Lisbon School of Economics & Management
Universidade de Lisboa

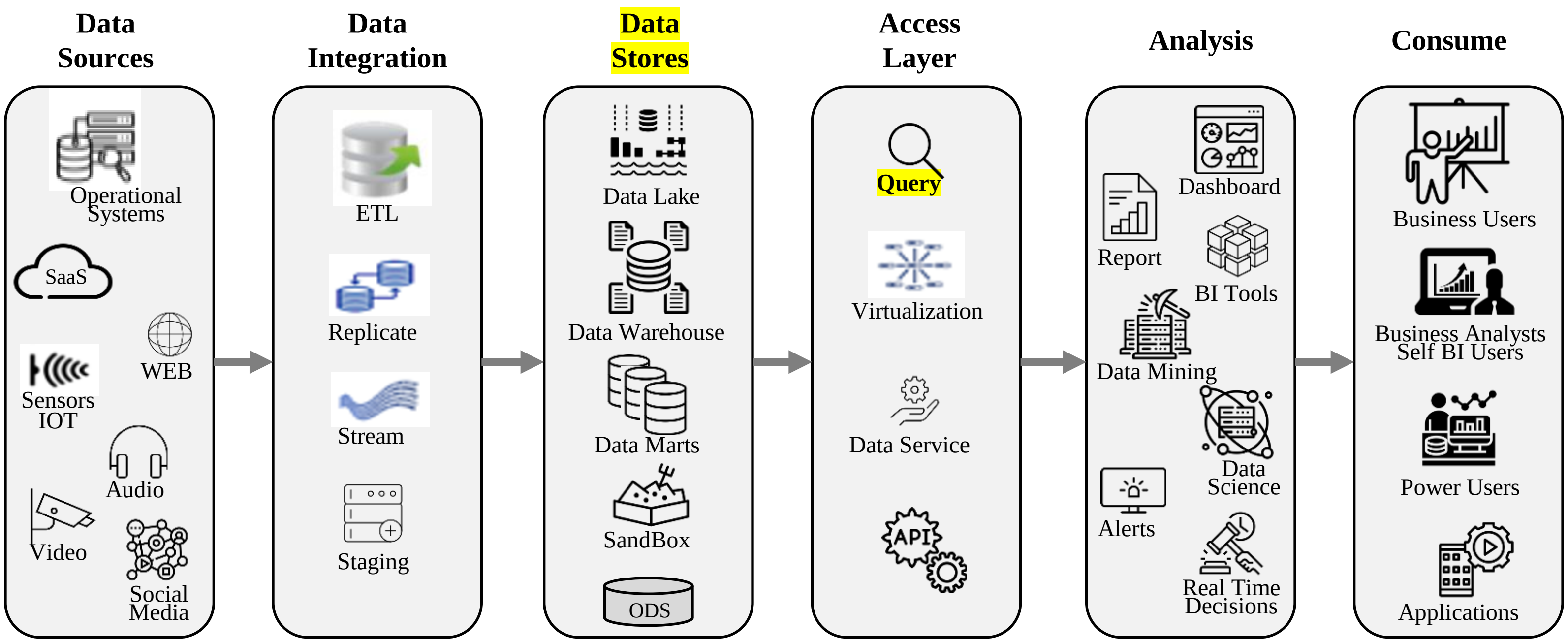
Big Data

2023/24

Eduardo Rodrigues

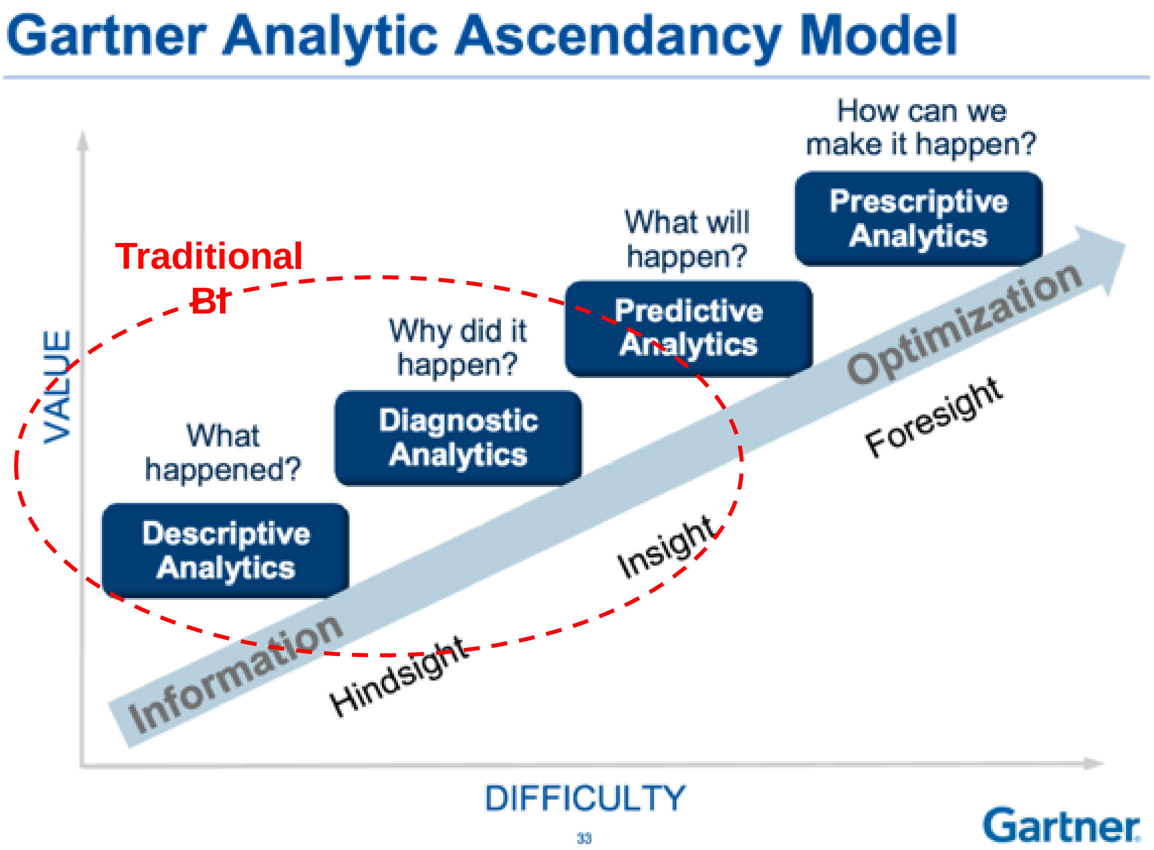
BI Architecture

BI Architecture: Key layers and components

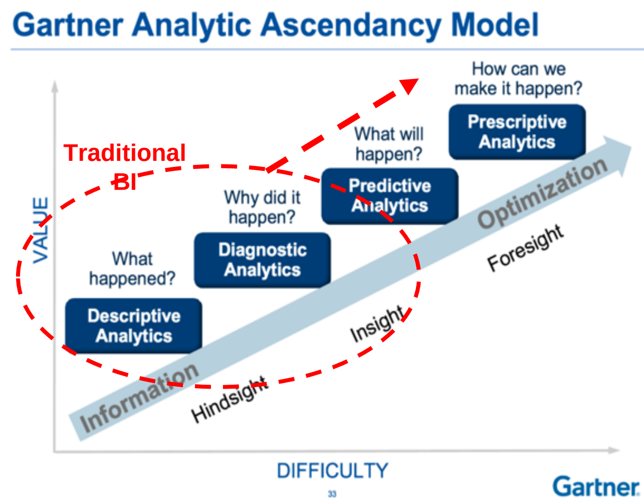


Data Governance (Architecture, Metadata, Data Quality, Security …) **& Infrastructure** (BI Tools, Cloud, DBMS, …)

What is Big Data?



What is Big Data?



Operational Systems

Big Data

What is Big Data?

Big data is defined as **collections of datasets** whose **volume, velocity or variety** is **so large** that it is difficult to store, manage, process and analyze it using traditional databases and data processing tools.

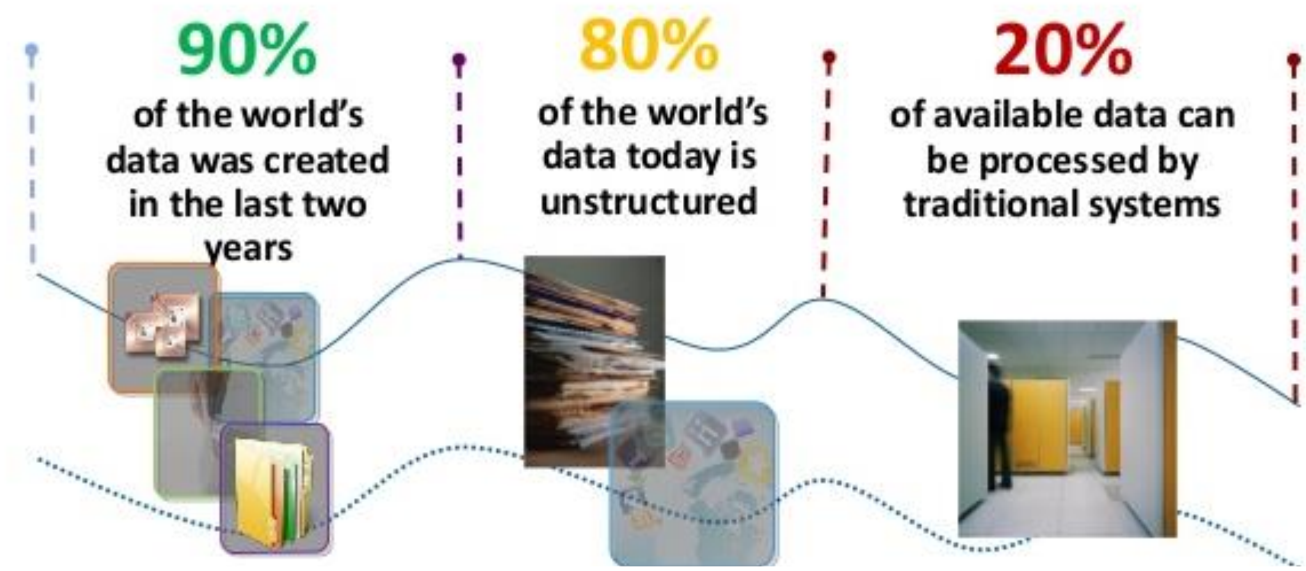
Operational Systems

With the amount of data being generated every minute by consumers and businesses worldwide, there is **significant value** to be found in Big Data analytics.

Big Data

Today, Big Data is one of the hottest buzzword around.

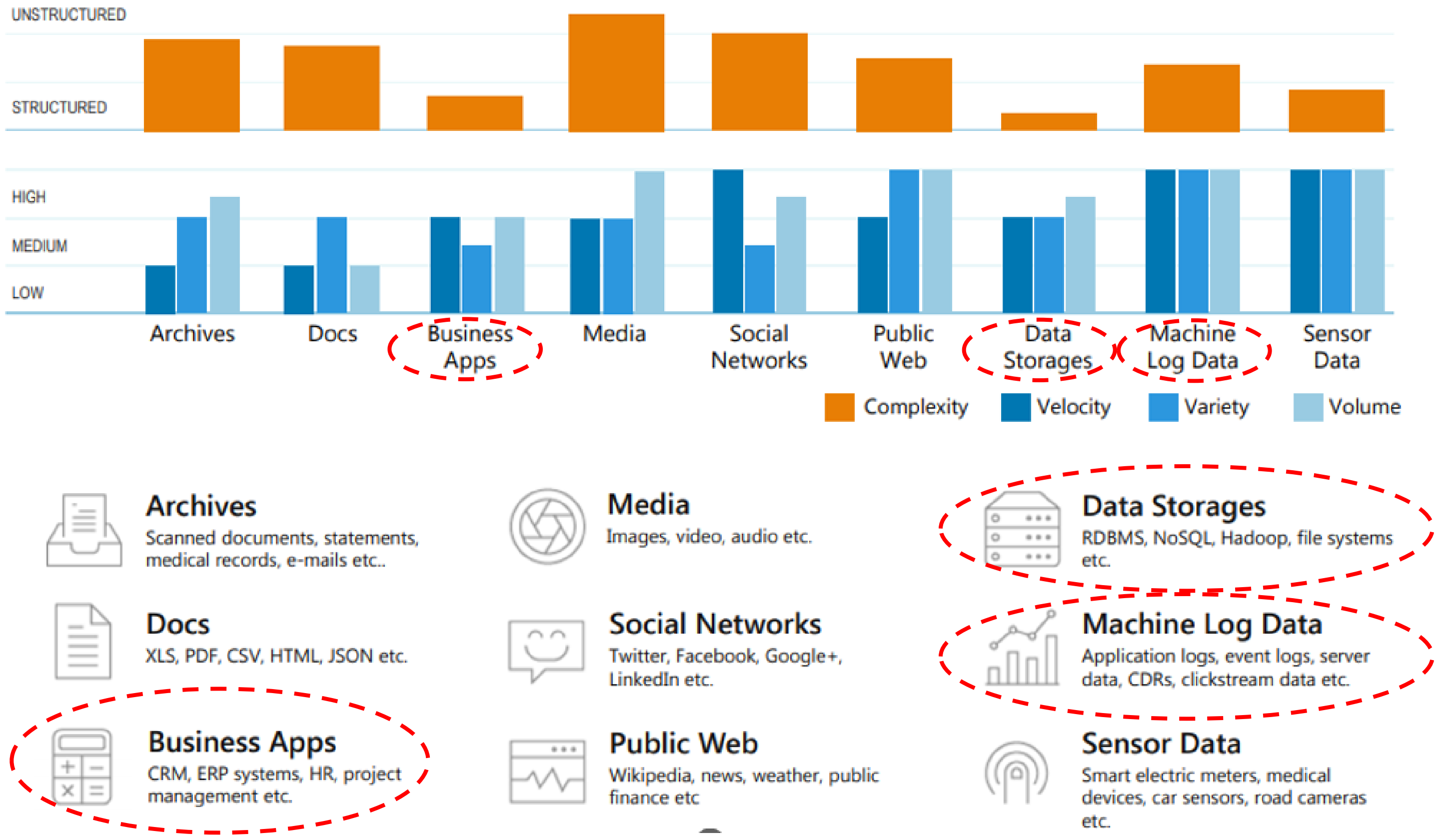
But Big Data poses new challenges



What is big data

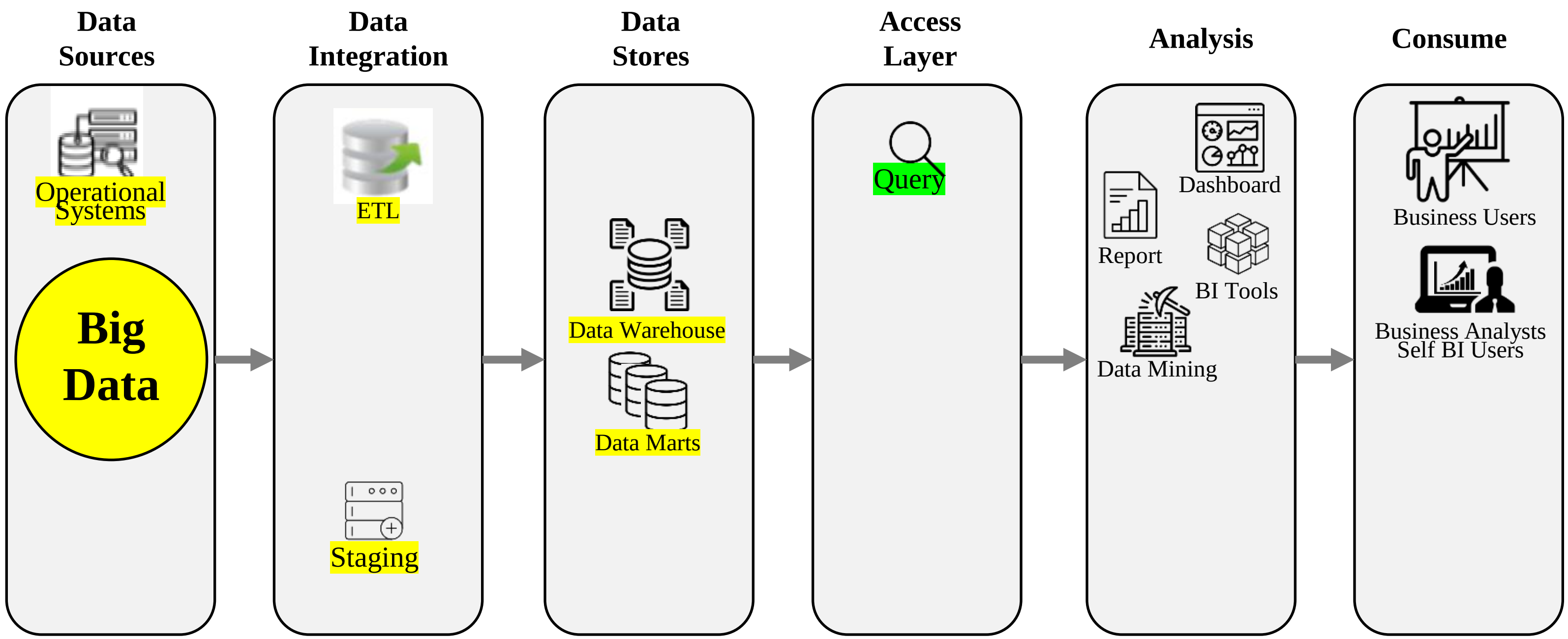
Big Data Challenges

In the recent years, there has been an exponential growth in the both structured and unstructured data generated by information technology, industrial, healthcare, Internet of Things, and other systems.



