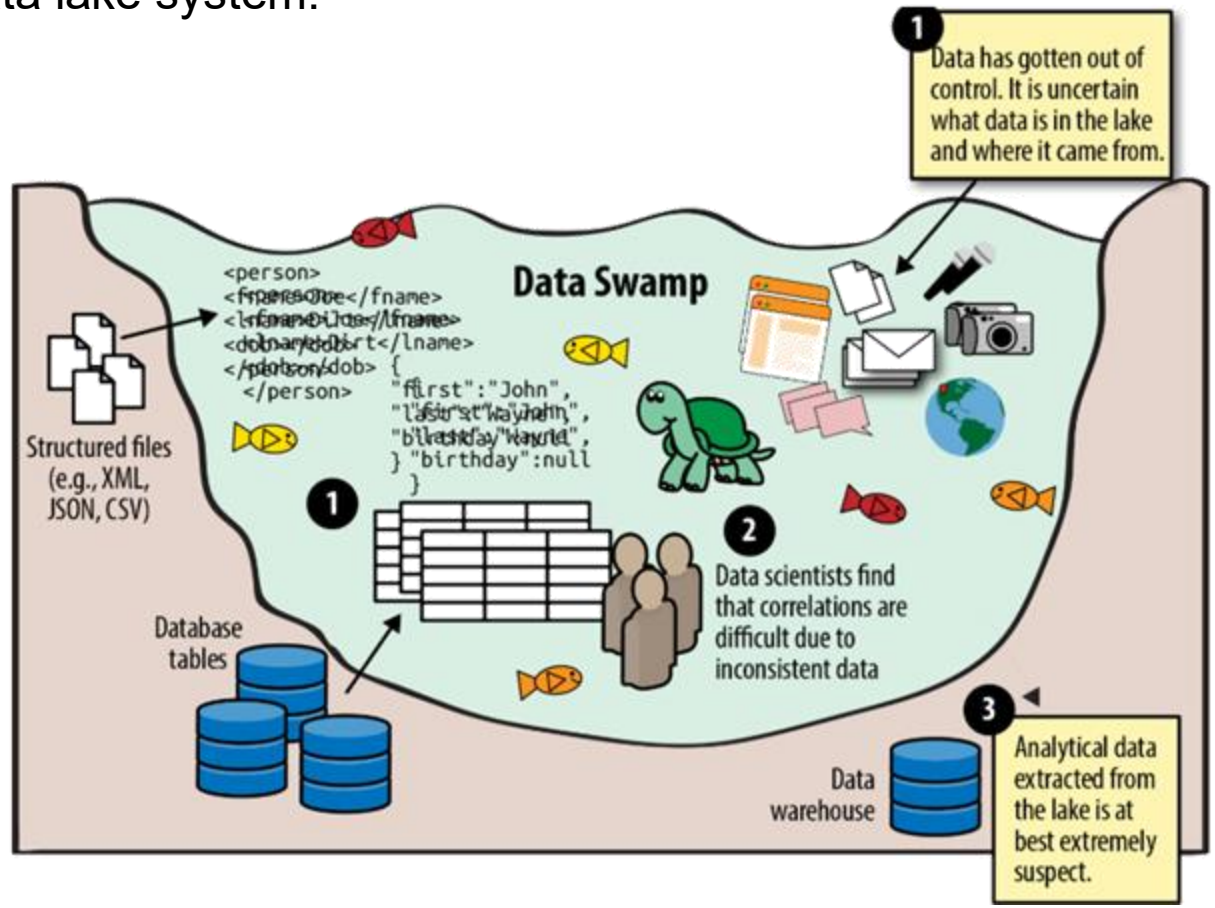
Three-Tier Architecture: Ingestion

Importing data from heterogeneous sources into the data lake system.

Two key challenges:

- Metadata extraction
- Metadata modelling

But what is metadata?

