NOVA IMS Information Management School

# MDSAA

Master Degree Program in

**Data Science and Advanced Analytics**

## Lakehouse Data Architecture

Data as a first-class citizen within an organization

Fábio Rafael Santos Lopes

Internship Report

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

**NOVA Information Management School**
**Instituto Superior de Estatística e Gestão de Informação**

Universidade Nova de Lisboa

**NOVA Information Management School**
**Instituto Superior de Estatística e Gestão de Informação**
Universidade Nova de Lisboa

# Lakehouse Data Architecture

by

Fábio Rafael Santos Lopes

Internship Report presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science

## Supervised by

Flávio Luís Portas Pinheiro, PhD, NOVA Information Management School

November, 2023

# STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

*Lisbon, 21th of November of 2023*

# DEDICATION

For Ana, whose support made these words possible.

# ACKNOWLEDGEMENTS

I stand at the culmination of a significant chapter in my academic journey, and it is with immense gratitude that I acknowledge those who have been instrumental in guiding me to this point.

Bruno Cardoso, your leadership has been nothing short of inspiring. Your dedication to helping others excel has significantly impacted my work and personal growth. Thank you for being such a motivating force and constantly pushing us to do our best.

José Viegas, my journey through this process would have been much harder without you. Your companionship and unwavering support have made all the difference. We've been through thick and thin together, and I am grateful for every step of the way.

Harish Kumar, you are the visionary who made things possible in our organization. Your innovative ideas and determination have contributed to our success and inspired me to aim higher. Thank you for making us see what could be achieved with hard work and imagination.

To my parents, your love, dedication, and perseverance have shaped me into who I am. I owe everything to you and cannot thank you enough for your endless support and encouragement.

Thank you all for your invaluable contributions to my thesis and my life.

# ABSTRACT

This thesis presents an in-depth exploration of the Lakehouse Data Architecture. This paradigm merges the strengths of data lakes and data warehouses, enabling organizations to harness the full potential of their data. The research investigates the architectural components, operational mechanisms, and strategic implications of implementing a Lakehouse within an organization using advanced technologies like Microsoft Azure, Google Cloud Platform, Databricks, Apache Spark, Delta Lake, and Dremio. The study also scrutinizes the Lakehouse's ability to facilitate a data-centric culture by integrating advanced analytics into business processes. The thesis further delves into the FAIR data principles, advocating for data to be Findable, Accessible, Interoperable, and Reusable, and the Data Mesh concept, a decentralized data management approach. The research concludes that the Lakehouse architecture provides a comprehensive and robust framework for managing vast and diverse data sets, optimizing data pipeline performance, reducing redundancy, and enhancing data security. It underscores the pivotal role of the Lakehouse in driving strategic innovation and positions it as a flexible and adaptable model for future technological advancements in AI and machine learning. The insights offered in this thesis serve as a guide for organizations aiming to navigate the complexities of becoming data-centric and underscore the transformative power of modern data platforms.



# KEYWORDS

Data Mesh; Databricks; Data Lakehouse; Delta Lake; Microsoft Azure; Apache Spark


## Sustainable Development Goals (SDG):

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| **GCP** | Google Cloud Platform |
| **AWS** | Amazon Web Services |
| **EMR** | Elastic Map Reduce |
| **GCS** | Google Cloud Storage |
| **DBU** | Databricks Unit |
| **EAN** | European Article Number |
| **UPC** | Universal Product Code |
| **FPI** | Financial Performance Indicator |
| **AAD** | Azure Active Directory |
| **RBAC** | Role-Based Access Control |
| **CDC** | Change Data Capture |
| **API** | Application Programmable Interface |
| **ISO** | International Organization for Standardization |
| **MLOPS** | Machine Learning Operations |
| **IaaS** | Infraestruture as a Service |
| **PaaS** | Platform as a Service |
| **HDFS** | Hadoop Distributed File System |
| **AI** | Artificial Intelligence |
| **ML** | Machine Learning |