Bi Architecture

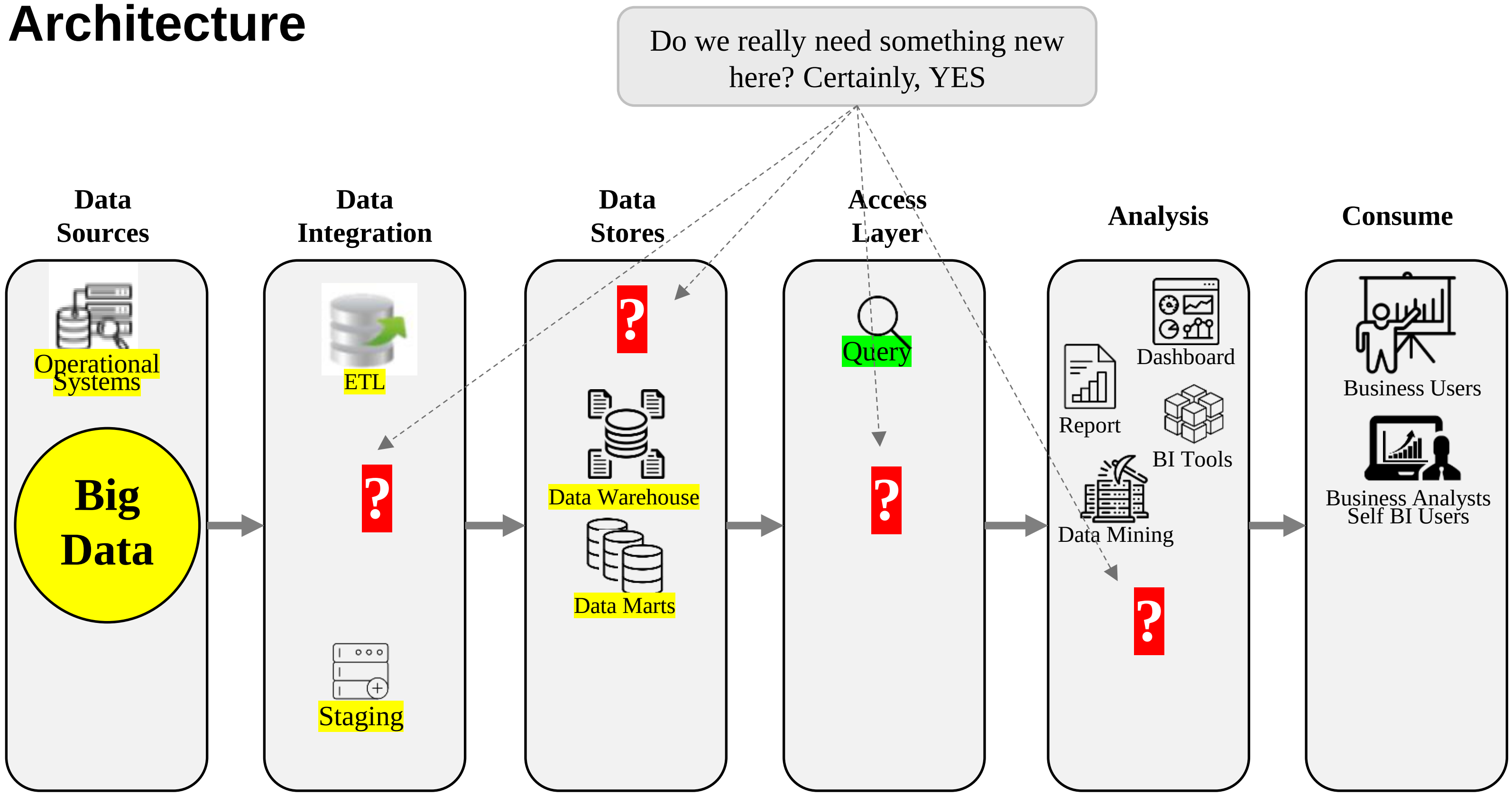
Big Data analytics is a relatively modern field of data science that explores **how large data sets can be broken down, processed** and analysed in order to systematically glean insights and information from them.



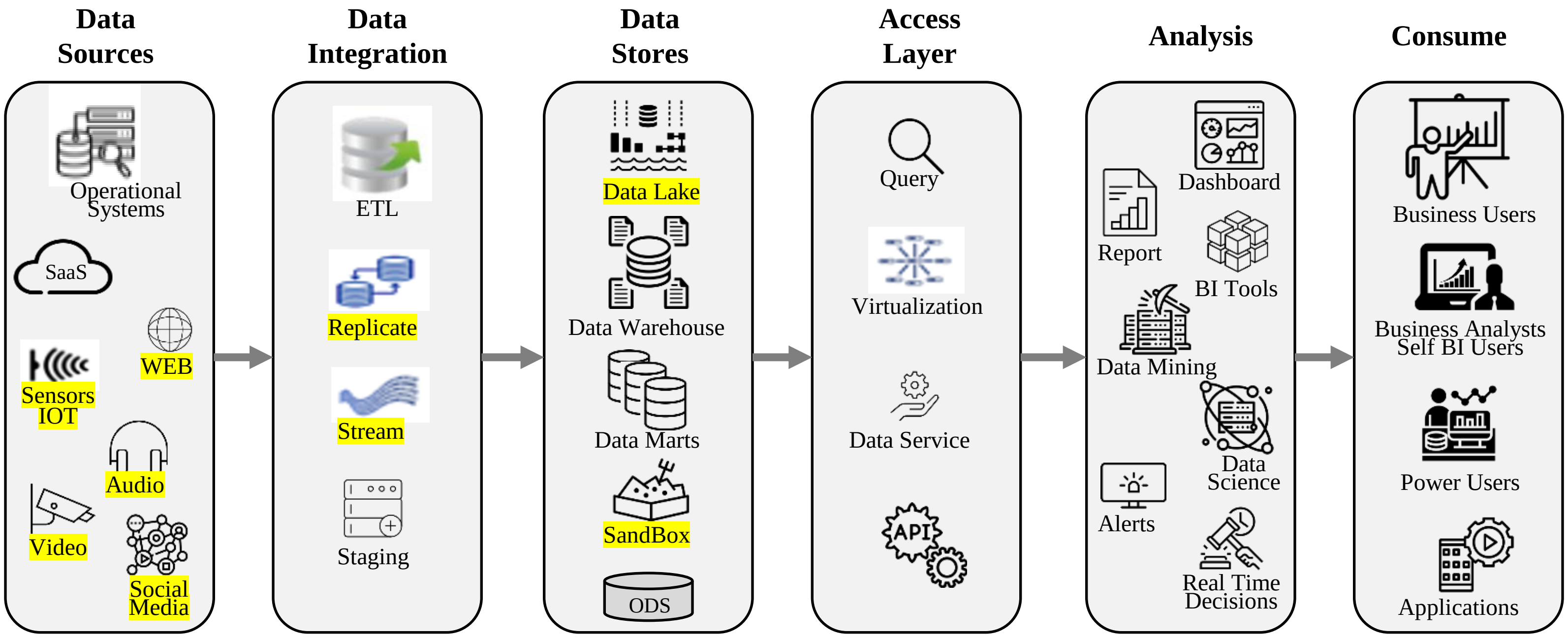
Companies with **traditional BI solutions are not able** to fully maximize the value of Big Data.

Conventional data processing solutions are not very efficient with respect to capturing, storing and analysing big data.

Bi Architecture



Bi Architecture

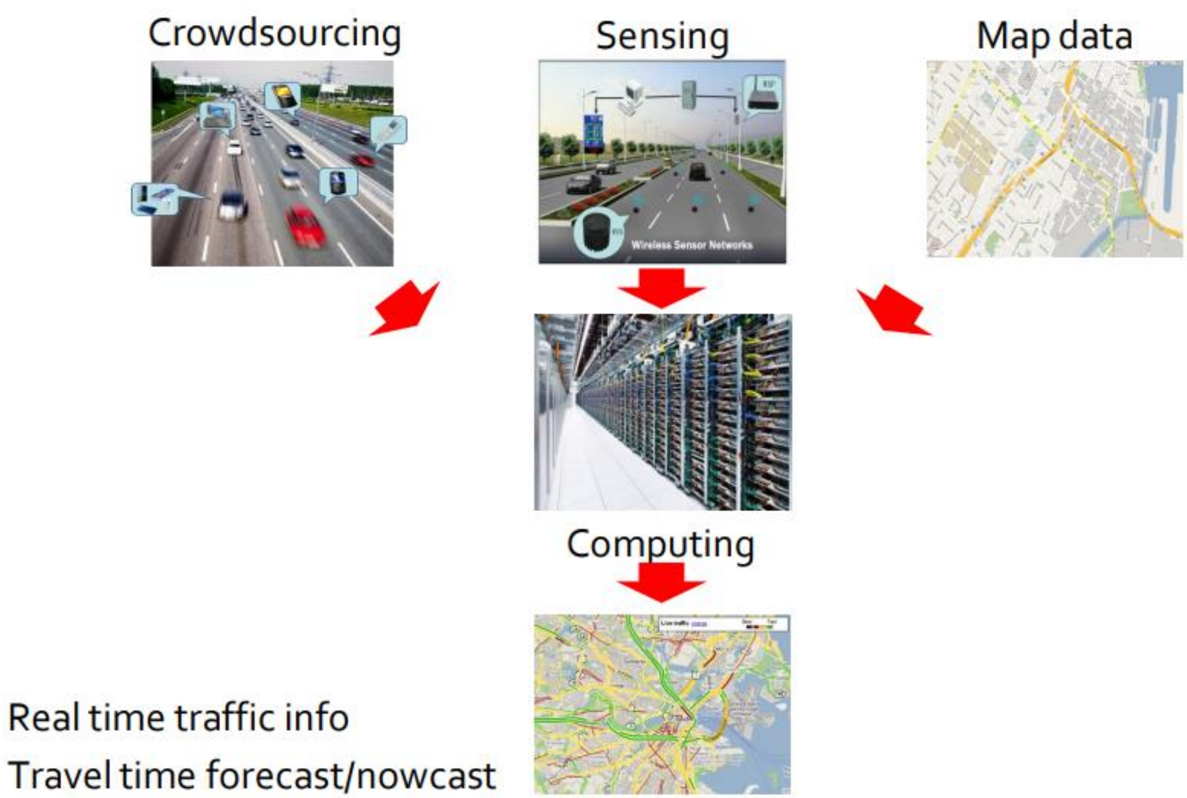


Data Governance (Architecture, Metadata, Data Quality, Security …) **& Infrastructure** (BI Tools, Cloud, DBMS, …)

Big Data

Big Data is also very important for operational systems

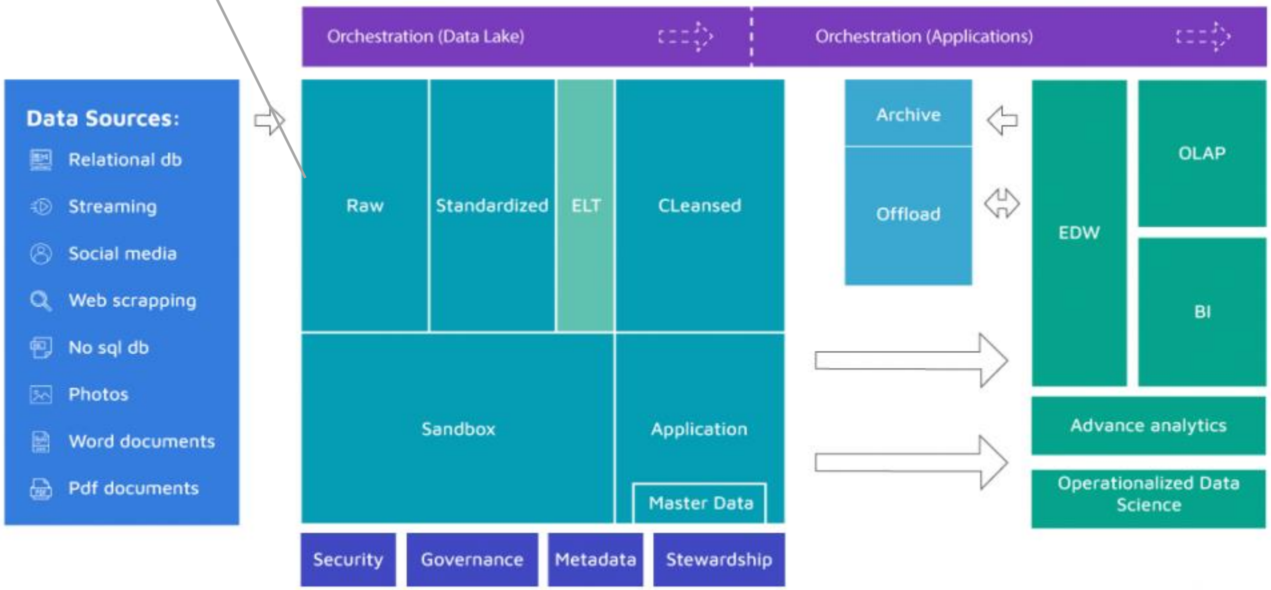
- **Example: Geographic Information Systems and Traffic Analysis**