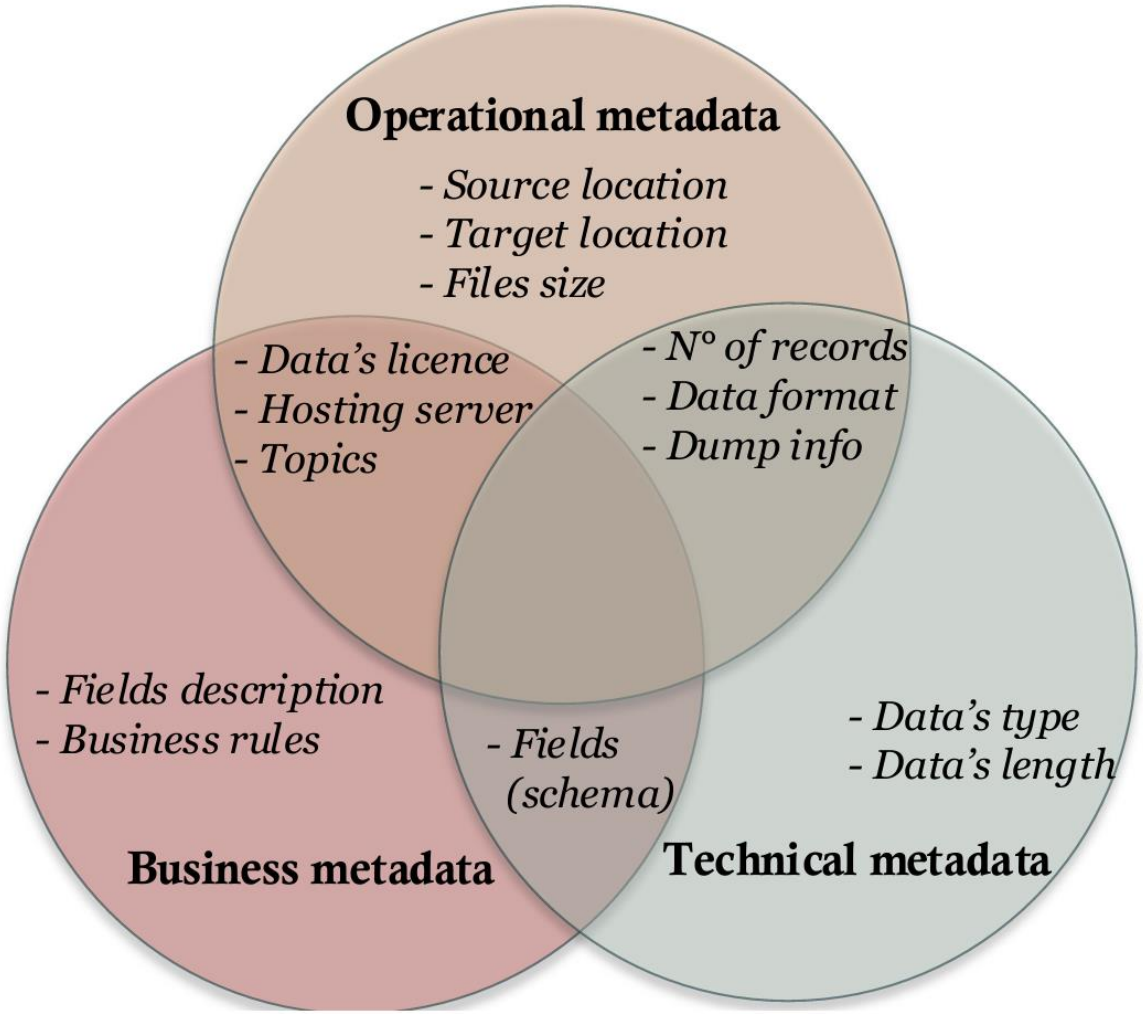


Metadata

Metadata Categories

Functional Metadata

- Business
- Operational
- Technical



Metadata

Metadata Categories

Functional Metadata

Structural Metadata

- Intra-object
- Inter-object
- Global

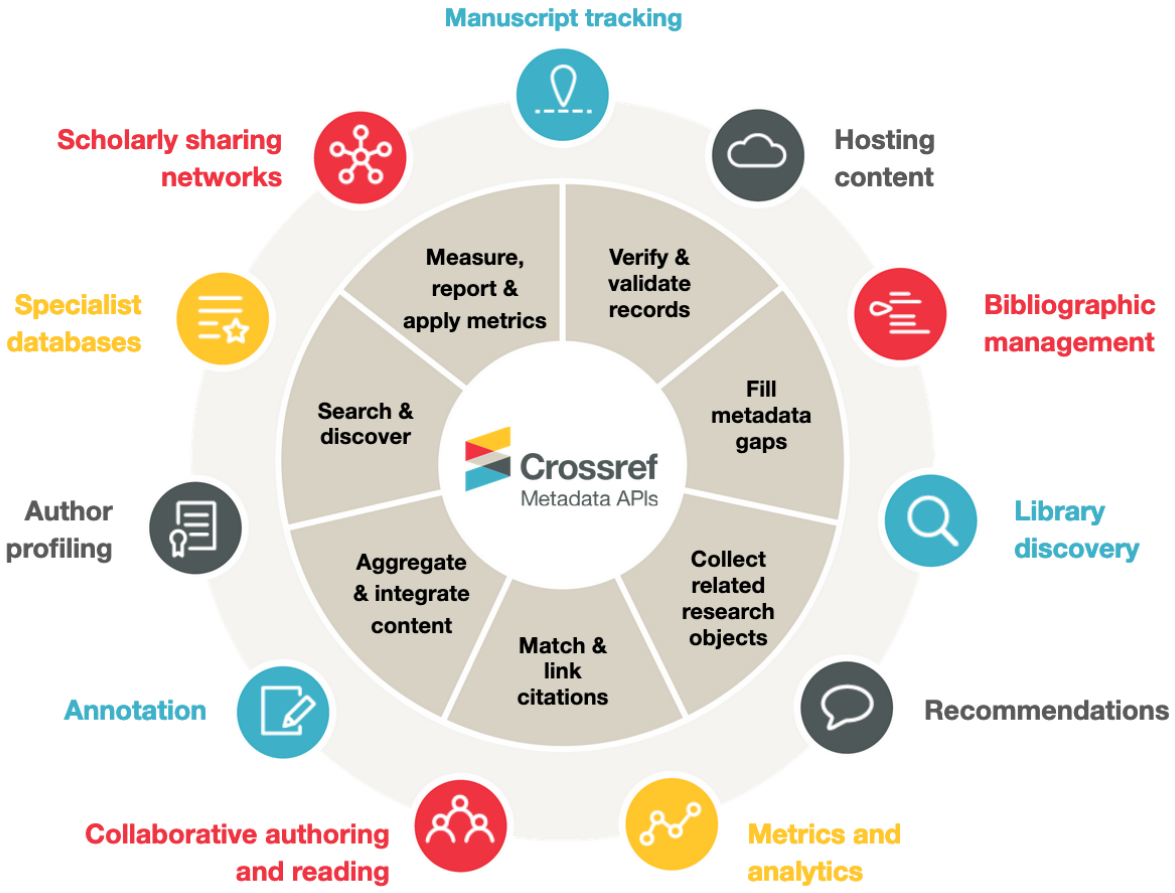
Intra-object
- Properties
- Summary
- Version
- Semantic

Inter-object
- Object groupings
- Similarity links
- Parenthood links

Global
- Semantic resources
- Logs

Metadata

Metadata:
Digital Object Identifier (DOI)
Research area code
Keywords



Example in manuscript retrieval and analysis

Metadata Extraction

The process of discovering metadata information of a dataset

Common extract steps: structural metadata then functional metadata

Essential for accessing datasets in a later phase

Techniques:

- Generic and Extensible Metadata Management System (GEMMS)
- DATAMARAN

Metadata Extraction: GEMMS

A framework for extracting metadata from heterogeneous sources, which is then stored in an extensible metamodel.

- Detects source format
- Initiates a corresponding parser
- Tree-structured inference algorithm
  - Structured vs. Semi-structured

Metadata Extraction: DATAMARAN

A three-step algorithmic approach to extract structures from semi-structured log files.

- Generates candidate structure templates
- Redundant templates reduction
- Further pruning

Metadata Modelling

Objective: keep data lake Findable, Accessible, Interoperable and Reusable. (FAIR guiding principles)

Structure and organize the metadata

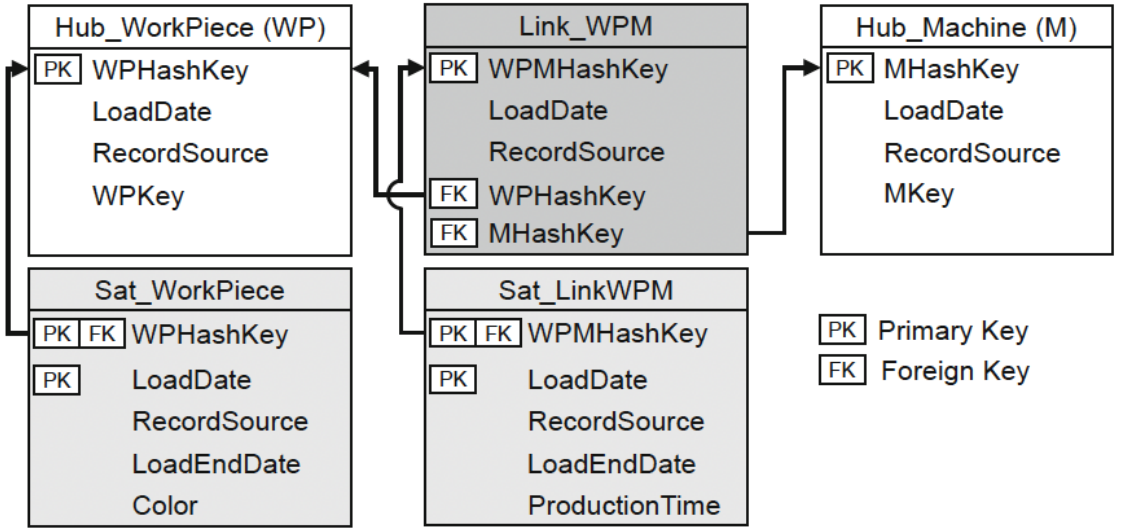
Two category of metadata modelling techniques:

- Graph models
- Data Vaults

Metadata Modelling

Data Vaults:

- Can store structured or semi-structured data
- Three key elements:
  - Hubs
    - A hub contains the business key of the business object it represents
  - Links
    - Represents the associations or hierarchies between hubs
  - Satellites
    - Stores additional information to hubs and links



Metadata Modelling

Graph models

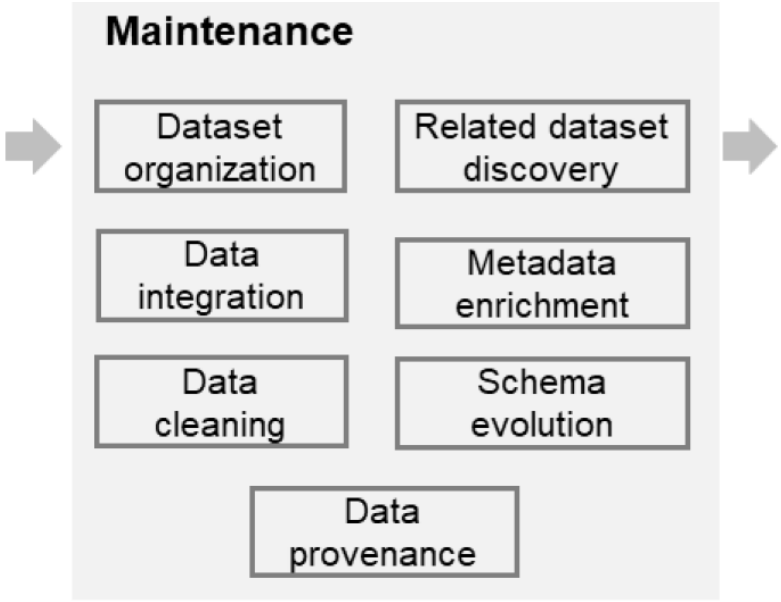
- Inspired from knowledge graphs from the linked data and semantic web communities
- Three subcategories:
  - Data provenance-centered
  - Similarity-centered
  - Composition-centered