# 1. INTRODUCTION

In the digital era, data is often referred to as the new oil (The Economist, 2017), an invaluable asset that can drive innovation, strategic decision-making, and competitive advantage (Chen et al., 2012a). However, the potential of data can only be fully harnessed when treated as a first-class citizen within the organizational ecosystem (Alaimo & Kallinikos, 2022). This report explores the Lakehouse Data Architecture, a new paradigm that promises to elevate data to the forefront of an organization, ensuring its accessibility, usability, and integrity across diverse business functions (Zaharia et al., 2021).

The impetus for the Lakehouse model emerges from the convergence of two traditionally distinct data management systems: the data warehouse, known for its structured, curated datasets ideal for analytical workloads (Chaudhuri & Dayal, 1997), and the data lake, a more flexible repository capable of storing vast amounts of raw, unstructured data (Miloslavskaya & Tolstoy, 2016). The Lakehouse architecture embodies the strengths of both, providing a unified platform that supports a wide array of data types and workloads, from machine learning to business intelligence (Zaharia et al., 2021).

This study delves into the architectural components, operational mechanisms, and strategic implications of implementing a Lakehouse within an organization. It scrutinizes the architecture's ability to facilitate a data-centric culture by integrating advanced analytics into the business processes. The thesis evaluates the role of cutting-edge technologies in realizing the Lakehouse vision, including Microsoft Azure, Google Cloud Platform, Databricks, Apache Spark, Delta Lake, Dremio, and the emerging concept of a Data Mesh.

In a landscape where data architectures are rapidly evolving (Serra, 2024), this thesis stands as a testament to the innovative spirit of the field. It provides a comprehensive technical review of the Lakehouse architecture and situates it within a practical business context. The insights offered are targeted to guide organizations toward becoming truly data-driven, underscoring the transformational power of modern data platforms.

As we navigate through the chapters, we will uncover the foundational framework of the Lakehouse, discover the journey of the organization in study through different technologies, the challenges of the current architecture, the realization of a Lakehouse architecture, and two successful use cases put in place.

As a Solutions Architect, I helped define the organization's data strategy, drawing on the comprehensive knowledge from my Master's program. Alongside the team of Solution Architects, I orchestrated the design of our Lakehouse Data Architecture, engaging closely with business stakeholders to ensure their needs were met and with engineering teams to guide the development of the data platform. This synergy was critical for a scalable system prepared for future data management challenges. My academic expertise, especially in Big Data Analytics, Big Data Modelling and Management, and Storing and Retrieving Data, was crucial in successfully implementing a robust data architecture that supports diverse analytical workloads and the company's strategic objectives.

# 2. FRAMEWORK & TECHNICAL REVIEW

## 2.1. TECHNICAL REVIEW

### 2.1.1. Lakehouse Architecture

The world of data management has witnessed significant shifts over the years, with data lakes and data warehouses being primary staples for organizations (Arif & Mujtaba, 2015; Khine Pwint Phyu & Wang Zhao Shun, 2018).

Organizations have used data warehouses since the early 1990s and are a cornerstone of the world of data management (Jarke et al., 2002). A data warehouse is a specialized system that supports business intelligence activities, primarily analytical processing, reporting, and data retrieval (Chaudhuri & Dayal, 1997). Unlike data lakes, which store raw data, data warehouses contain structured, processed, and aggregated data.

The primary advantage of data warehouses lies in their optimized performance since they are built to handle complex queries swiftly, ensuring that businesses can derive insights and make decisions on time (Garani et al., 2019). Their structured nature ensures data consistency, reliability, and accuracy, making them a trusted source for critical business operations.

However, this structured approach brings its own set of challenges. Data warehouses require a significant upfront investment in design and architecture (Subramanian et al., 1997). The need for data to fit into predefined schemas can sometimes make them less adaptable to changing business needs or new data sources (March & Hevner, 2007). Scalability can be a concern, especially when dealing with vast and rapidly growing data volumes.