Data Lake Architecture

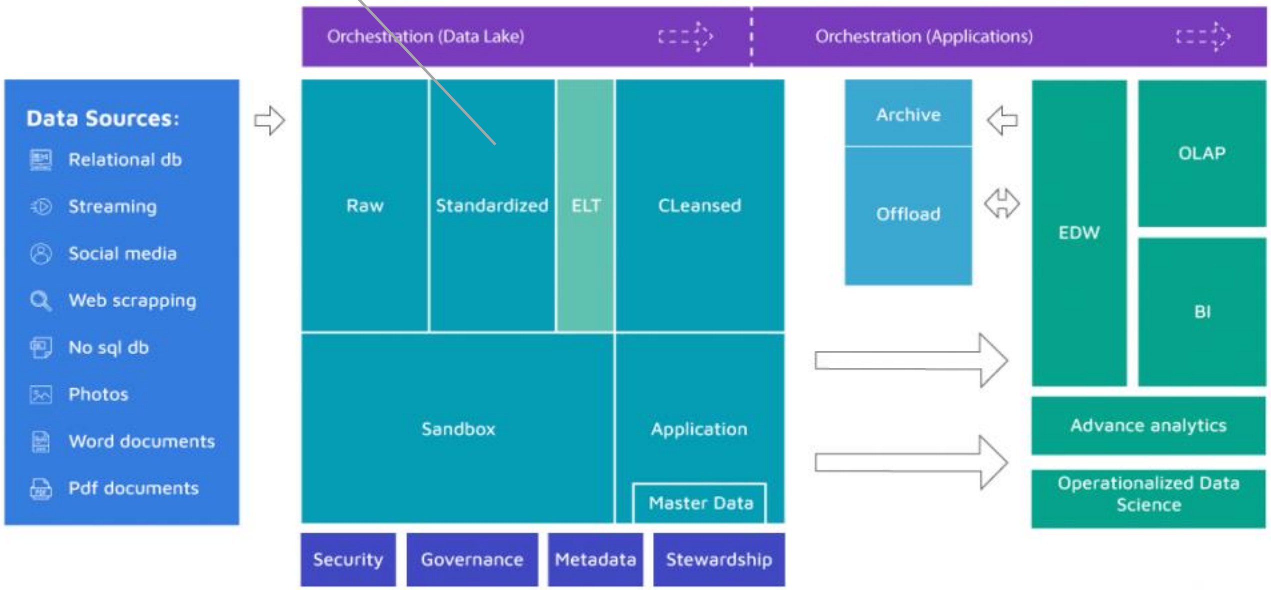
Data Lake Architecture

- **Ingest data in its native state**
- No transformation or override
- Although raw, it needs some organization (by folders⇔subject area, data source, period, market, …)
- **Users should not be granted access** to raw layer (data not ready to use => It requires knowledge and tools to be consumed)
- Similar to the staging area of DW architecture



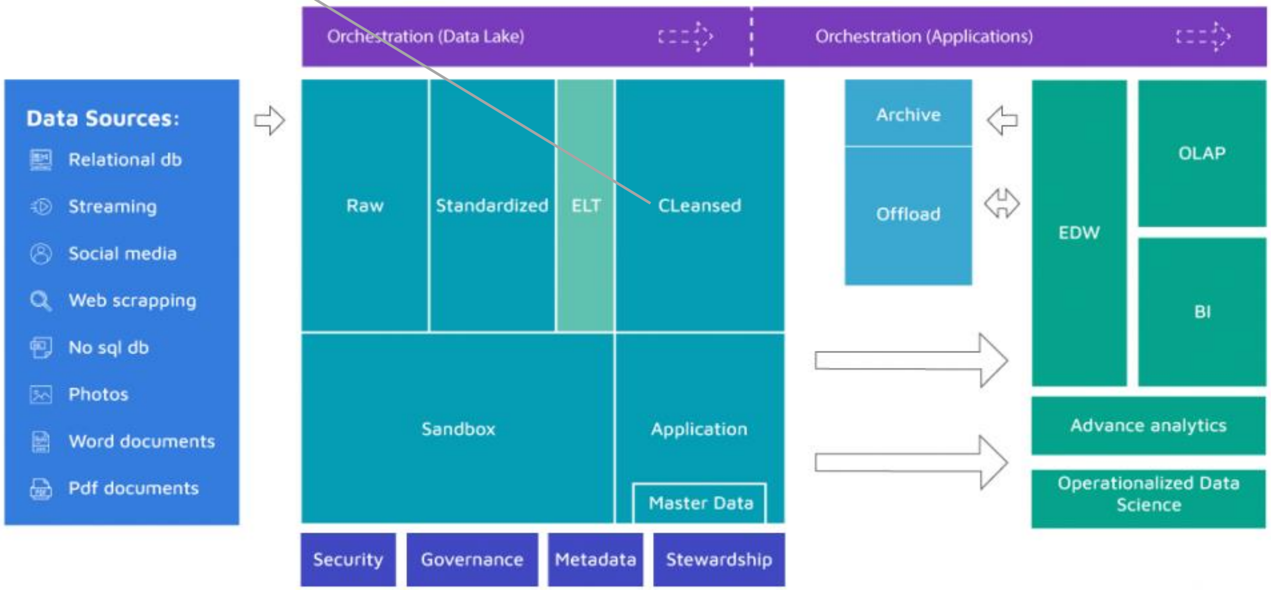
Data Lake Architecture

- **<u>Optional in most implementations</u>**. <u>But if anticipated that **DL will grow fast**, it shall be considered</u>
- The main objective of this layer is to **improve performance** in data transfer from Raw to Curated
- **Purpose for data is still not yet fully known**
- While in Raw, data is stored in its native format, in Standardized we choose the format that fits best for cleansing. **<u>The structure is the same</u>** as in the previous layer but it may be partitioned to lower grain if needed.



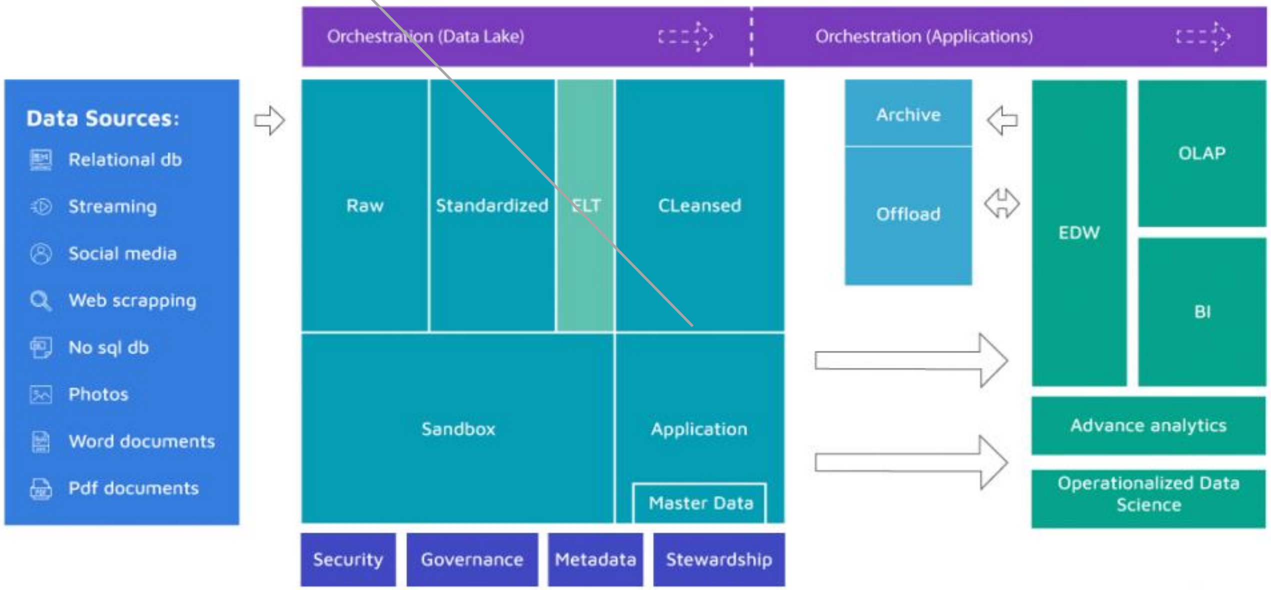
Data Lake Architecture

- Also called **Curated Layer / Conformed Layer.** Cleansing and transformations happen before this layer
- **Here we already know the purpose of the data**
- **Data is transformed into consumable data sets** and it may be stored in **files or tables**. The aim is to structure and uniform the way files/tables are stored in terms of context, encoding, format, data types and content (i.e. strings).
- Denormalization and consolidation of different objects are common, as well.
- Usually, **end users are granted access** only to this layer.



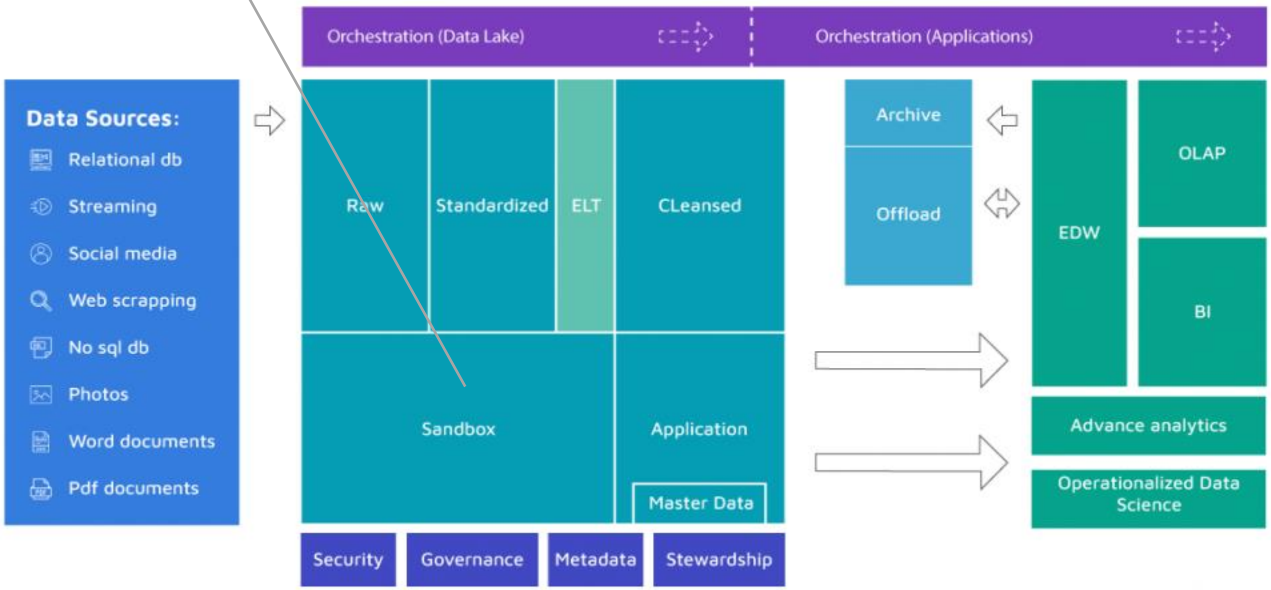
Data Lake Architecture

- Also called the **Trusted Layer/Secure Layer/Production Layer**, it is sourced from Cleansed layer and enforced with any needed business logic
- These might be surrogate keys, row level security or anything else that is specific of the application consuming this layer.
- **The structure of the data will remain the same**, as in Cleansed.



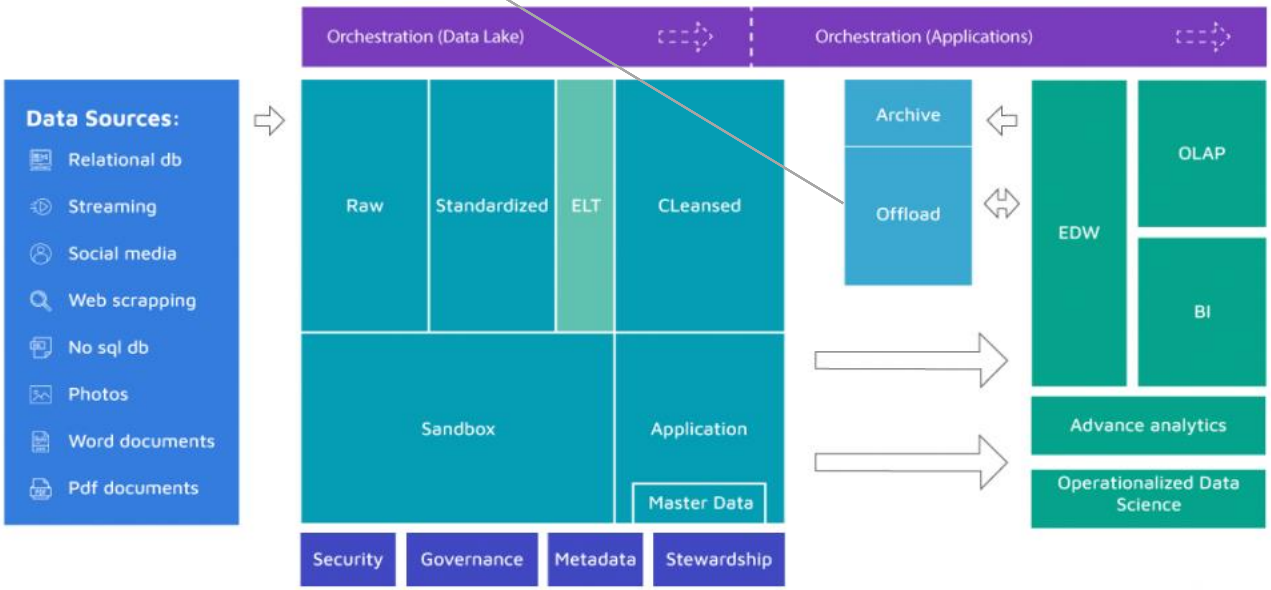
Data Lake Architecture

- Another layer that might be considered **optional**, is meant for advanced analysts’ and data scientists’ work
- Here they can carry out their **experiments when looking for patterns or correlations**.
- Whenever you have an idea to **enrich your data with any source from the Internet**, Sandbox is the proper place for this.



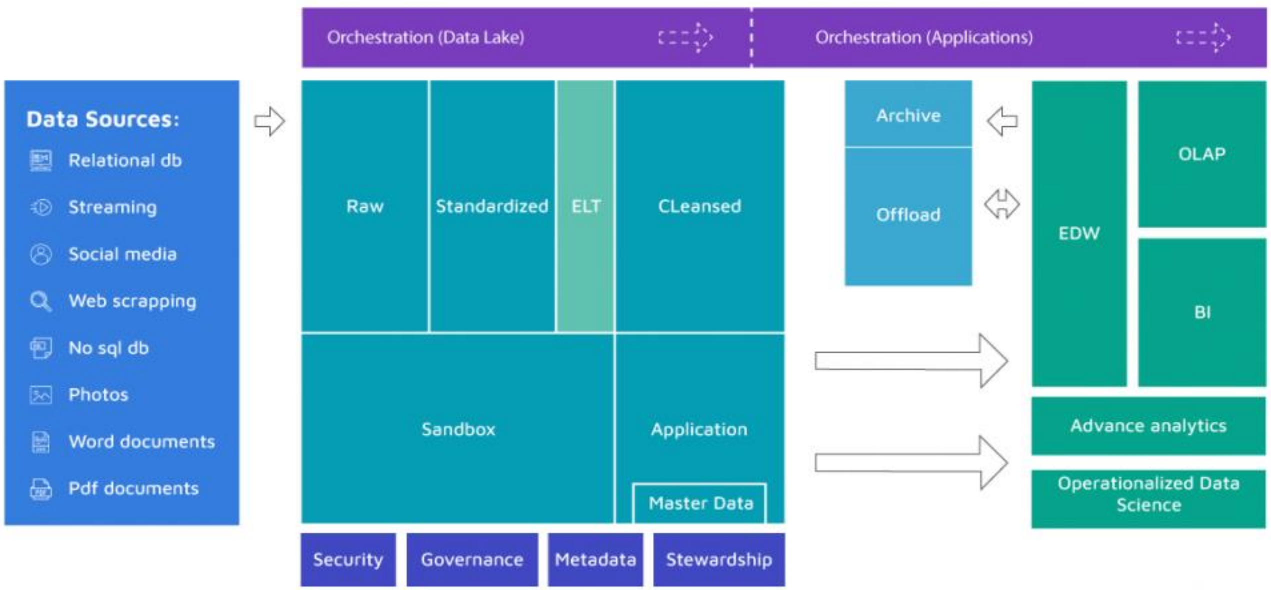
Data Lake Architecture

This area may be used to offload some time/resource consuming ETL processes to the Data Lake, which might be cheaper and faster.