- Nodes of the graph can be merged based on lexical and string similarities
- More flexibility than data vaults

Three-Tier Architecture: Maintenance

General management and organization of ingested datasets

A key component to keep data lake from being a data swamp



Data Organization

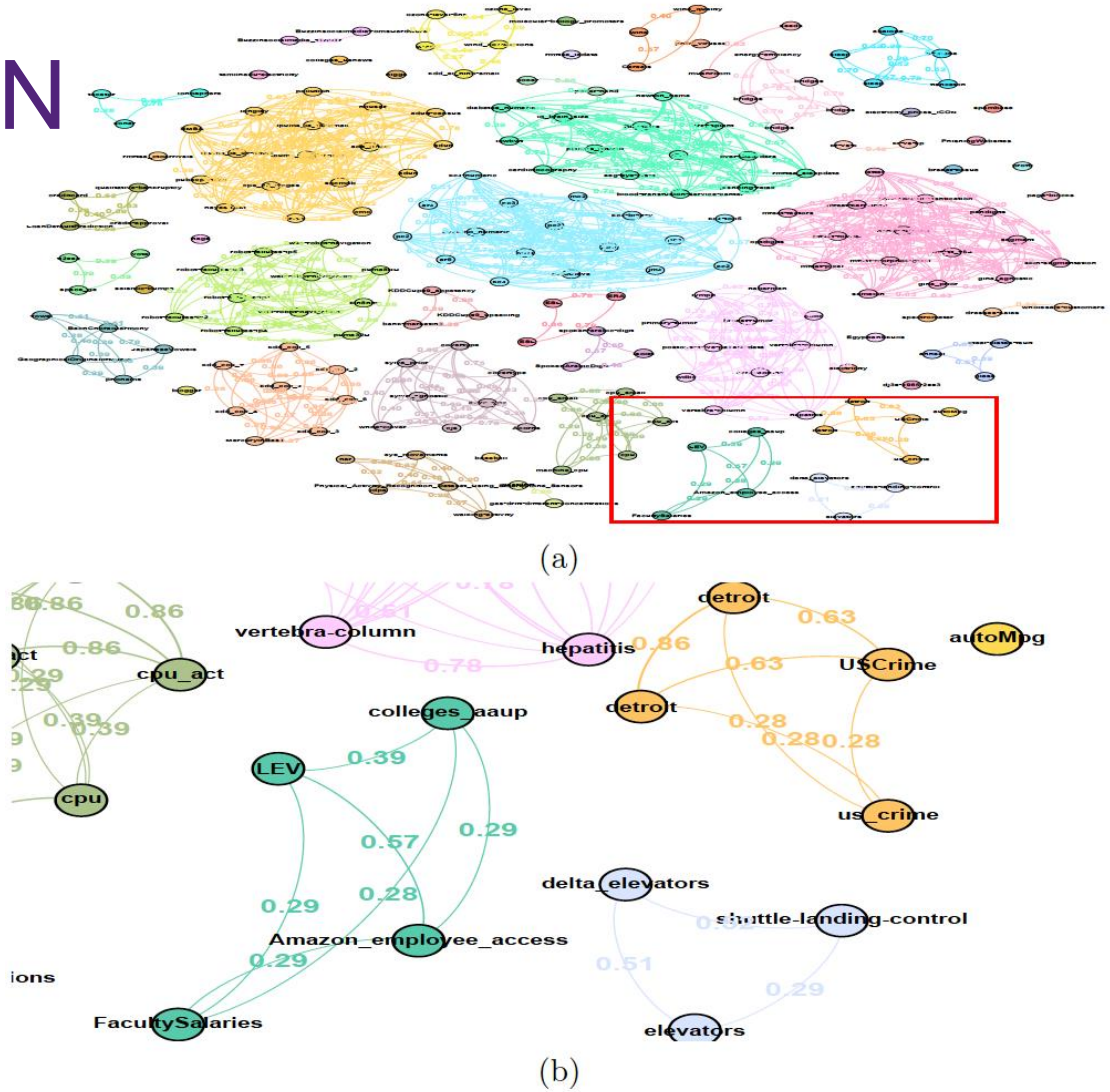
How to structure and navigate the massive heterogeneous datasets in data lakes.

Three main categories:

- Catalog-based organization
- Classification model-based organization
- Directed Acyclic Graphs (DAG)-based organization

Data Organization: DS-KNN

- Dataset organization problem can be a classification problem.
- Incrementally adds every dataset into a new or existing category by applying KNN search.



DS-KNN: STEPS

Preparation: Feature extraction

- General dataset level information

General	Number of Instances	The number of instances in the dataset
	Number of Attributes	The number of attributes in the dataset
	Dimensionality	The ratio of number of attributes to number of instances
Attributes by Type	Number per Type	The number of attributes per type (Nominal or Numerical)
	Percentage per Type	The percentage of attributes per type (Nominal or Numerical)

DS-KNN: STEPS

Preparation: Feature extraction

- General dataset level information
- Information aggregated from samples

Nominal Attributes	Average Number of Values	The average number of distinct values per nominal attribute
	Standard Deviation of Number of Values	The standard deviation in the number of distinct values per nominal attribute
	Minimum/Maximum Number of Values	The minimum and maximum number of distinct values per nominal attribute
Numeric Attributes	Average Numeric Mean	The average of the means of all numeric attributes
	Standard Deviation of the Numeric Mean	The standard deviation of the means of the numeric attributes
	Minimum/Maximum Numeric Mean	The minimum and maximum mean of numeric attributes

DS-KNN: STEPS

Preparation: Feature extraction

- General dataset level information
- Information aggregated from samples
- Special cases: missing values

Missing Values	Missing Attribute Count	The number of attributes with missing values
	Missing Attribute Percentage	The percentage of attributes with missing values
	Minimum/Maximum Number of Missing Values	The minimum and maximum number of instances with missing values per attribute
	Minimum/Maximum Missing Values Percentage	The minimum and maximum percentage of instances with missing values per attribute
	Mean Number of Missing Values	The mean number of missing values from each attribute
	Mean Percentage of Missing Values	The mean percentage of missing values from each attribute

DS-KNN: STEPS

Preparation: Feature extraction

- Attribute: statistical or distribution-based

Distance calculation:

- Edit distance

Data Organization: DS-KNN

Algorithm 1: DS-kNN Categorization of a dataset ingested in a Data Lake

Input: A new ingested dataset D_a, each existing dataset D_b in the data lake DL, Dataset-level meta-features distance metrics MF for each pair of datasets $\{D_a, D_b\}$, the category Cat_{D_b} for each existing dataset in the DL, the classification model $M_{ds-prox}$, algorithmic parameters: the number k of nearest neighbours, and the similarity score threshold c_{rel}

Output: The set SP of the ingested dataset and its similarity scores $Sim(D_a, D_b)$ and category Cat_{D_b} for each pair $\{D_a, D_b\}$ passing c_{rel}, the set $SP\text{-}Top$ of top matching k datasets and their categories, the assigned category for the new dataset Cat_{D_a}

$SP \leftarrow \emptyset$;