A data lake is a centralized repository that stores vast amounts of raw data, regardless of source or format. This can range from structured data from databases to unstructured data like images or videos.

Central to the data lake concept is the Hadoop Distributed File System, HDFS, a foundational storage system designed by Google to house vast amounts of data across distributed clusters of machines. Born from a paper that introduced the world to the Hadoop framework, HDFS was designed to be highly fault-tolerant, scalable, and suitable for applications with large data sets (Ghemawat et al., 2003). It breaks down large files into blocks, distributing them across nodes in a cluster, ensuring data redundancy and resilience.

The primary benefit of data lakes is their inherent flexibility. Organizations can store data in the lake without structure or processing, making it a scalable solution for ever-growing data volumes (Miloslavskaya & Tolstoy, 2016). This flexibility also allows for diverse analytical tools and methods to be applied, catering to varied business needs.

However, this flexibility brings some challenges. The very nature of data lakes, accepting data in its raw form, can lead to data quality and consistency issues. Without proper governance and management, data lakes can quickly become "data swamps", where the sheer volume and messiness of the data obscures valuable insights. (Knight, 2018)

Many organizations have opted for a two-tiered approach, leveraging the strengths of both data lakes and data warehouses (Zaharia et al., 2021). In this model, raw data is initially ingested into a data lake, taking advantage of its vast storage capabilities and flexibility in handling diverse formats. Once in the lake, data undergoes preliminary processing and refinement, transforming it into a more structured and analyzable form. The processed data is migrated to a data warehouse upon reaching this stage. Optimized for high-performance queries and analytics, this warehouse serves as a middle layer, ensuring that analysts and business users access clean, structured, and consistent data. This layered approach harnesses the raw storage power of data lakes while providing that the final analytical layer is robust and efficient, mirroring the functionalities of a traditional data warehouse at the cost of data duplication.

The Lakehouse seeks to bridge these two approaches, providing a platform that is adaptable to diverse data formats and capable of efficient analysis (Zaharia et al., 2021). By integrating the vast storage capabilities of data lakes with the structured analytical environment of data warehouses, the Lakehouse offers a unified platform, Figure 2-1. This platform is both adaptable to diverse data types and optimized for analytics.

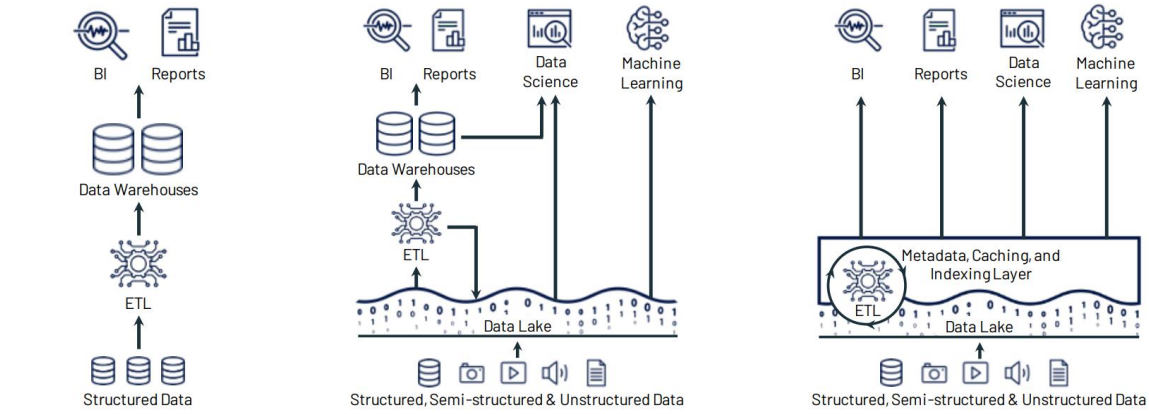



Figure 2-1 - Evolution of data platform architectures to today's two-tier model (a-b) and the new Lakehouse model (c) (Zaharia et al., 2021)