Data Lake Architecture

- As data is being pushed from the Raw Layer, through the Cleansed to the Application and Sandbox layer, a tool to orchestrate the flow is needed.
- Most likely, companies will need to apply transformations. An orchestration tool capable will be needed.



Data Lake Architecture

Objectives

- ✓ Plan the structure based on optimal data retrieval
- ✓ Avoid a chaotic, unorganized data swamp

Common ways to organize the data:

Time Partitioning
Year/Month/Day/Hour/Minute

Subject Area

Security Boundaries
Department
Business unit
etc...

Downstream App/Purpose

Data Retention Policy
Temporary data
Permanent data
Applicable period (ex: project lifetime)
etc...

Business Impact / Criticality
High (HBI)
Medium (MBI)
Low (LBI)
etc...

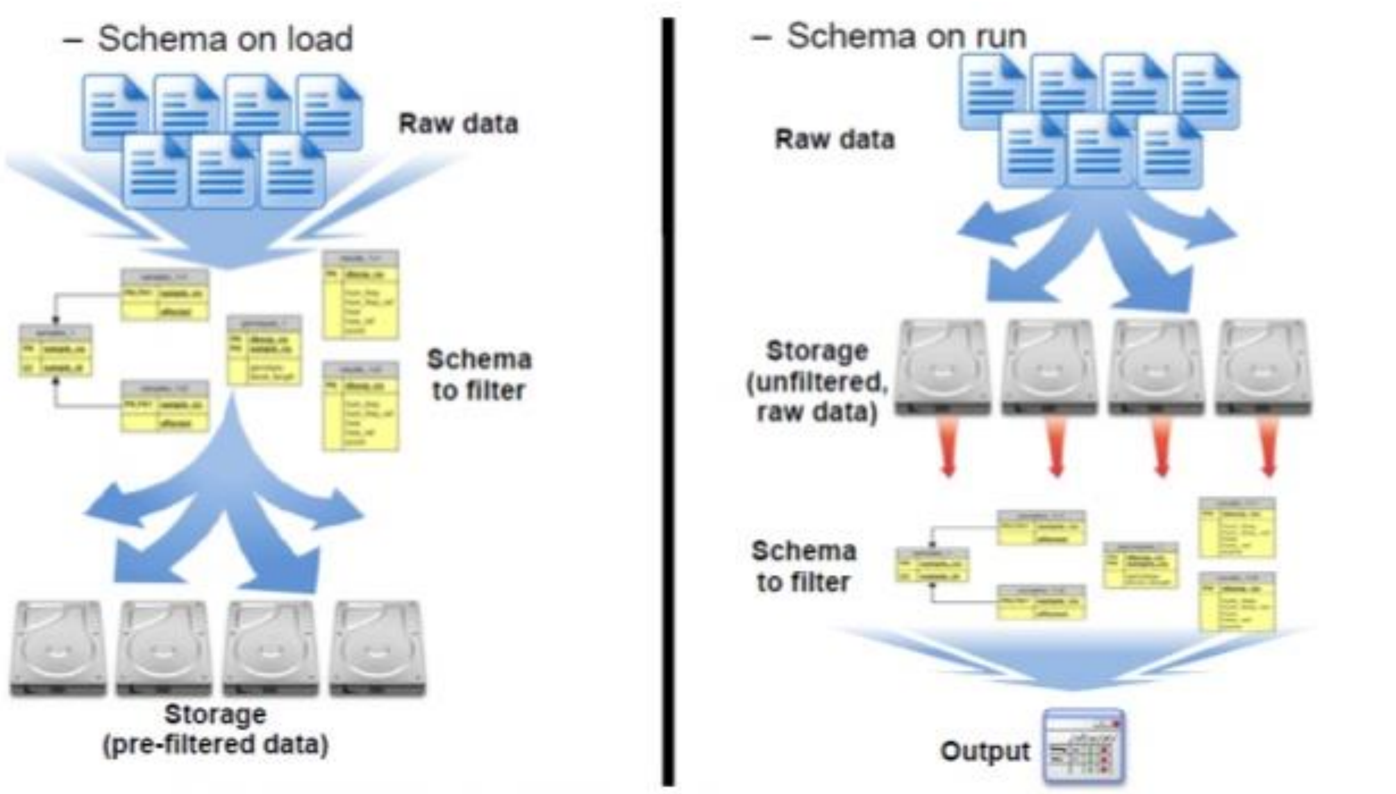
Owner / Steward / SME

Probability of Data Access
Recent/current data
Historical data
etc...

Confidential Classification
Public information
Internal use only
Supplier/partner confidential
Personally identifiable information (PII)
Sensitive – financial
Sensitive – intellectual property
etc...

Paradigm Shifts - More data being captured and leveraged

Paradigm Shifts – Reduce effort to leverage data

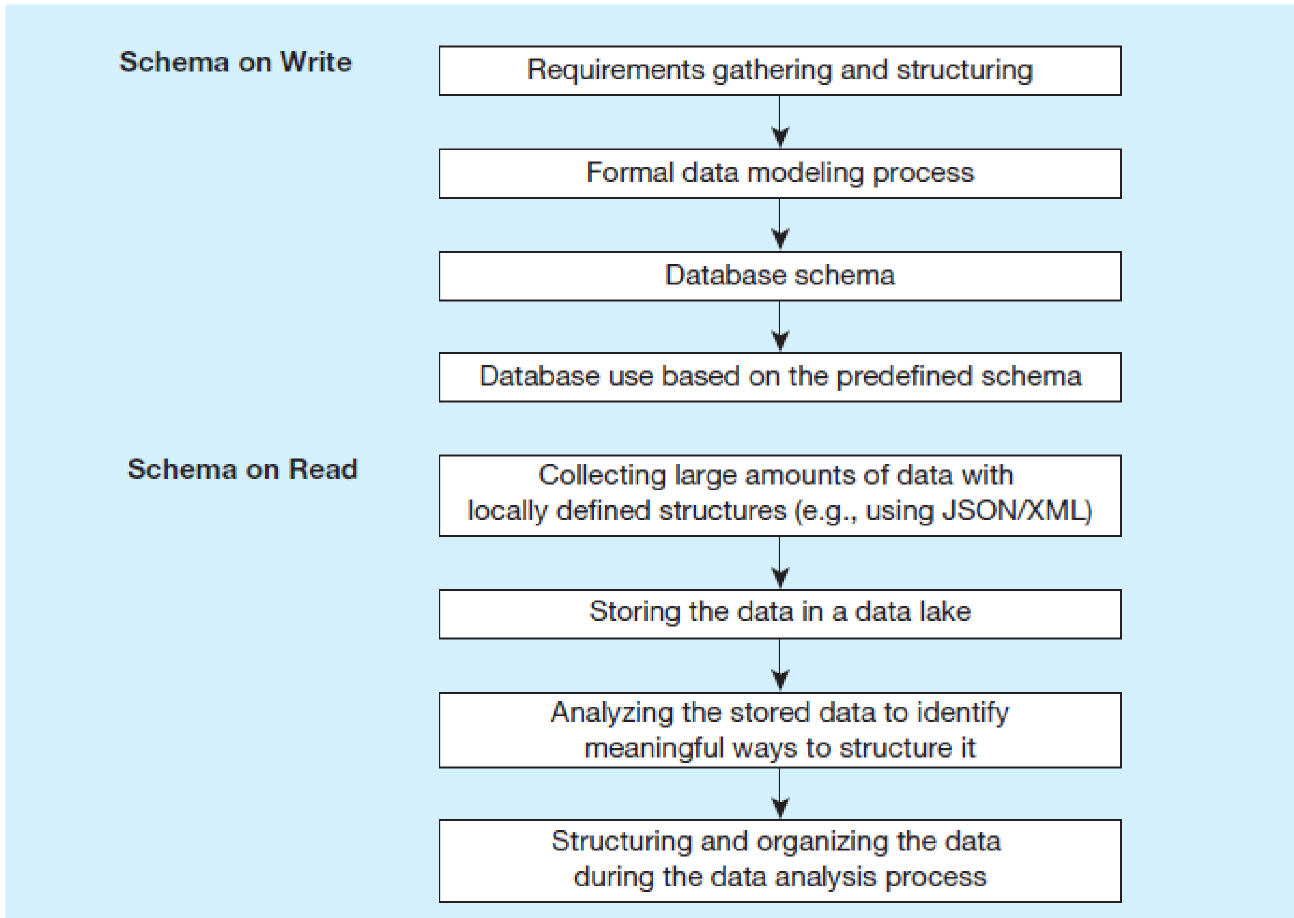


Big Data

- **Schema on Read, rather than Schema on Write**

Traditional datawarehouse design

The big data approach



Data Lake Architecture … not a Data Swamp

Objectives

✓ Plan the structure based on optimal data retrieval

✓ Avoid a chaotic, unorganized data swamp

Common ways to organize the data:

Time Partitioning
Year/Month/Day/Hour/Minute

Subject Area

Security Boundaries
Department
Business unit
etc…

Downstream App/Purpose

Data Retention Policy
Temporary data
Permanent data
Applicable period (ex: project lifetime)
etc…

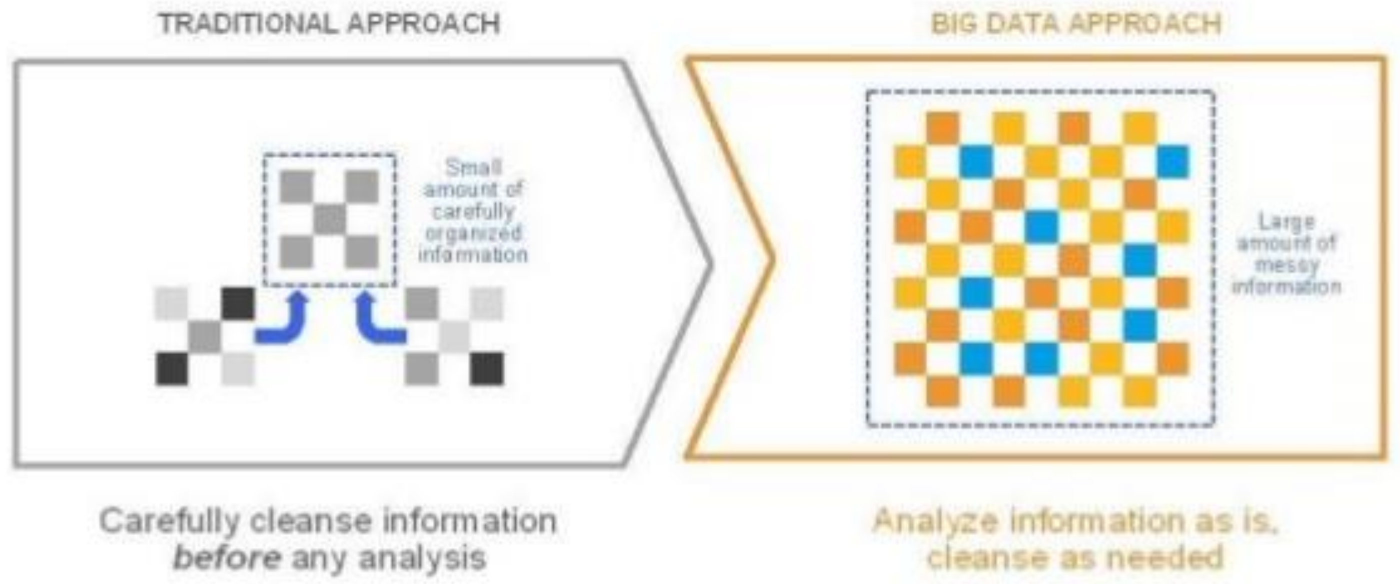
Business Impact / Criticality
High (HBI)
Medium (MBI)
Low (LBI)
etc…

Owner / Steward / SME

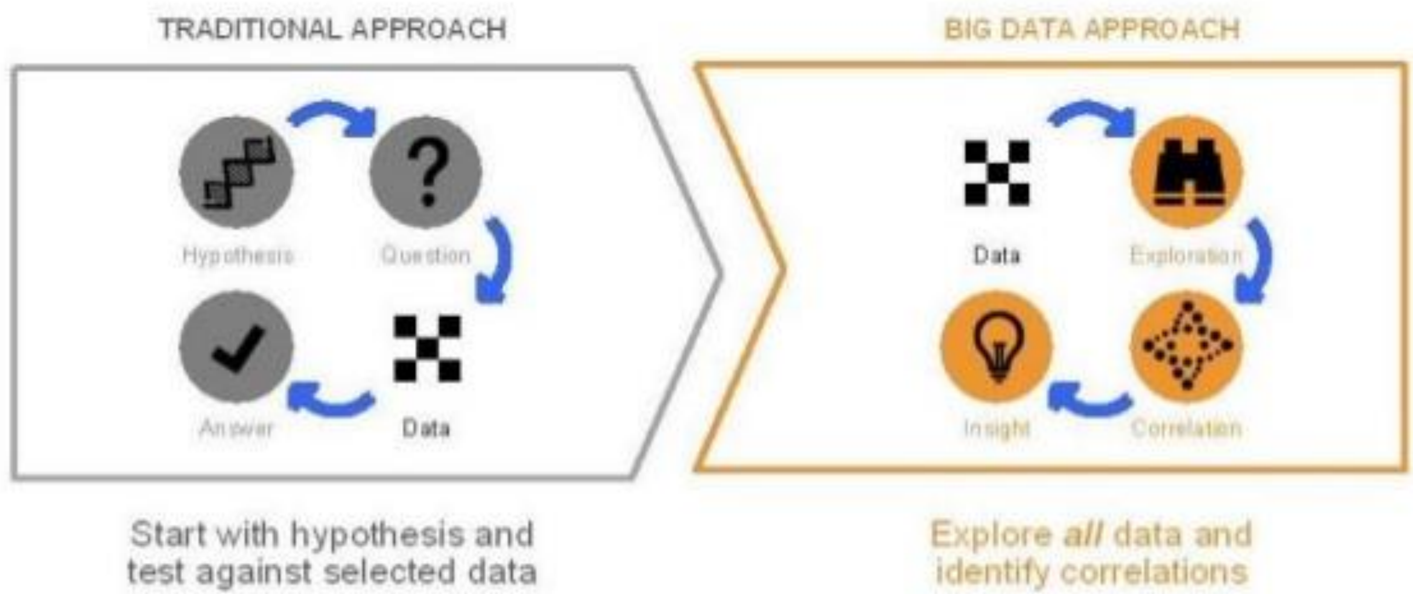
Probability of Data Access
Recent/current data
Historical data
etc…

Confidential Classification
Public information
Internal use only
Supplier/partner confidential
Personally identifiable information (PII)
Sensitive – financial
Sensitive – intellectual property
etc…