Algorithm 1: DS-kNN Categorization of a dataset ingested in a Data Lake

Input: A new ingested dataset D_a, each existing dataset D_b in the data lake DL, Dataset-level meta-features distance metrics MF for each pair of datasets $\{D_a, D_b\}$, the category Cat_{D_b} for each existing dataset in the DL, the classification model $M_{ds-prox}$, algorithmic parameters: the number k of nearest neighbours, and the similarity score threshold c_{rel}

Data Organization: DS-KNN

Algorithm 1: DS-kNN Categorization of a dataset ingested in a Data Lake

Input: A new ingested dataset D_a, each existing dataset D_b in the data lake DL, Dataset-level meta-features distance metrics MF for each pair of datasets $\{D_a, D_b\}$, the category Cat_{D_b} for each existing dataset in the DL, the classification model $M_{ds-prox}$, algorithmic parameters: the number k of nearest neighbours, and the similarity score threshold c_{rel}

Output: The set SP of the ingested dataset and its similarity scores $Sim(D_a, D_b)$ and category Cat_{D_b} for each pair $\{D_a, D_b\}$ passing c_{rel}, the set $SP\text{-}Top$ of top matching k datasets and their categories, the assigned category for the new dataset Cat_{D_a}

$SP \leftarrow \emptyset$;

$SP\text{-}Top \leftarrow \emptyset$;

foreach $\{D_a, D_b\} \subset DL$ *and* $a \neq b$ **do**

$[D_a, D_b, Sim(D_a, D_b)] = M_{ds-prox}(MF_{\{D_a, D_b\}})$;

number k of nearest neighbours, and the similarity score threshold c_{rel}

Output: The set SP of the ingested dataset and its similarity scores $Sim(D_a, D_b)$ and category Cat_{D_b} for each pair $\{D_a, D_b\}$ passing c_{rel}, the set $SP\text{-}Top$ of top matching k datasets and their categories, the assigned category for the new dataset Cat_{D_a}

$SP \leftarrow \emptyset$;

$SP\text{-}Top \leftarrow \emptyset$;

Data Organization: DS-KNN

foreach $\{D_a, D_b\} \subset DL$ *and* $a \neq b$ **do**
  $[D_a, D_b, Sim(D_a, D_b)] = M_{ds-prox}(MF_{\{D_a, D_b\}})$;
  if $Sim(D_a, D_b) > c_{rel}$ **then**
    $SP \leftarrow SP \cup \{[D_a, D_b, Sim(D_a, D_b), Cat_{D_b}]\}$;
  end
end
$SP\text{-}Top$ = Top-k_Nearest_Neighbours(SP, k); \\Retrieve the subset of the highest ranking k-pairs by similarity score
$Cat_{D_a} = Top\text{-}category(SP\text{-}Top)$; \\Get category with majority vote from Top-k
if $(Cat_{D_a} = NULL)$ **then**
  $Cat_{D_a} =' Outlier'$;
end

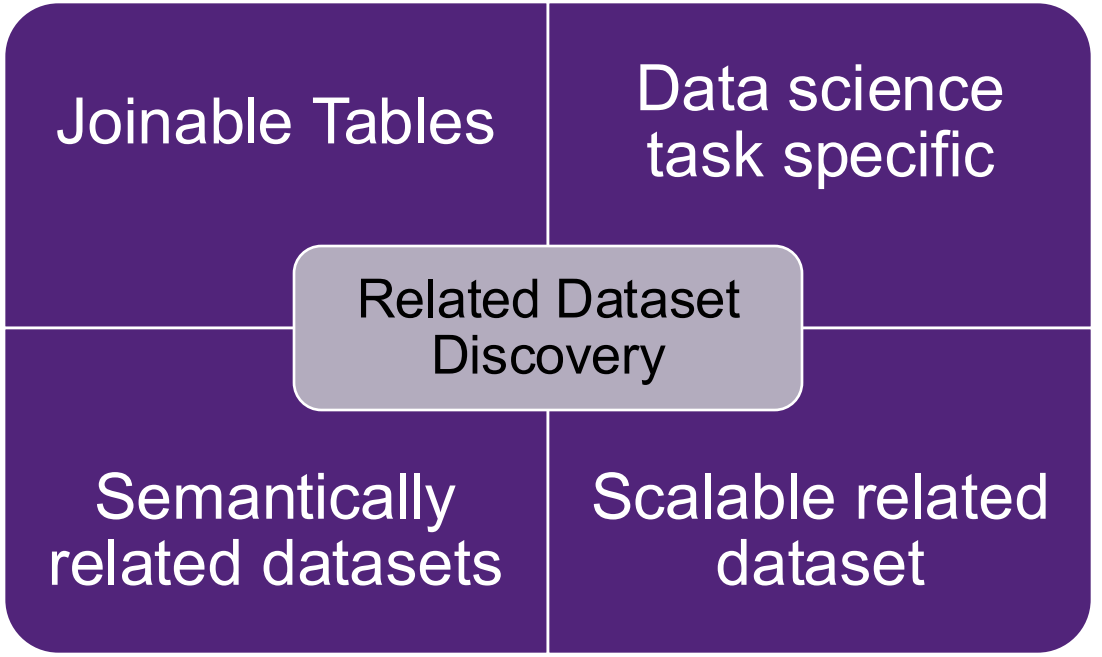
Related Dataset Discovery

Find a subset of relevant datasets that are similar or complementary to a given dataset.

- E.g., with similar attribute names or overlapped instance values.

Dataset purpose and relatedness

- How to define relatedness?



Relatedness:

Features:

- Instance value overlap
- Attribute: (name, data type, value range, …)
- PK-FK candidate
- Semantics
- Descriptive metadata
- Data value representation pattern
- (Numerical) data distribution
- …

Measures:

- Jaccard distance
- Cosine distance
- …

Metadata enrichment

Create implicit metadata from raw data in the data lake

- Enrichment vs. Extraction

Metadata enrichment	Metadata extraction
Infer from data	Directly from data
Time-costly	Time-efficient

Metadata enrichment

Three general types of metadata enrichment:

Semantic metadata enrichment

- Add synonyms and stems to extracted metadata (e.g., Keywords)

Structural metadata enrichment

- Discovering relaxed functional dependencies

Descriptive metadata enrichment

- Human in the loop:
- Adding descriptive metadata of datasets, marking datasets worth additional security attention.

Data cleaning

Discovering and fixing data quality problems

- One source / multiple source
- Schema level / sample level

Two general types

- Data cleaning by constraint inference
- Data cleaning by validation rule inference

Schema Evolution and Data Provenance

Schema Evolution

Handling the changes of schemata and integrity constraints

- Heterogeneity of the schemata and the frequency of the changes

Data Provenance

Meta information of data records, which indicates their origin, usage, status in the life cycle, etc.

- A special type of metadata

Data provenance: Google Dataset Search

Upstream datasets and downstream datasets.

Steps:

- Analysis of production logs, which provide information on which jobs read and write each dataset.
- Creating a transitive closure of this graph connecting datasets and jobs

Trade off the completeness of the provenance associations for efficiency

- materializing only the downstream and upstream relations within a few hops

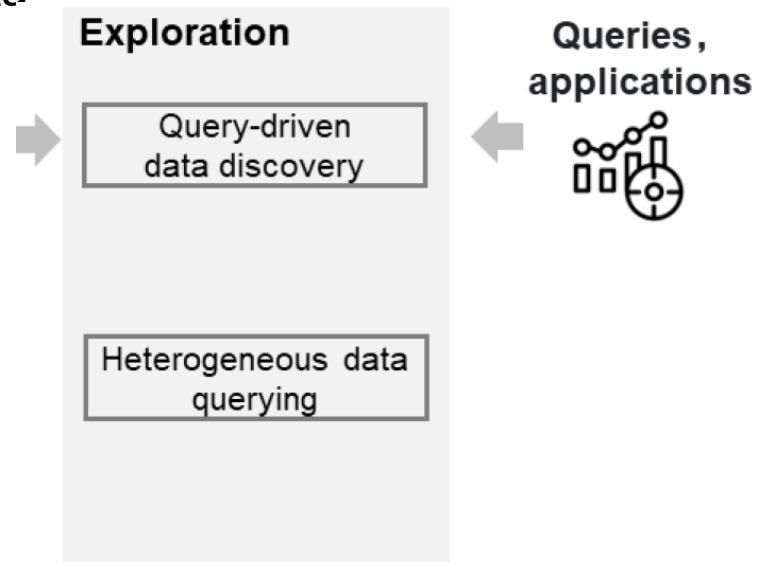
Three-Tier Architecture: Exploration

Allow end user to access the data lake

Challenges: large number of ingested sources, and the heterogeneity of data

- User may never have information on all datasets

External applications: visualization, machine learning, etc.



Exploration: Query-driven data discovery

Search a data lake based on the measured relatedness

Given the input of description of dataset, returns the top-k most related datasets. (Three ways of exploration)

- Given a table, and a column, find top-k datasets that are overlap with the source table.
- Given a table, the system returns top-k tables that contain relevant attributes for populating T
- Given a table and the search type for external applications (e.g., a data science task), the system returns top-k tables that are most relevant.

Exploration: Query-driven data discovery

Search a data lake based on the measured relatedness

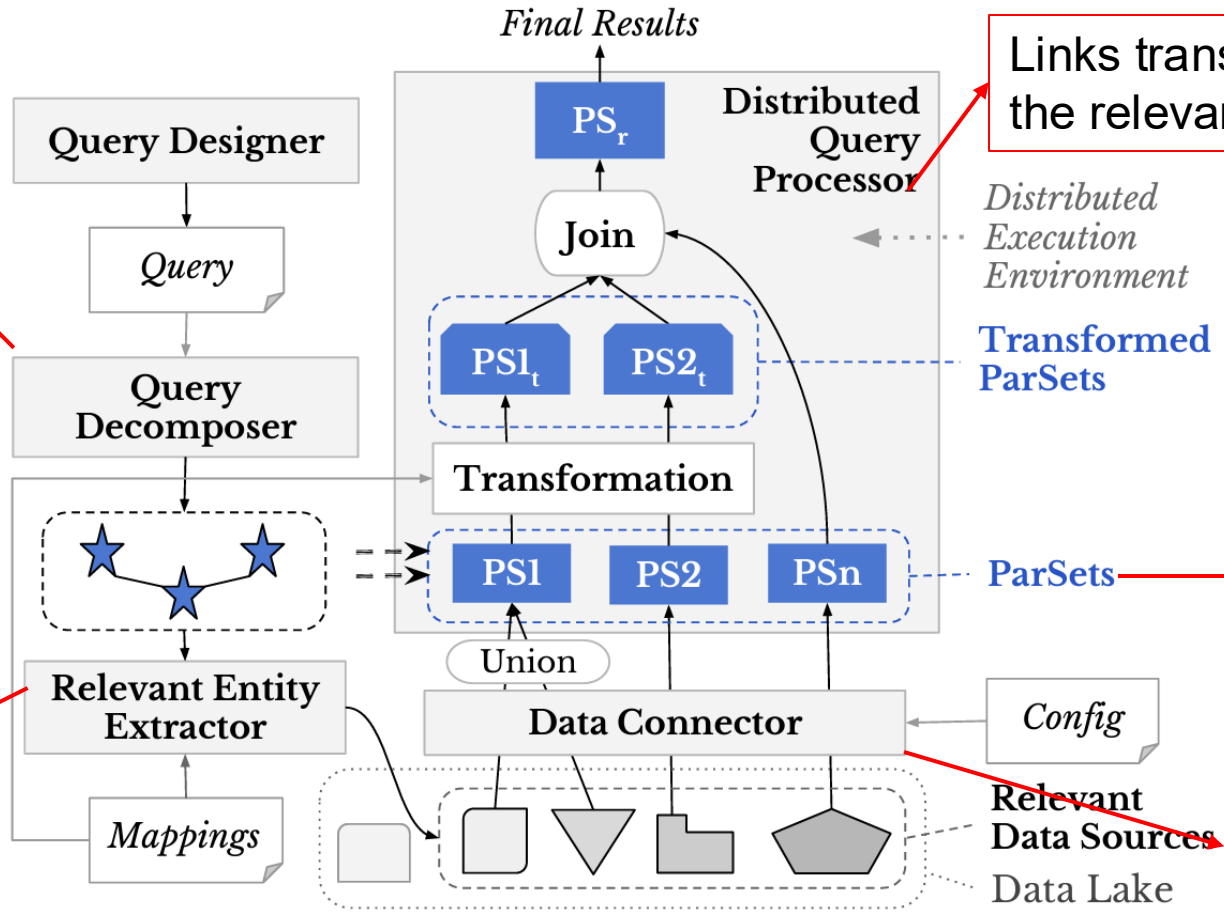
Given the input of description of dataset, returns the top-k most related datasets.

Querying methods and indexing.

- Index-based similarity measurement. (E.g., Local sensitivity hashing)

Exploration: Heterogeneous data querying

Validate and analyse queries provided by a user.



Links transformed into joins between the relevant detected data entities

Data structures that can be distributed and operated on in parallel.

Search in the mappings for entities that are mapped to every predicate of the star.

Once relevant data entities are detected, they are connected to the distributed execution environment

Topics

Relational Database vs. Data Warehouse

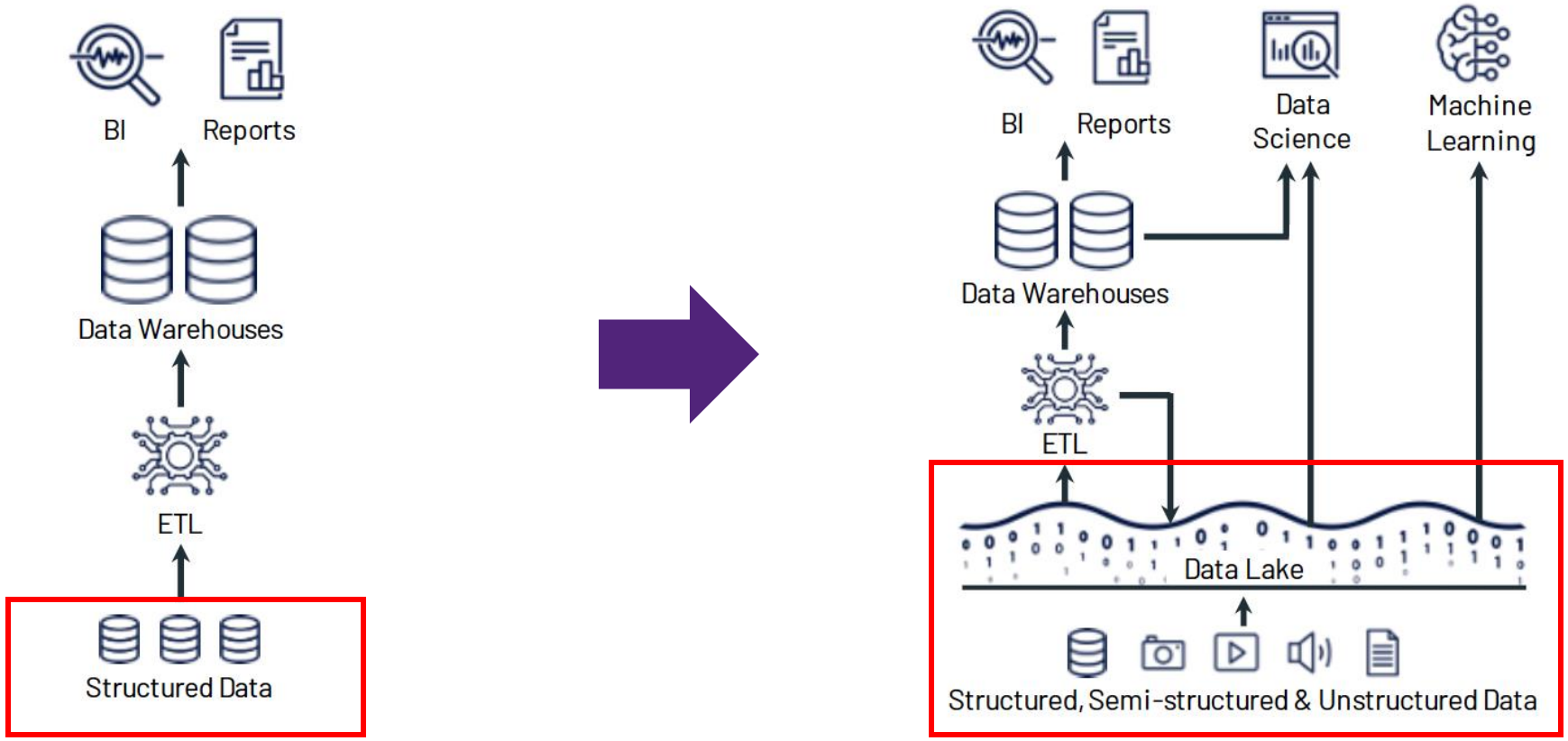
- What motivates data lake?