Paradigm Shifts – Reduce effort to leverage data

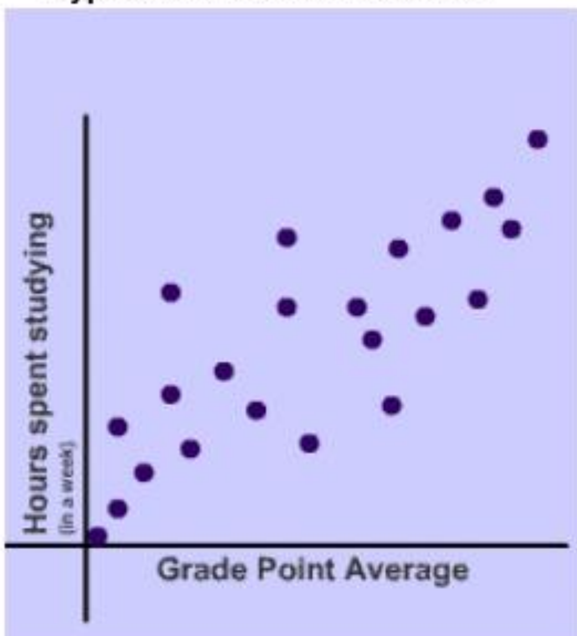


Paradigm Shifts – Data leads the way

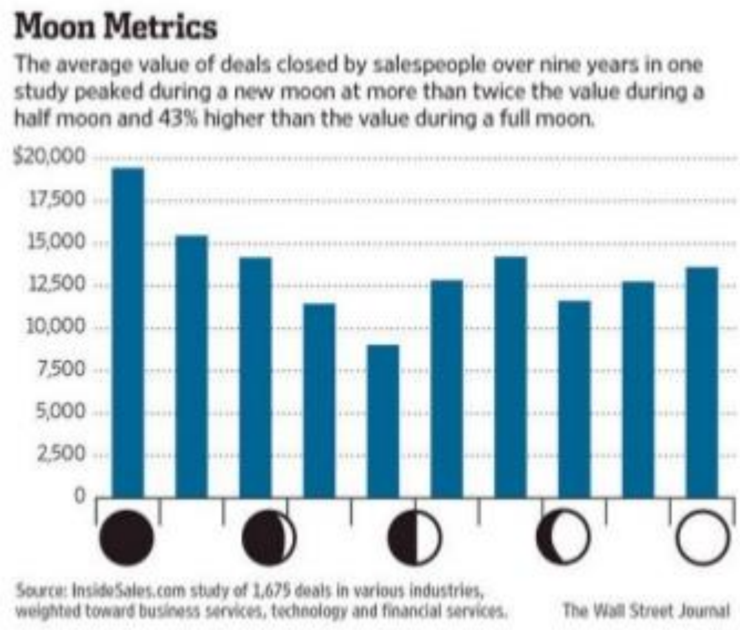


Paradigm Shifts – Data leads the way

Hypothesis based correlation

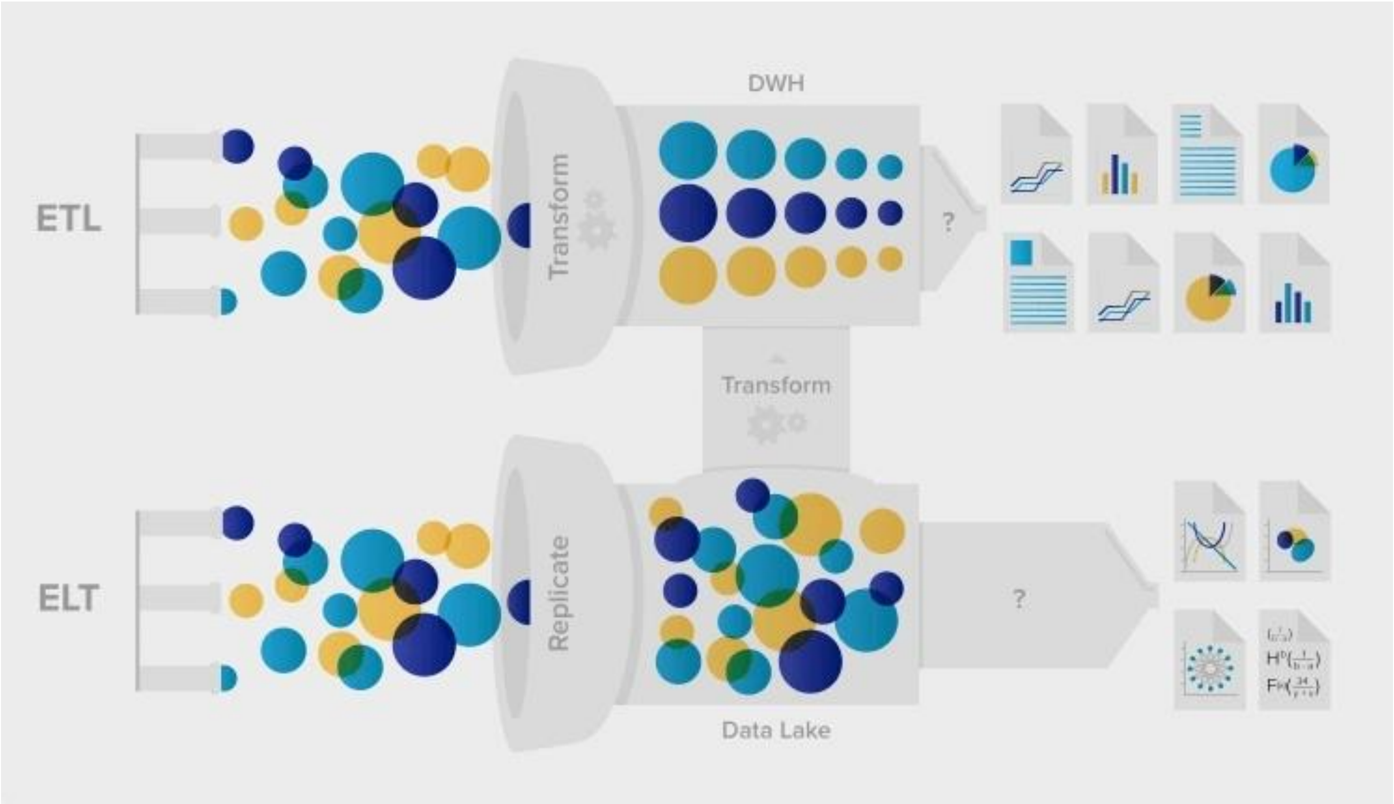


Weird correlation



Paradigm Shifts – Leverage data as it is captured

Paradigm Shifts – Leverage data as it is captured



Data Warehouse

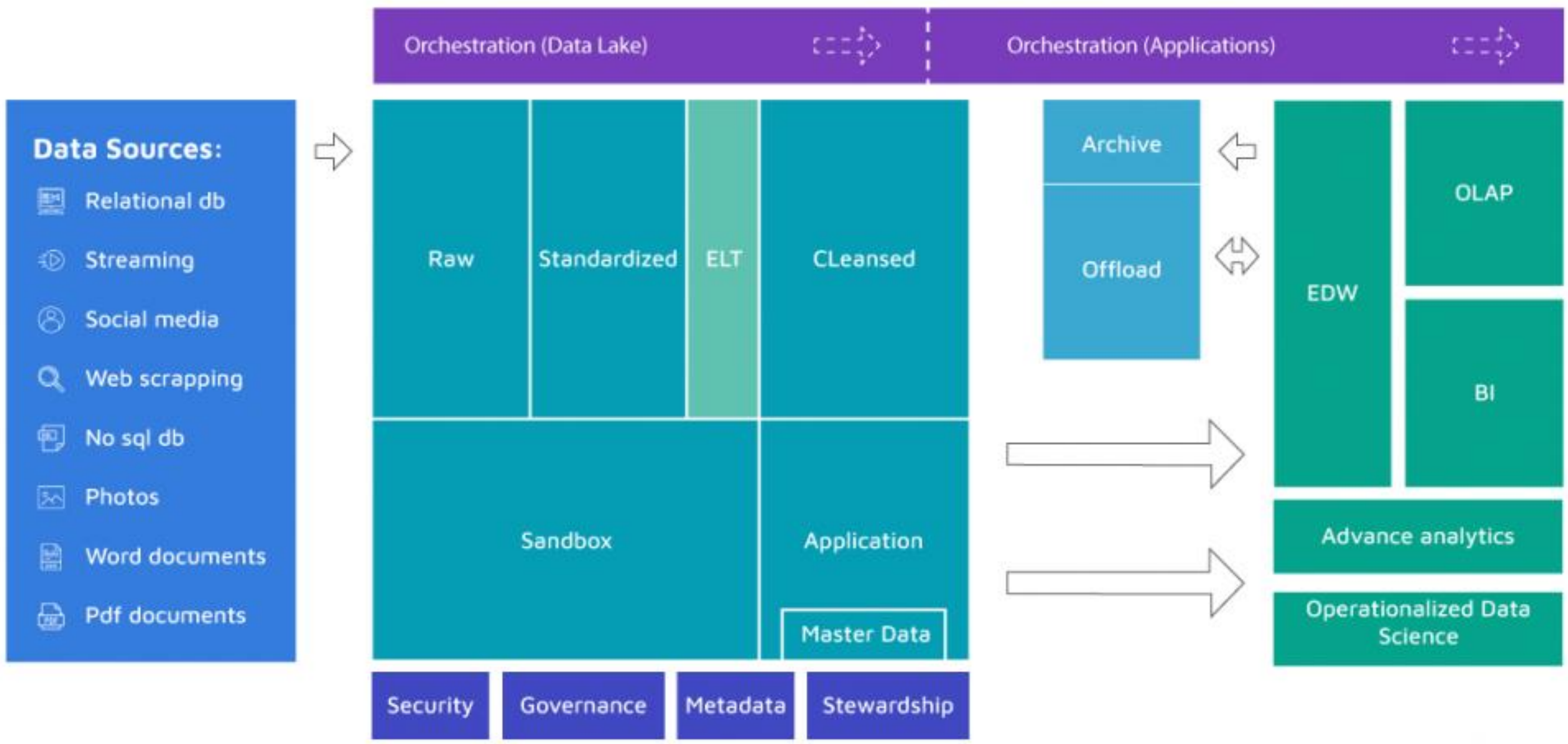
1. Report Data Structuring
2. Ingest Data
3. Analyze

Data Lake

1. Ingest Data
2. Analyze
3. Define Data Structure

Big Data

Data Lake Architecture … Data Warehouse is still there!

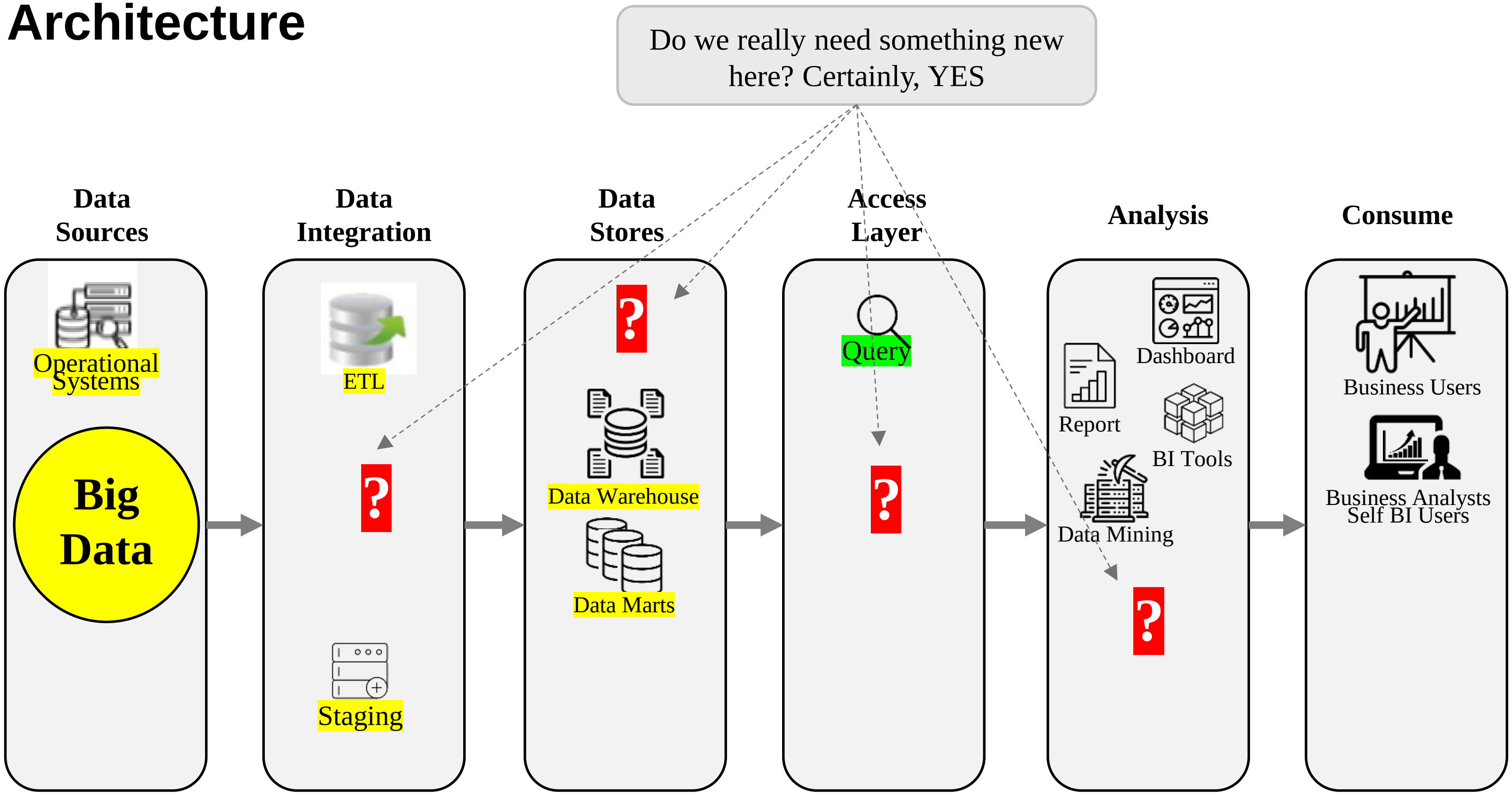


Big Data

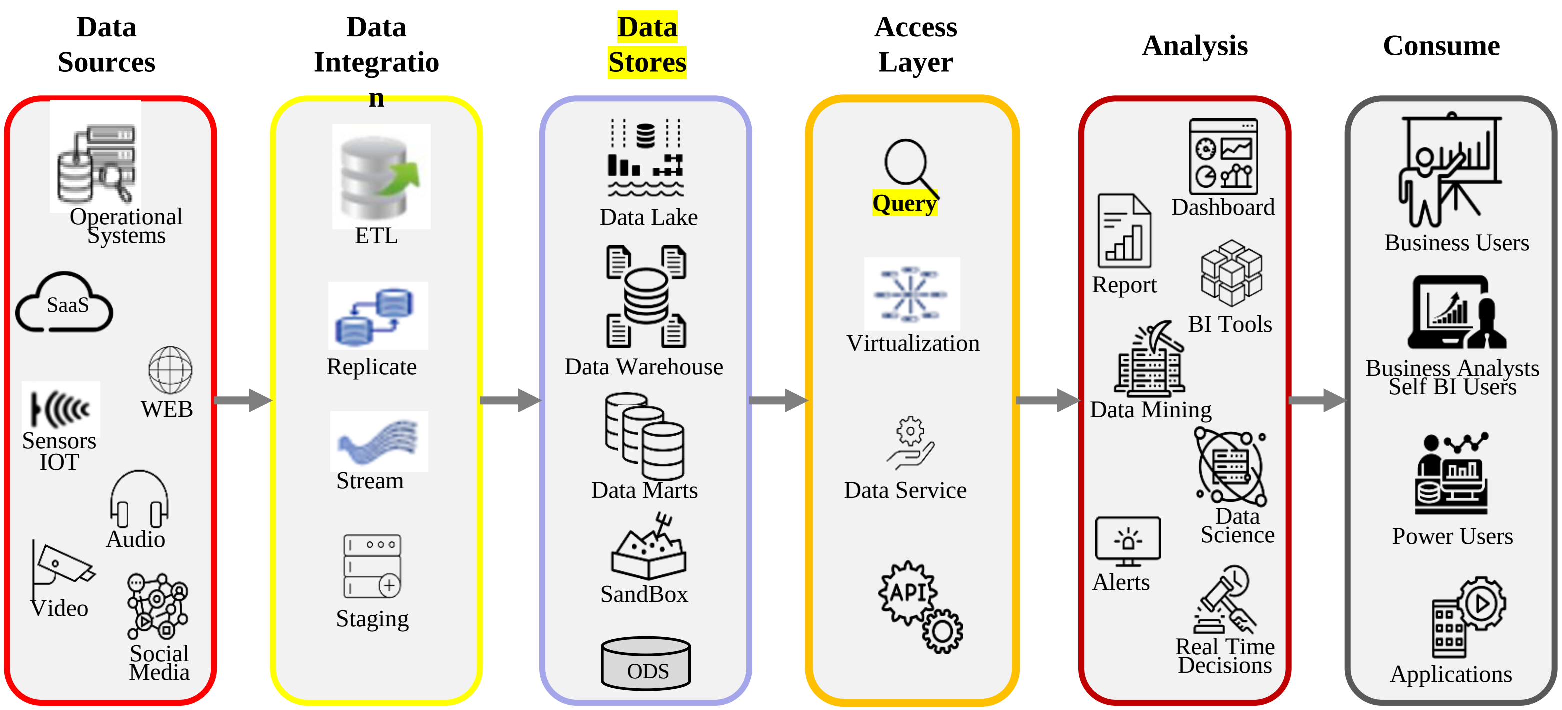
AREA	DATA LAKE	DATA WAREHOUSE
Data Store	It can capture and retain unstructured, semi-structured, and structured data in its raw format. A Data Lake stores all types of data, irrespective of the source and structure.	It can capture and retain only structured data. A Data Warehouse stores data in quantitative metrics with their attributes. Data is transformed and cleansed.
Schema Definition	Typically, the schema is defined after data is stored. This offers high agility and data capture quite easily, but it requires work at the end of the process (schema-on-read).	Typically, a schema is defined prior to when data is stored. It requires work at the start of the process, but it offers performance, security, and integration (schema-on-write).
Data Quality	Any data that may or may not be curated (such a raw data).	Highly curated data that serves as the central version of the truth.
Users	A Data Lake is ideal for the users who indulge in deep analysis, like Data Scientists, Data Engineers, and Data Analysts.	A Data Warehouse is ideal for operational users like Business Analysts because of being well structured and easy to use and understand.
Price & Performance	The storage cost is relatively low, compared to a Data Warehouse, and querying results is better.	The storage cost is high, and querying results is time consuming.
Accessibility	A Data Lake has few constraints and is easily accessible. Data can be changed and updated quickly.	A Data Warehouse is structured by design, which makes it difficult to access and manipulate.

Big Data

Bi Architecture



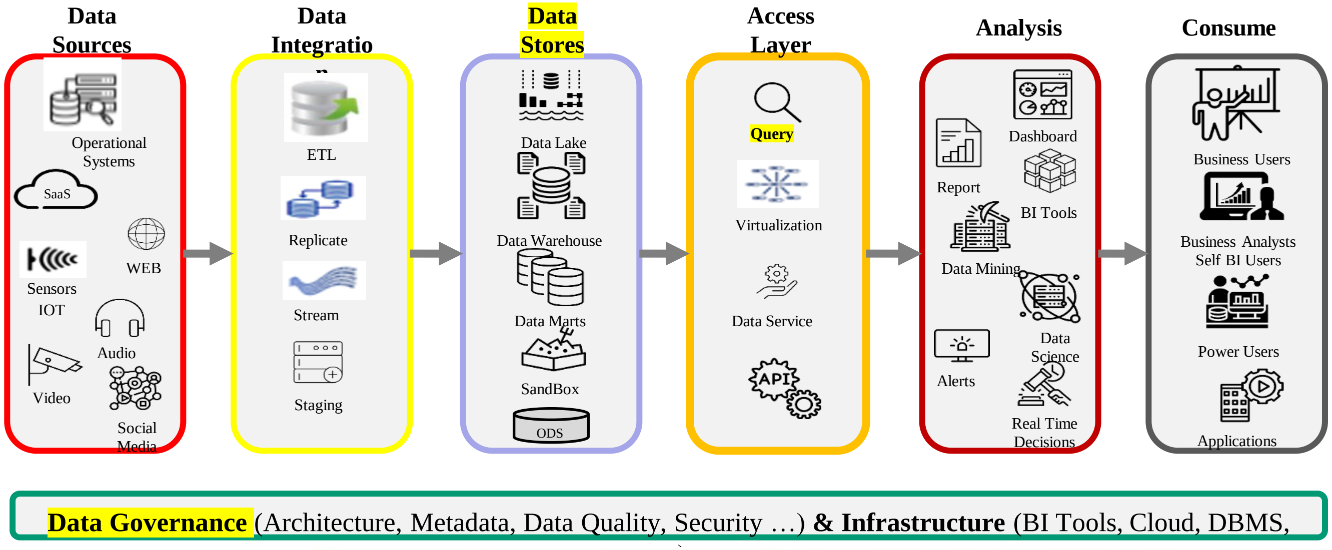
Big Data

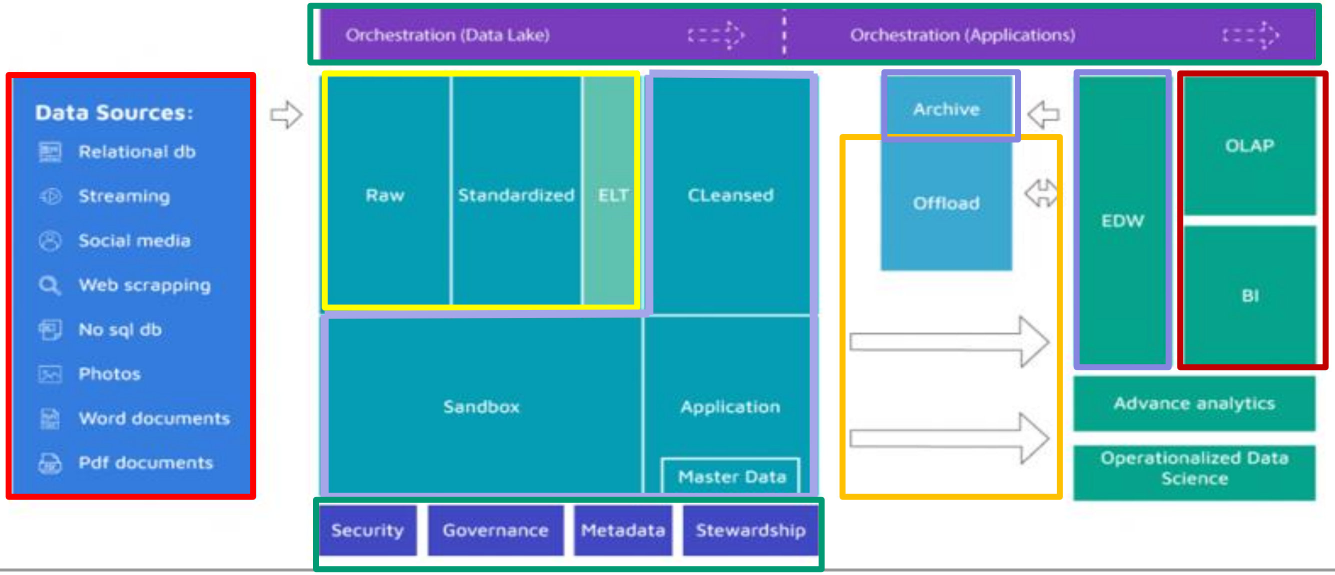


Data Governance (Architecture, Metadata, Data Quality, Security …) **& Infrastructure** (BI Tools, Cloud, DBMS, …)

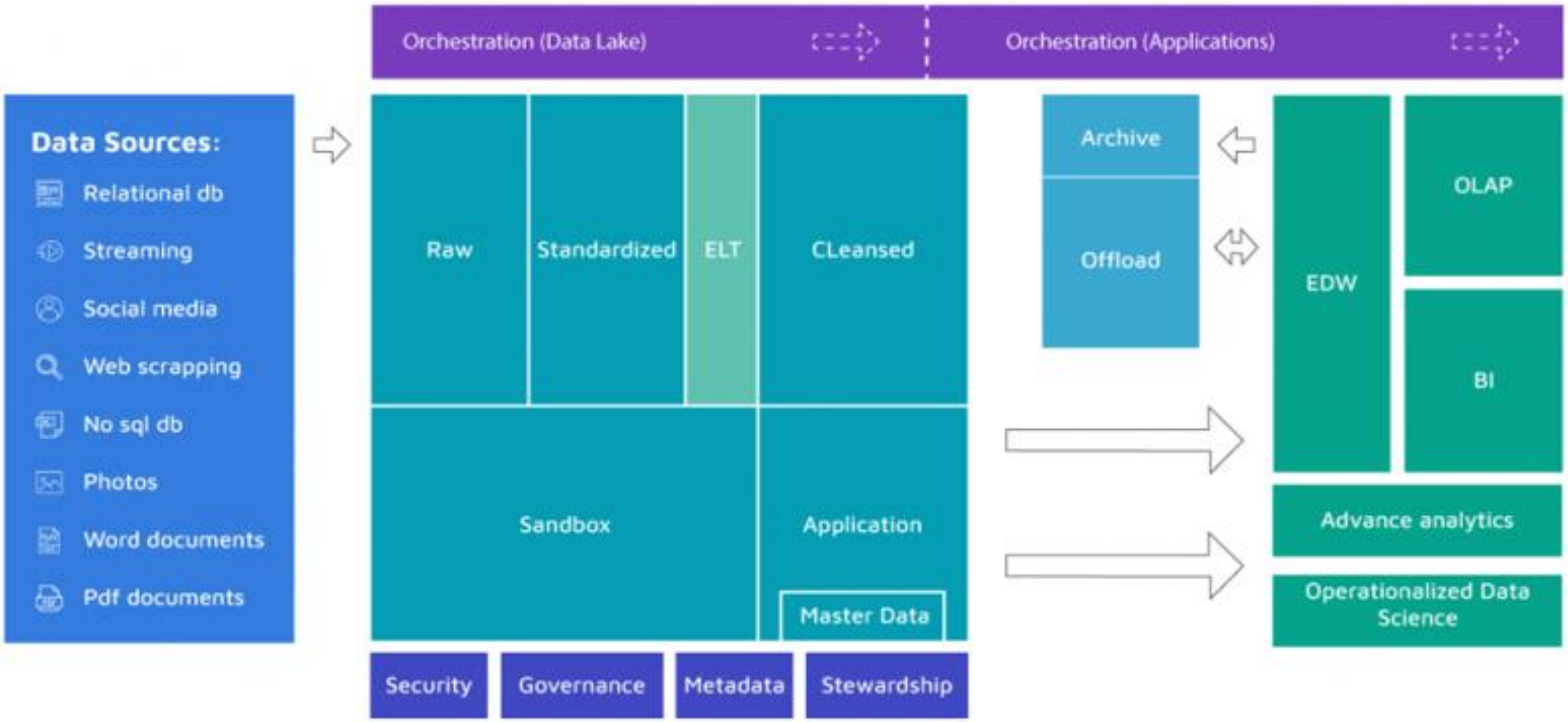
Big Data

BI Architecture vs Data Lake





Data Lake Tools

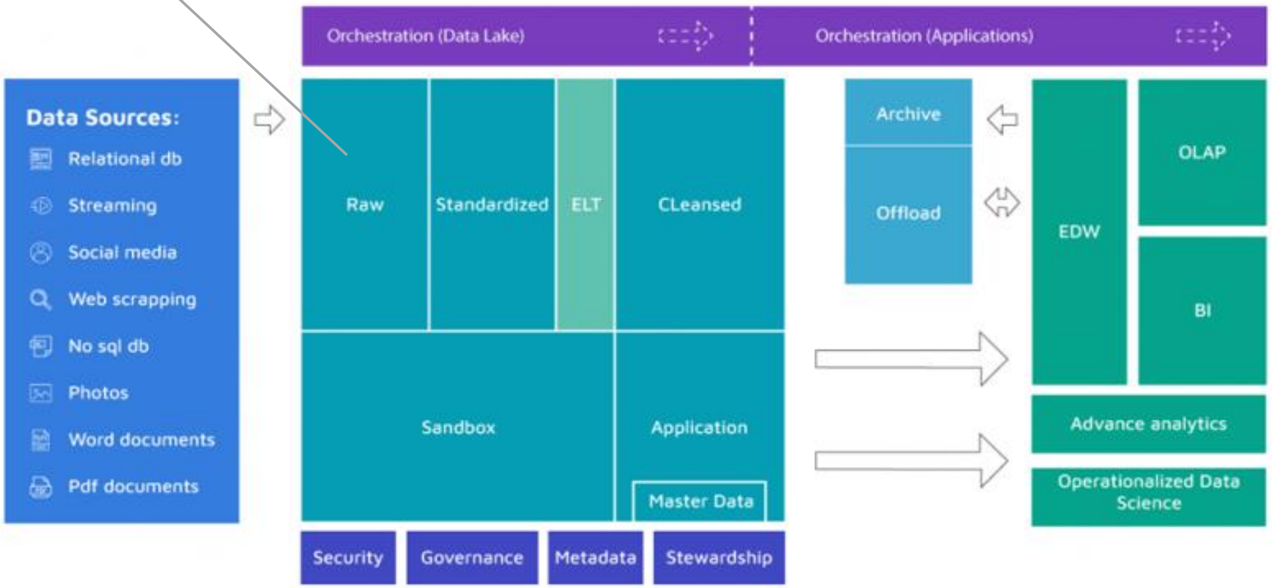


Data Lake Tools

Raw Storage

Raw storage is the basis of a data lake. It must support large volumes of data in a variety of formats (structured, semi-structured and unstructured).

- Amazon S3 (Simple Storage Service): One of the most popular services for data lake raw storage due to its scalability, cost-effectiveness and integration with other tools.
- Azure Data Lake Storage (ADLS): Designed specifically for data lakes in the Azure ecosystem.
- Google Cloud Storage: A scalable and highly durable service for storing data in Google Cloud data lakes.
- HDFS (Hadoop Distributed File System): Traditionally used in on-premise data lake implementations.
- MinIO: An open-source solution compatible with S3, ideal for private implementations.

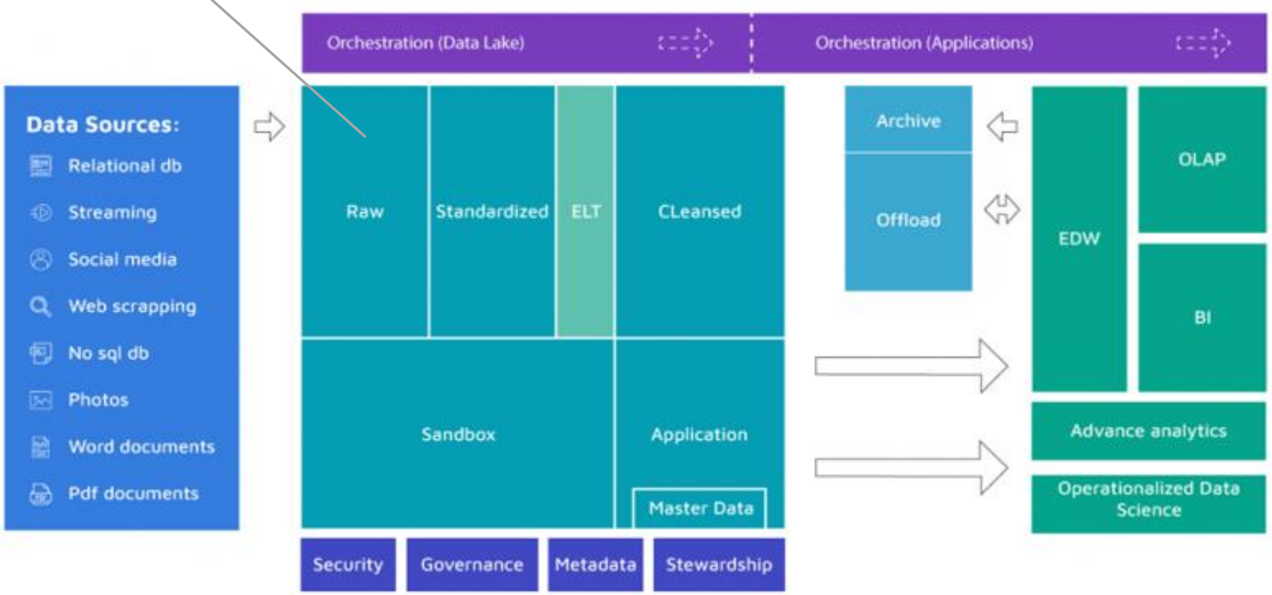


Data Lake Tools

Data Processing Tools

Tools for processing large-scale raw data and transforming it into useful formats for analysis.