Data Lake: Conceptual Design

- Data Lake history
- Data Lake architecture overview
- Data Lake components

From Lake To Lakehouse – Modern Big Data Management Systems

Perform Analytics with Data lake



Common Problems in Existing Data Architectures:

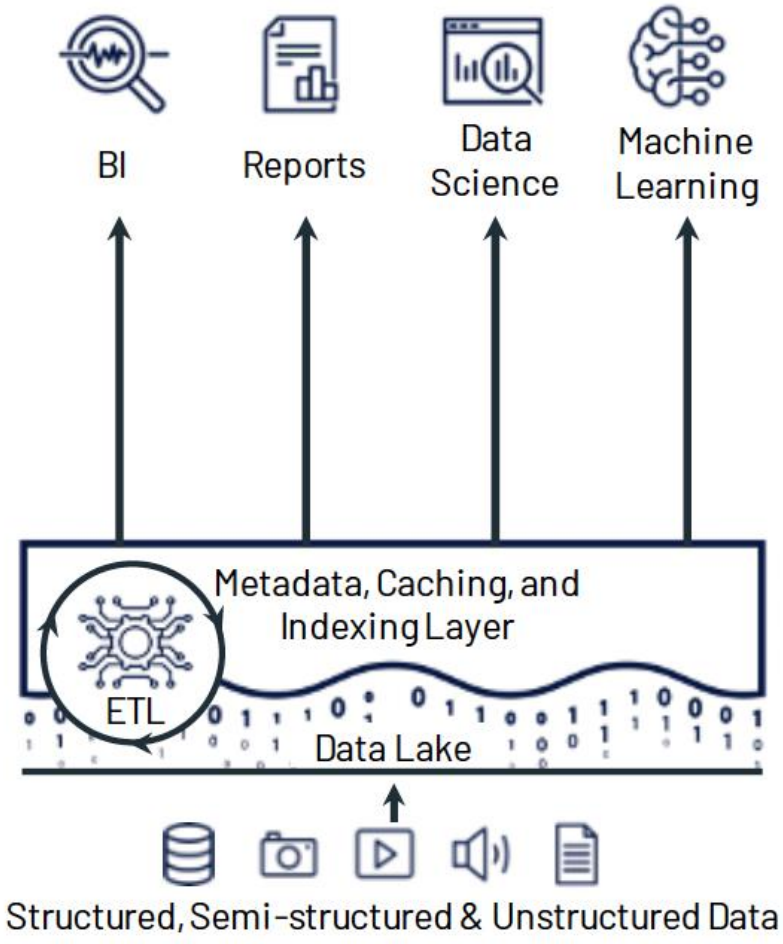
Reliability: Keeping the data lake and warehouse consistent is difficult and costly.

Data staleness: The data in the warehouse is stale compared to that of the data lake, with new data frequently taking days to load.

Limited support for advanced analytics.

Total cost of ownership: double the storage cost

From Data Lake to Data Lakehouse

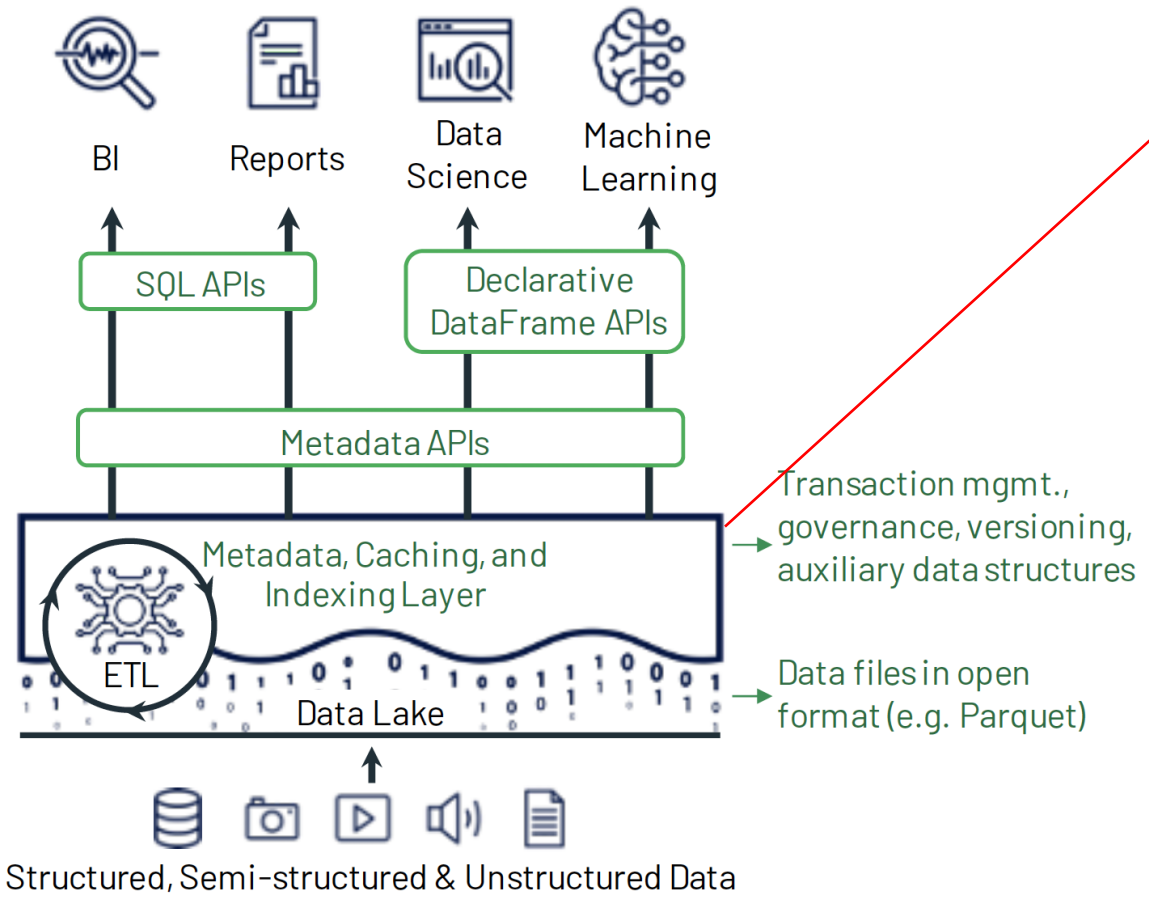


- Simultaneously supporting ETL/ELT processes and raw data storage
- Low-cost storage

DELTA LAKE

amazon web services | S3

Data Lakehouse



Optimized SQL performance: caching, auxiliary data structures such as indexes and statistics, and data layout optimizations.