- Apache Spark: A powerful framework for distributed and scalable data processing.
- Apache Hadoop: Offers an ecosystem for batch processing using MapReduce.
- Apache Flink: Ideal for real-time data processing.
- Databricks: Unified platform based on Apache Spark for data processing and machine learning.
- Google Dataflow: A managed service for real-time and batch stream processing.
- AWS Glue: A serverless service for extraction and transformation processing in the AWS ecosystem.



Data Lake Tools

Data Governance and Catalogue

Tools for cataloguing, organising and applying security and governance policies to data.

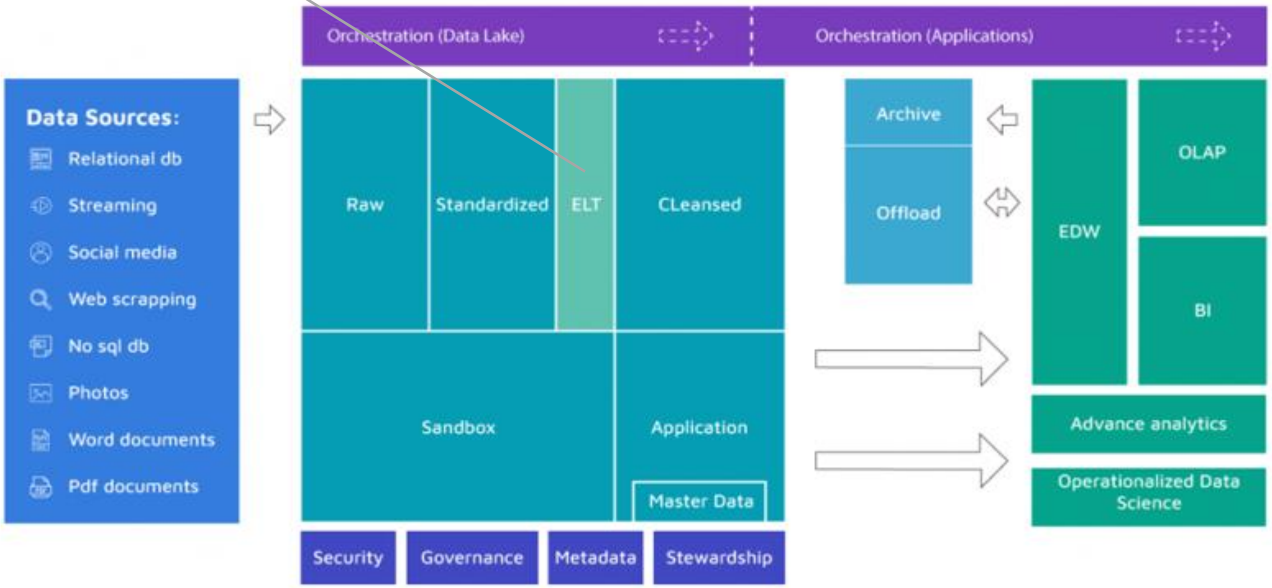
- Apache Atlas: An open-source framework for data governance and metadata management.
- AWS Lake Formation:A tool for creating, managing and protecting data lakes on AWS.
- Azure Purview: A data governance service for organising and cataloguing data in Azure.
- Collibra: Commercial tool for data governance and cataloguing.
- Alation: Popular for data cataloguing and discovery.

Data Lake Tools

Data Integration (ETL/ELT)

Tools for extracting, transforming and loading data into the data lake.

- Apache Nifi: Designed for data flow automation and real-time integration.
- Talend: Robust platform for data integration and ETL.
- Informatica: Traditional tool for ETL in business environments.
- Fivetran: Ideal for automated data synchronisation between systems.
- Google Cloud Data Fusion: For data integration based on the Google Cloud ecosystem.



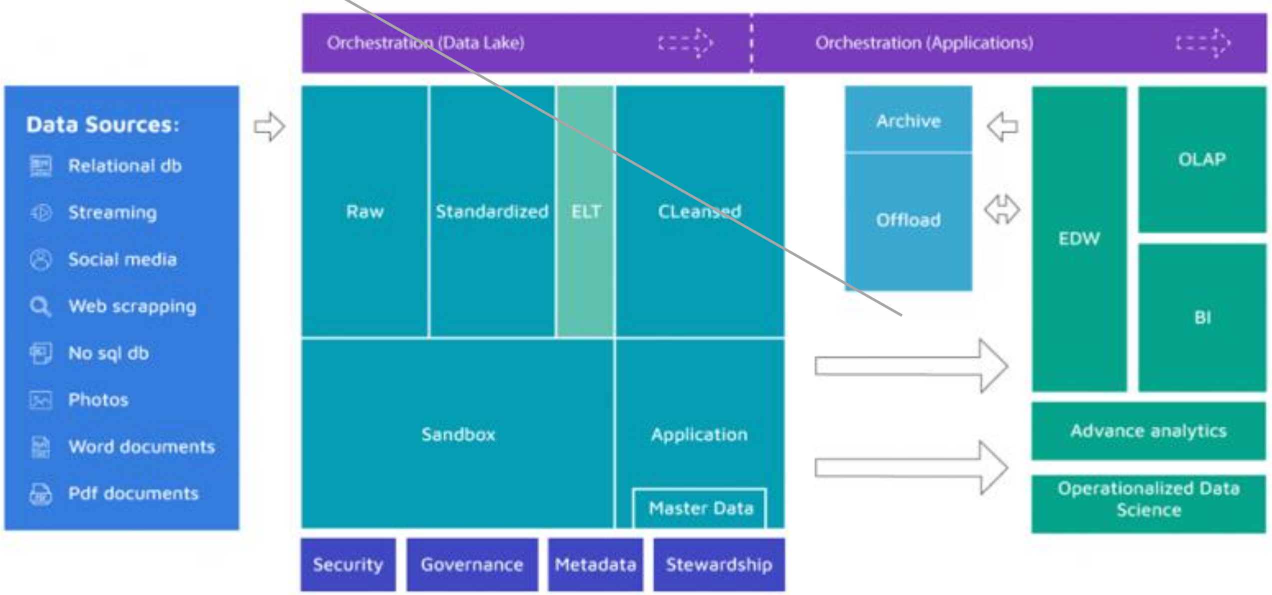
Big Data

Data Lake Tools

Query and exploration tools

Technologies for querying and exploring data directly in the data lake.

- Presto/Trino: Distributed query engines, optimised for SQL queries directly on the data lake storage.
- AWS Athena: Serverless service that allows SQL queries to be executed directly on Amazon S3.
- Google BigQuery: An integrated data warehouse that also works as an interface to data lakes.
- Azure Synapse Analytics: Unified analysis platform that includes support for data lakes.

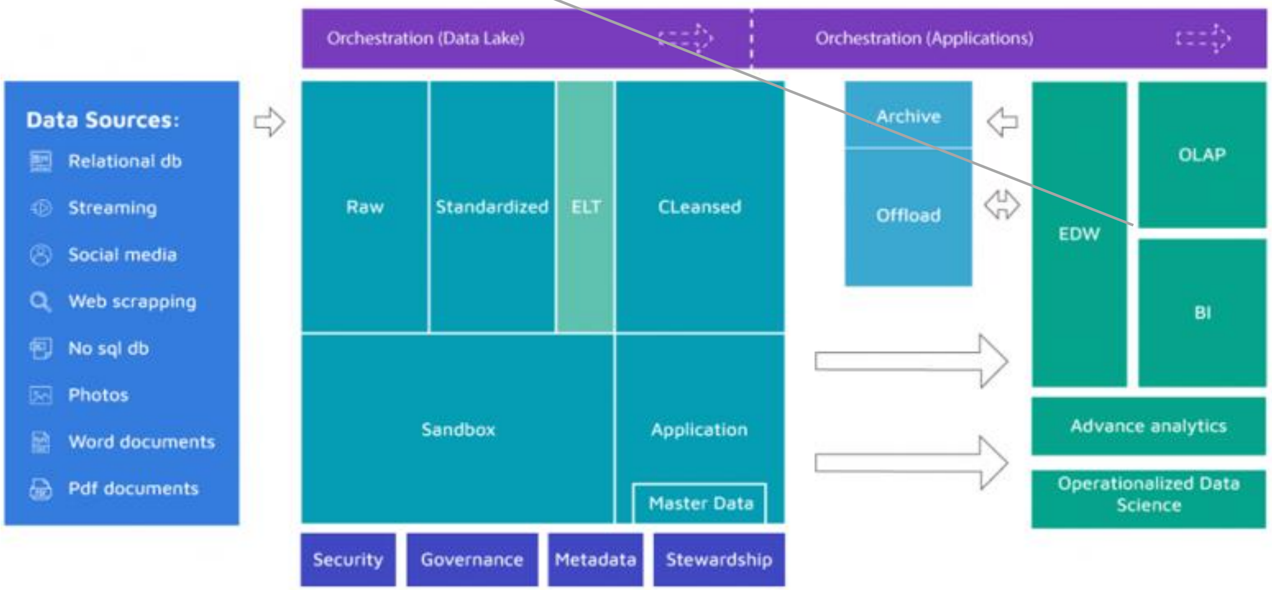


Data Lake Tools

Data Analysis and Visualisation

Tools for analysing and presenting data stored in the data lake.

- Tableau: Popular for creating dashboards and interactive visualisations.
- Power BI: Microsoft tool widely used for data visualisation.
- Apache Superset: An open-source alternative for data visualisation.
- Qlik Sense: Offers integration with data lakes for analysing data.

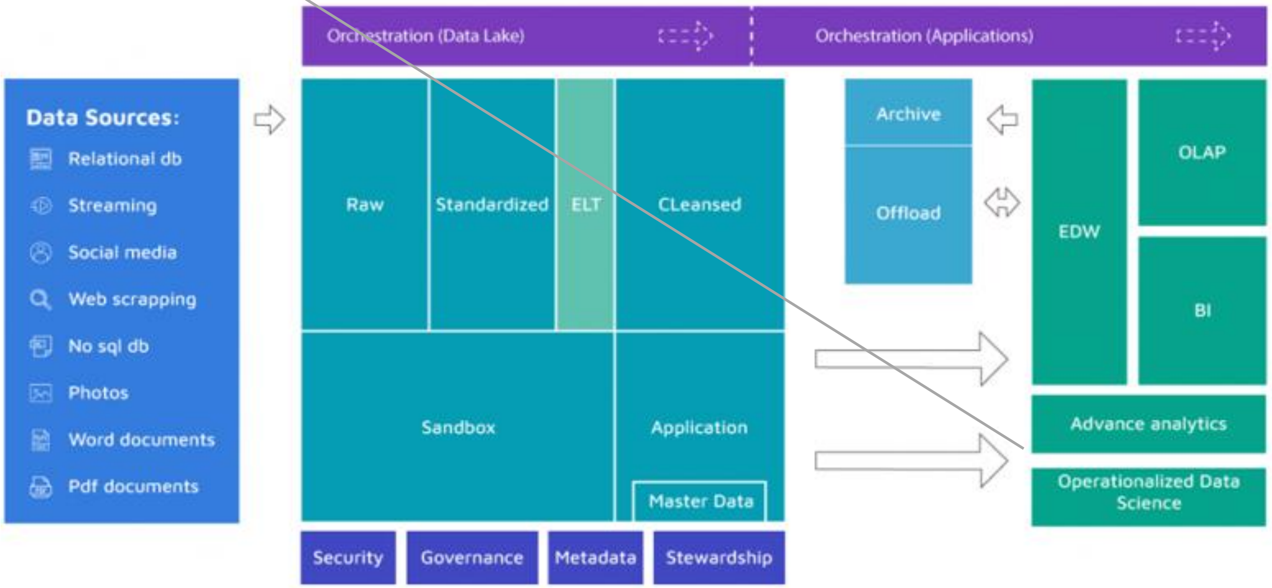


Data Lake Tools

Machine Learning and Artificial Intelligence

Platforms for creating and training machine learning models using data in the data lake.

- Apache Mahout: Framework for distributed machine learning.
- TensorFlow and PyTorch: Widely used libraries for building and training AI models.
- AWS SageMaker: Managed platform for building, training and deploying machine learning models.
- Azure Machine Learning: Managed service for machine learning in the Azure ecosystem.
- Google AI Platform: For building and deploying AI models in Google Cloud.

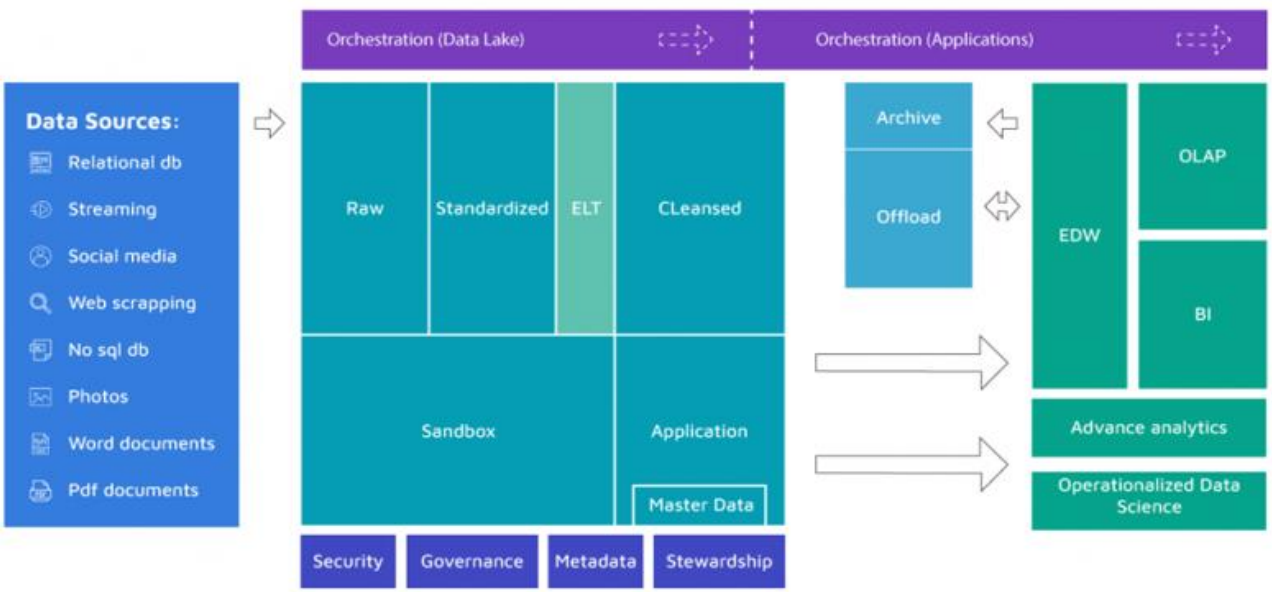


Data Lake Tools

Orchestration

Tools for managing workflows and data pipelines in the data lake.

- Apache Airflow: Open-source tool for orchestrating data pipelines.
- Luigi: Another open-source tool for orchestrating tasks.
- Prefect: A modern solution for orchestrating workflows.
- AWS Step Functions: For orchestrating services and pipelines in the AWS ecosystem.



Big Data

- **Relational Database ACID characteristics**

  Atomicity, Consistency, Isolation, Durability

vs

- **No SQL – Not Only SQL BASE characteristics**

  Basically Available, Soft state, Eventually consistent

- **Relational Database ACID characteristics**

Atomicity	Assure that a transaction COMPLETELY succeeds or COMPLETELY fails (power failures, errors, crashes, …)
Consistency	Transaction won't put the database into an invalid state (constraints, cascades, integrity, triggers, …)
Isolation	When multiple transactions occur at the same time, they will force the database into the same state as if they had been run one at a time (locks, …)
Durability	Once a transaction has been committed, the database will hold that state even in the event of an outside event such as a power loss or error (volatile memory, …)

The **ACID** properties, in totality, provide a mechanism to ensure the correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations, and updates that it makes are durably stored.

Big Data

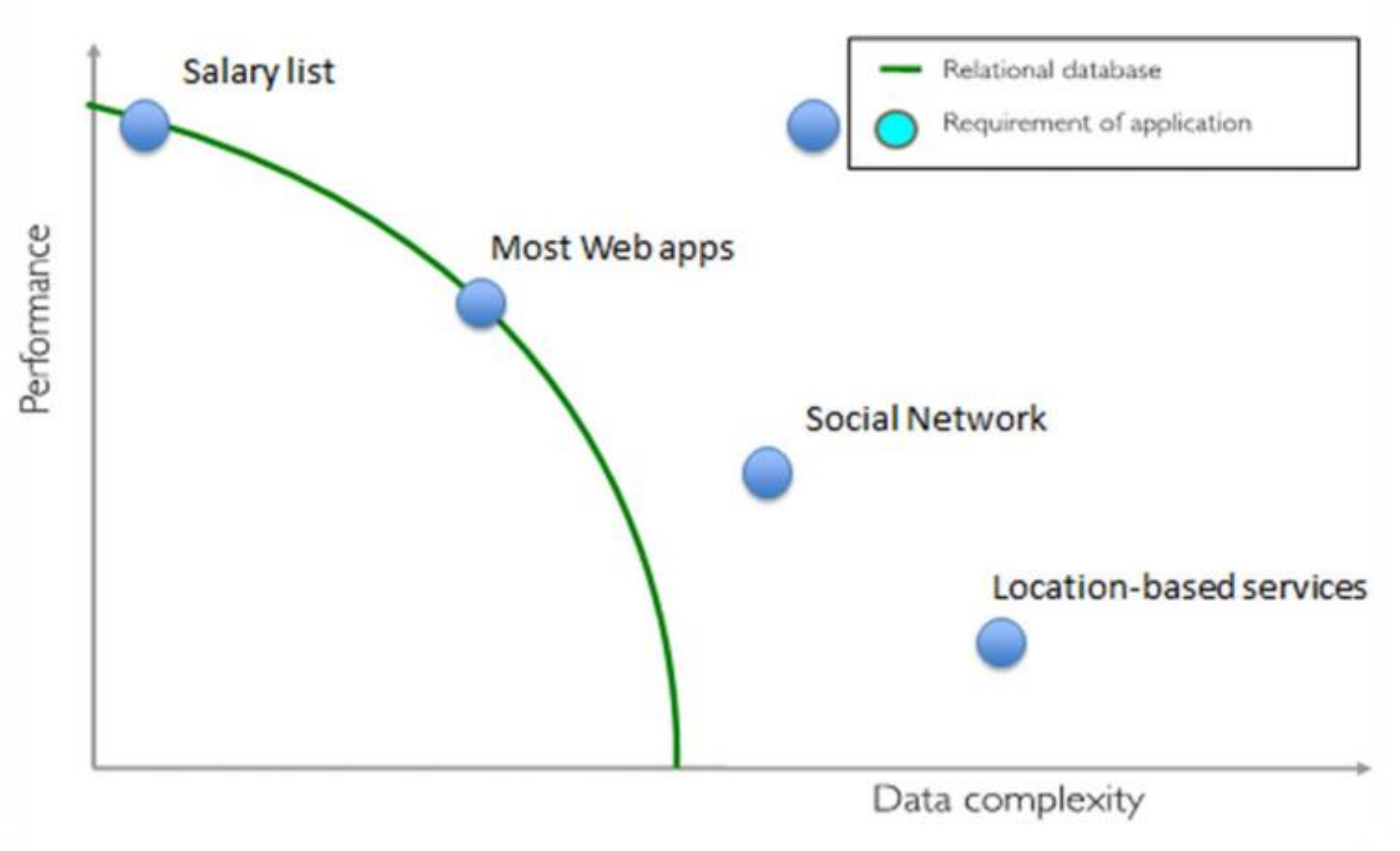
- **Benefits of Relational Databases**
  - Designed for all purposes
  - ACID
  - Strong consistancy, concurrency, recovery
  - Mathematical background
  - Standard Query language (SQL)
  - Lots of tools to use with i.e: Reporting services, entity frameworks, ...

Big Data

- **Benefits of Relational Databases**
  - Designed for all purposes
  - ACID
  - Strong consistancy, concurrency, recovery
  - Mathematical background
  - Standard Query language (SQL)
  - Lots of tools to use with i.e: Reporting services, entity frameworks, ...

Big Data

- **RDBMS Performance is not good enough for all purposes …**

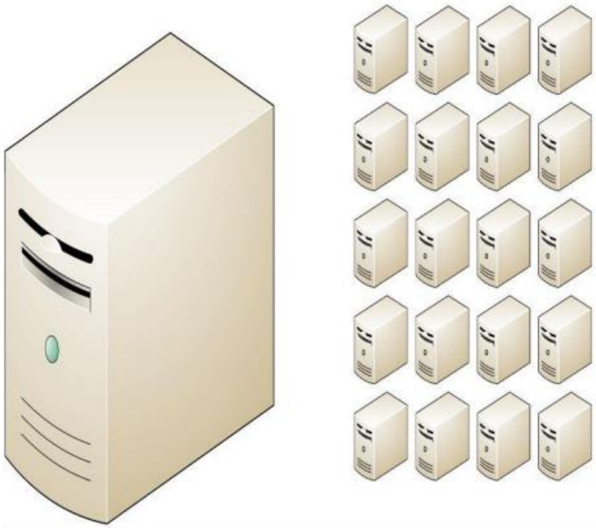


Big Data

- **… and**

  - ❑ Relational databases were not built for **distributed applications.**

Era of Distributed Computing



Because…

- ❑ Joins are expensive
- ❑ Hard to scale horizontally
- ❑ Expensive (product cost, hardware, Maintenance)

And….

It's weak in:

- ❑ Speed (performance)
- ❑ High availability
- ❑ Partition tolerance

Big Data

- **Big Data Technologies used to implement Data Lakes**

  - **<u>NoSQL Big Data systems</u>**

    ✓ designed to take advantage of new cloud computing architectures, avoiding the complex schemas of RDBMS.

  - **<u>Massively Parallel Processing (MPP) database systems and MapReduce</u>**

    ✓ provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

No SQL – Not Only SQL

- A category of “recently” introduced data storage and retrieval technologies not based on the relational model and **employing less constrained consistency**

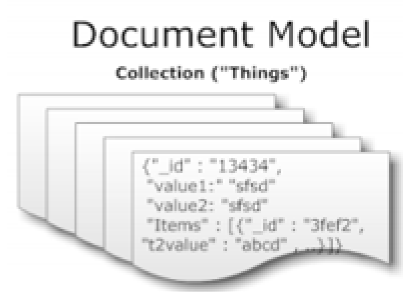
- **NoSQL** doesn’t really mean that there isn’t SQL available but rather the backend database **doesn’t follow the relational model**.

- *No-SQL databases* refer to high-performance, non-relational data stores. They excel in their **ease-of-use, scalability, resilience, and availability characteristics**.

- Instead of joining tables of normalized data, **NoSQL stores unstructured or semi-structured data**, (often in key-value pairs or JSON - JavaScript Object Notation - documents).

No SQL – Not Only SQL Advantages

- Schema less / Flexible
- Can handle structured, semi-structured, and unstructured data with equal effect
- Supports schema on read, avoiding of up-front schema design
- Allows fast development / easy to use
- Horizontal scaling
- Largely open source
- Not ACID compliant!
- **BASE** – Basically Available, Soft state, Eventually consistent
- No single point of failure
- Most of considerations are done in application layer
- Gather all items in an aggregate (ex: document) / Fast queries



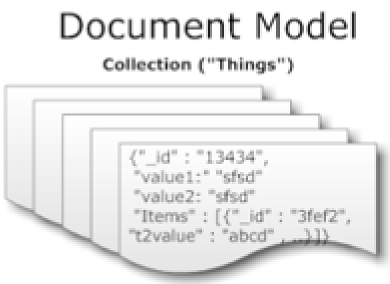
- **NoSQL BASE characteristics**

Basic Avalability	Focused on the availability of data even in the presence of multiple failures. Achieves this by using a highly distributed approach to database management.
Soft state	BASE databases abandon the consistency requirements of the ACID model pretty much completely. One of the basic concepts behind BASE is that data consistency is the developer's problem and should not be handled by the database.
Eventual consistency	The only requirement that NoSQL databases have regarding consistency is to require that at some point in the future, data will converge to a consistent state, although no guarantees are made. ACID requirements that prohibits a transaction from executing until the prior transaction has completed, keeping a consistent state, are completely abandoned in BASE.

BASE provides less assurance than ACID, but it scales very well and reacts well to rapid data changes.

No SQL – There are disadvantages as well

- No standardization rules
- More limited query capabilities
- RDBMS databases and tools are comparatively more mature and accepted
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it will become more difficult to maintain unique values as keys
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.



No SQL – Aggregate Data Models

NoSQL databases are classified in four major datamodels:

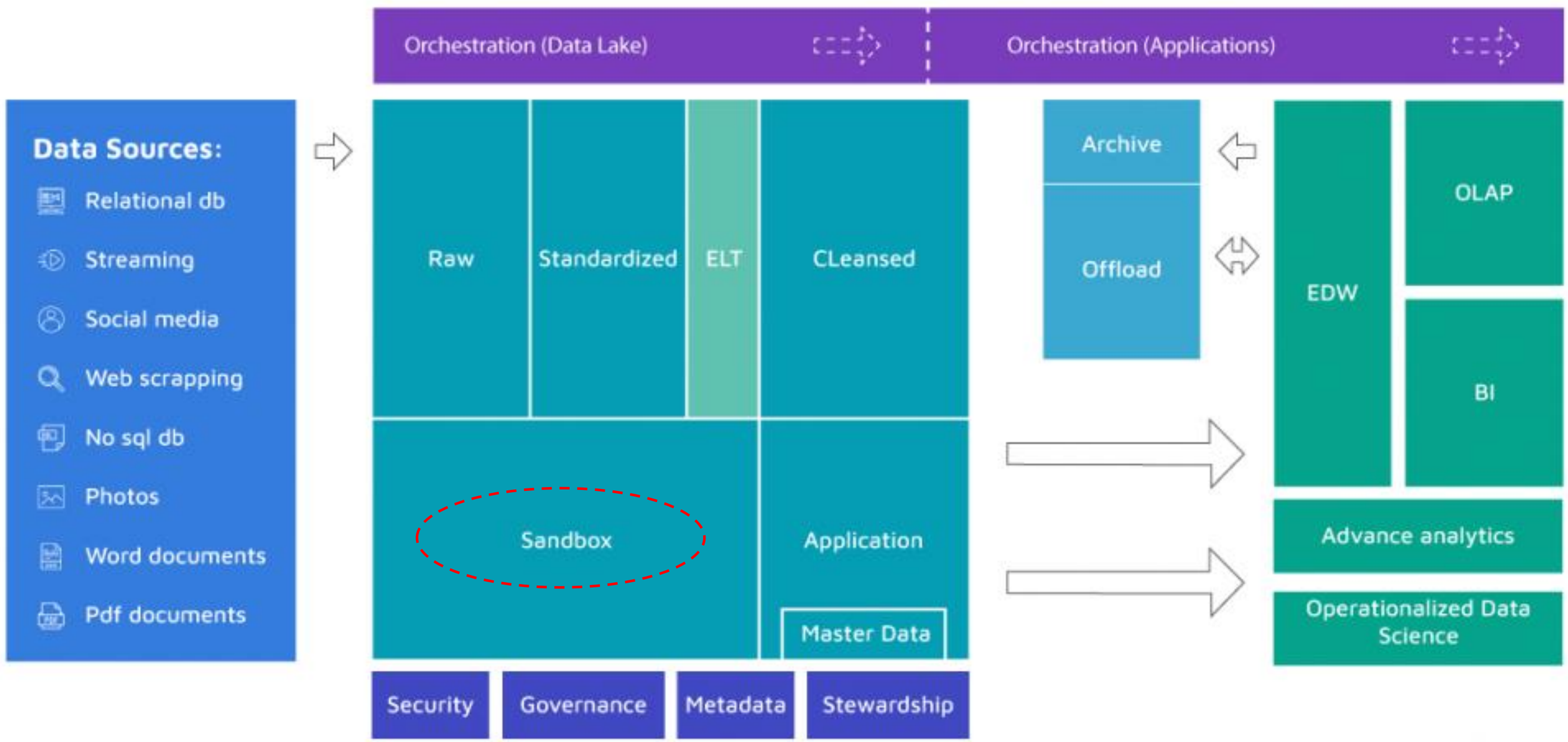
- Key-value
- Document
- Column family
- Graph

Each DB has its own query language

Big Data

Consider a NoSQL datastore when:	Consider a relational database when:
You have high volume workloads that require predictable latency at large scale (e.g. latency measured in milliseconds while performing millions of "*transactions*" per second)	Your workload volume generally fits within thousands of transactions per second
Your data is dynamic and frequently changes	Your data is highly structured and requires referential integrity
Relationships can be de-normalized data models	Relationships are expressed through table joins on normalized data models
Data retrieval is simple and expressed without table joins	You work with complex queries and reports
Data is typically replicated across geographies and requires finer control over consistency, availablity, and performance	Data is typically centralized, or can be replicated regions asynchronously
Your application will be deployed to commodity hardware, such as with public clouds	Your application will be deployed to large, high-end hardware

Data Lake Architecture



What do you think when you hear “Sandbox”?

Big Data

What do you think when you hear “Sandbox”?

What do you think when you hear “Sandbox”?

BI Sandbox - the challenge

- Often, the business has not had an **opportunity to work with selected data**, so can't clearly define metrics and BI reports properly.
- Another issue is the difficulty of **integrating external data** with data in an existing data mart. In short, the traditional approach doesn't work for these cases, nor does it work for one-off exploratory initiatives due to the long development time and cost involved for what is essentially "throw away."

BI Sandbox - purpose:

A data sandbox is primarily explored by data science teams that obtain sandbox platforms from stand-alone or external data, data marts, logical partitions in enterprise data warehouses, or selected partitions of data lakes.

Objectives:

- Facilitate short term ad-hoc exploratory analysis
- Remove roadblocks to user self-service
- Allow external and/or private data integration
- Avoid the creation of unmanaged spreadsheets based data on user desktops
- Increase partnership between IT and Business

Questions