SQL Performance in a Lakehouse

Key challenge: Maintain SQL performance while giving up a significant portion of the data independence

Caching

- cache files from the cloud object store on faster storage devices such as SSDs and RAM on the processing nodes

Auxiliary data

- data that helps optimize queries in auxiliary files. E.g., column min-max statistics

Data layout

- Related data record should be clustered together

Delta Lake as the data lake storage

Traditional advantage of cloud storage

- pay-as-you-go billing,
- economies of scale
- scale computing and storage resources separately

Achieving performant and mutable table storage over these systems is challenging:

- multi-object updates are not atomic
- Rolling back writes is difficult

In early years of Databrick: 50% of support escalation is related to data corruption, consistency or performance issues due to cloud strategy

Delta Lake

Time Travel and Rollbacks to let users query point-in-time snapshots or rollback erroneous updates to their data.

Schema evolution allowing Delta to continue reading old files without rewriting them if a table's schema changes

Caching: Because the objects in a Delta table and its log are immutable, cluster nodes can safely cache them on local storage.

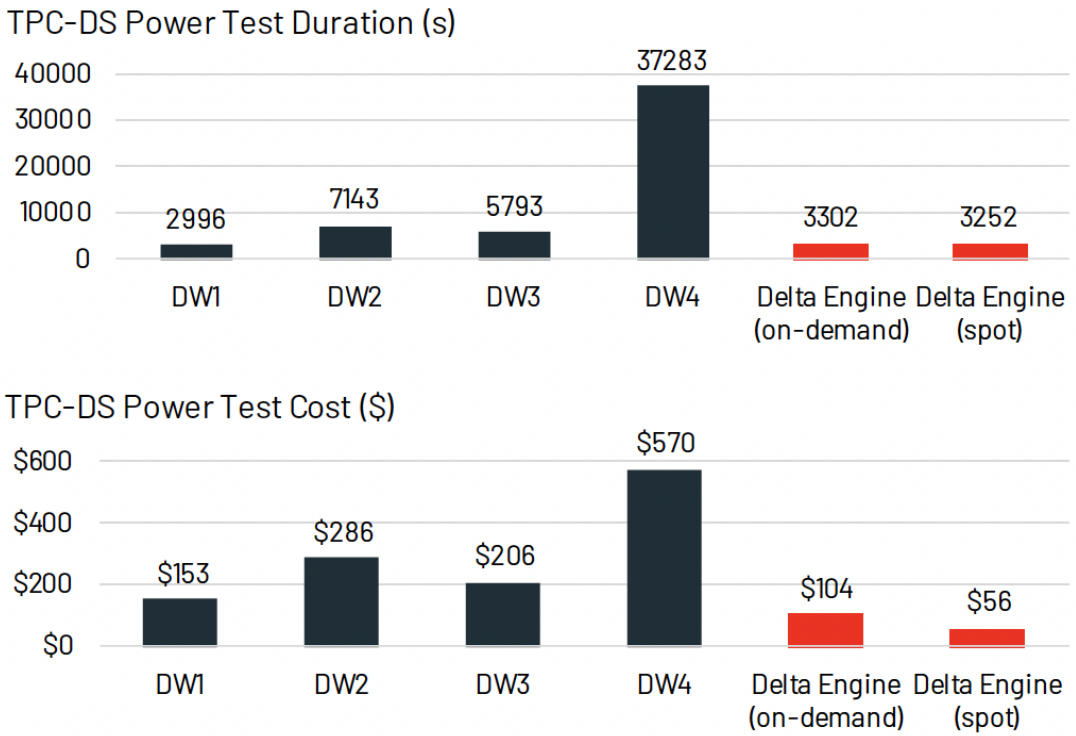
Delta Lake

Data Layout Optimization: automatically optimizes the size of objects in a table and the clustering of data records without impacting running queries.

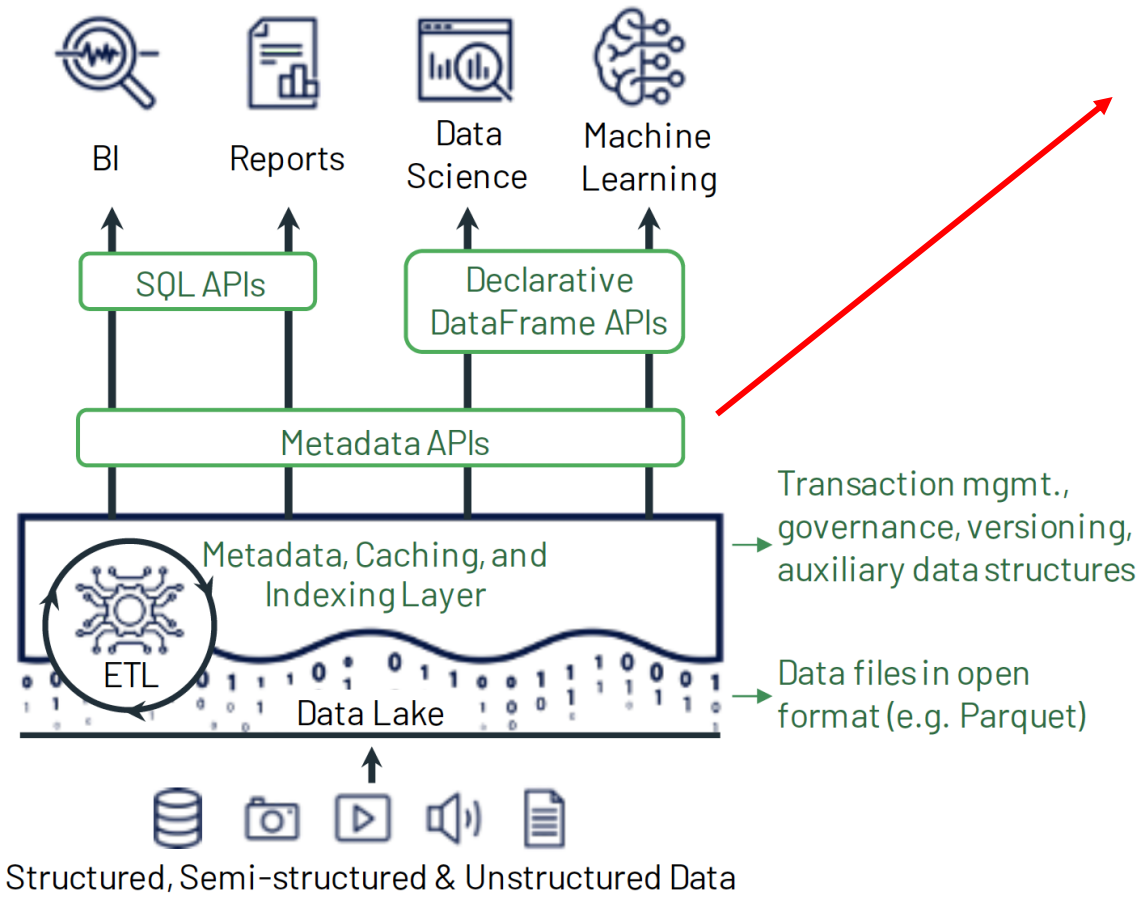
- Automatic optimization
- Z-order: make each data object contain a small range of the possible values in each of the chosen attributes, so that more data objects can be skipped when running a selective query.

SQL Performance in a Lakehouse

Cost Comparison: *popular cloud data warehouses on AWS, Azure and Google Cloud and third-party implemented DW on cloud service.*



Data Lakehouse



Transactional metadata layer on top of the object store that defines which objects are part of a table version.

Metadata Layers for Data Management

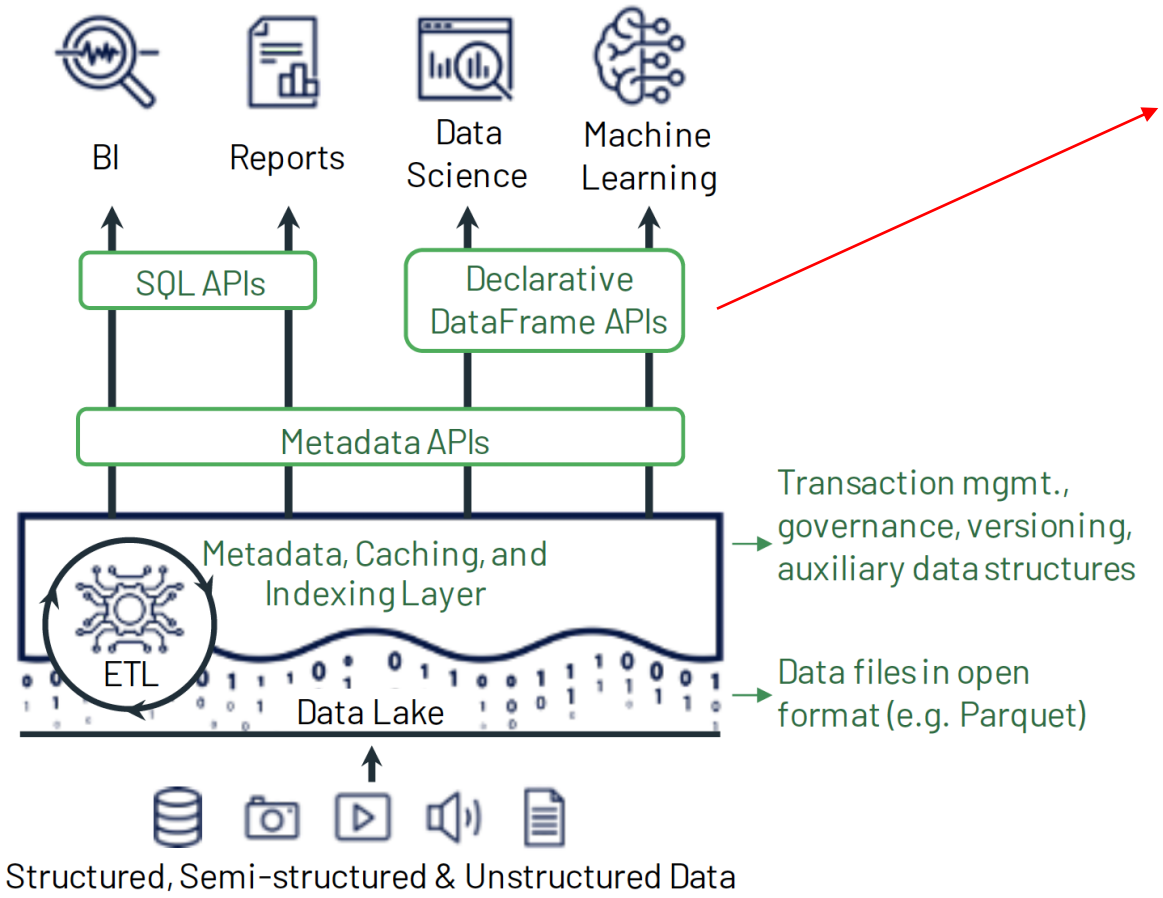
Raise the abstraction level to implement ACID transactions and other management features.

- Low-level object store or filesystem interface make operations non atomic

Stores the information about objects origin as part of a table in the data lake itself as a transaction log

- Sample level → Dataset level

Data Lakehouse



Integrated machine learning data loading APIs

Efficient Access for Advanced Analytics

Key Challenge: advanced analytics libraries are usually written using imperative code that cannot run as SQL

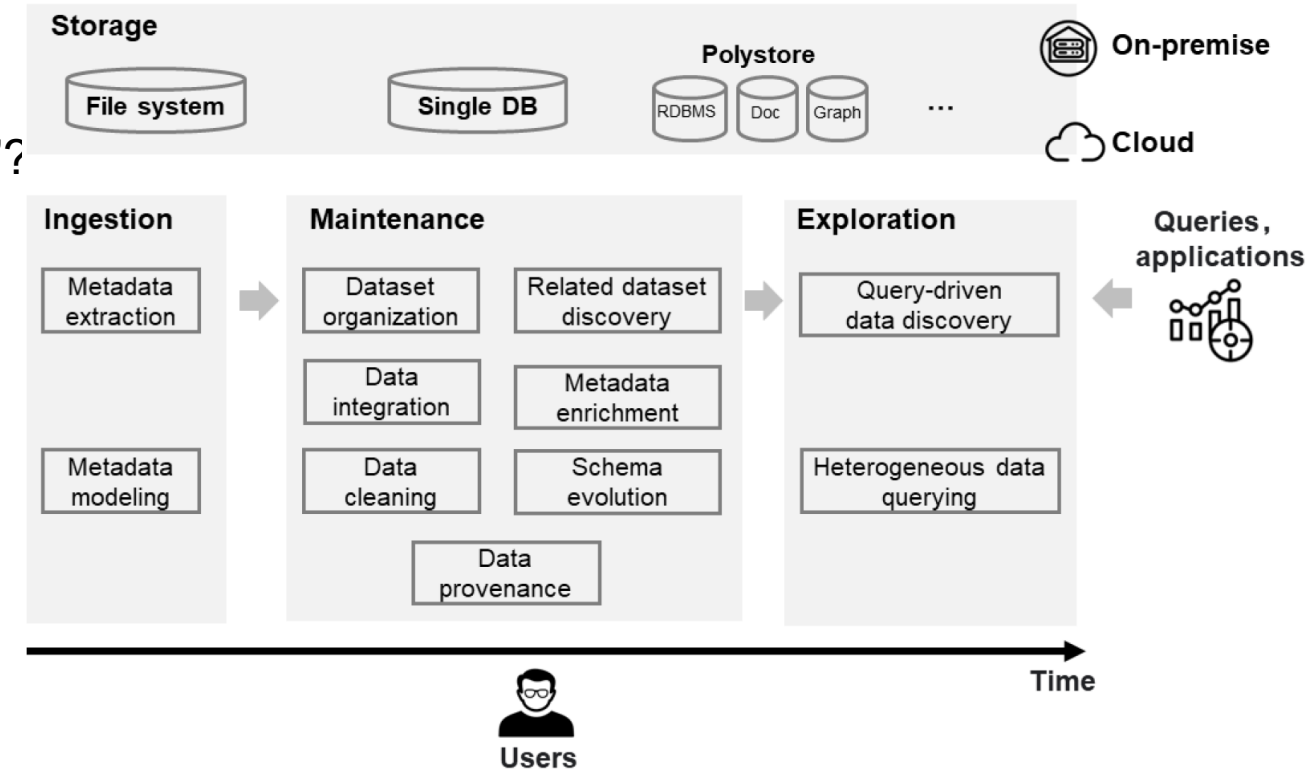
Lakehouse offers a declarative version of the DataFrame APIs used in these libraries,

- maps data preparation computations into Spark SQL query plans
- Also benefits from the optimizations in Delta Lake and Delta Engine.

Summary

Why data lake is important in the big data era?

What keeps ‘data lake’ from being ‘data swamp’?



Summary

What are the advantages of data Lakehouse?

Key considerations in Lakehouse design