To achieve this, the first challenge that the Lakehouse tackles is the storage layer, both from the point of view of guaranteeing data quality/consistency and an efficient storage format. To this purpose, Databricks released Delta Lake in 2017 (*Databricks Launches Delta to Combine the Best of Data Lakes, Data Warehouses and Streaming Systems*, 2017) and open-sourced it to the Linux Foundation in 2019 (*The Delta Lake Project Turns to Linux Foundation to Become the Open Standard for Data Lakes*, 2019).

Delta Lake is a storage format that brings reliability to Data Lakes by ensuring that vast amounts of data stored are consistent and of high quality, addressing some primary challenges traditional data lakes face (Armbrust et al., 2020).

Delta Lake makes use of the open-source format Parquet. Parquet is a columnar storage format tailored for the big data ecosystem. Unlike row-based formats, Parquet organizes and stores data by columns. This structure is particularly advantageous for analytical operations, as it allows for efficient data compression and improved query performance (Belov et al., 2021), and by reading only the necessary columns for a query, Parquet minimizes I/O operations, leading to faster results and reduced storage costs (Le Dem & Li nong, 2014). However, Parquet does not guarantee the data quality/consistency in the Data Lake.

Traditionally reserved for the data warehouse, Delta Lake introduces ACID (Atomicity, Consistency, Isolation, Durability) transactions to data lakes, which are the central aspect of the ability of Data Warehouses to guarantee data quality/consistency (Haerder & Reuter, 1983). They guarantee that data, once written, is reliable, ensuring that subsequent reads retrieve accurate and consistent information.

- Atomicity, ensures that all operations in a transaction are completed successfully; if not, the transaction is aborted at the fail point and previously completed operations are rolled back to their former state.
- Consistency, after a transaction has been completed, it ensures that the system remains consistent.
- Isolation, ensures that multiple transactions can occur concurrently without leading to inconsistencies.
- Durability, once a transaction is completed, it ensures that the changes are permanent and can withstand system failures.

To further complement the data quality/consistency within the Lakehouse, Delta Lake introduces a pivotal feature in the quest to merge the flexibility of data lakes with the structured nature of data warehouses: schema enforcement and evolution. This characteristic ensures data remains adaptable to changing needs and structured for analytical precision.

Schema enforcement ensures the data adheres to a predefined structure or format (Andany et al., 1991). Even though data can be ingested in its raw form, it eventually conforms to a specific schema, ensuring consistency and reliability. This structured approach facilitates efficient querying and analytics, much like the benefits offered by traditional data warehouses.

However, as business needs evolve, so do the structure and nature of the data. As such, Delta also supports schema evolution. This means that schemas aren't set in stone; they can be modified and adapted over time without losing existing data or disrupting ongoing processes. It ensures that data remains organized for analysis while being resilient to the ever-changing landscape of business requirements.

In traditional data architectures, like Lambda, there's often a clear divide between real-time and batch processing methods. Batch processing deals with large volumes of data at once, typically at scheduled intervals, while stream processing handles data in real-time as it arrives (Kiran et al., 2015).

The Lakehouse, in its innovative approach, blurs this distinction. Supporting unified batch and stream processing ensures that data, whether arriving in vast batches or as continuous streams, is processed and made available for analysis with minimal delay. In Figure 2-2, historical data is available in a Delta table by ingesting raw files in a batch process, while real-time data is ingested in real-time from a database using Change Data Capture processes, CDC, and Apache Kafka (Thein, 2014). This unified approach simplifies the data processing pipeline and ensures that insights derived from the data are timely and relevant. (*Delta vs. Lambda: Why Simplicity Trumps Complexity for Data Pipelines*, n.d.)

For organizations, this means greater agility. They can make informed decisions based on historical data trends (from batch processing) while reacting to real-time events and anomalies (stream processing).

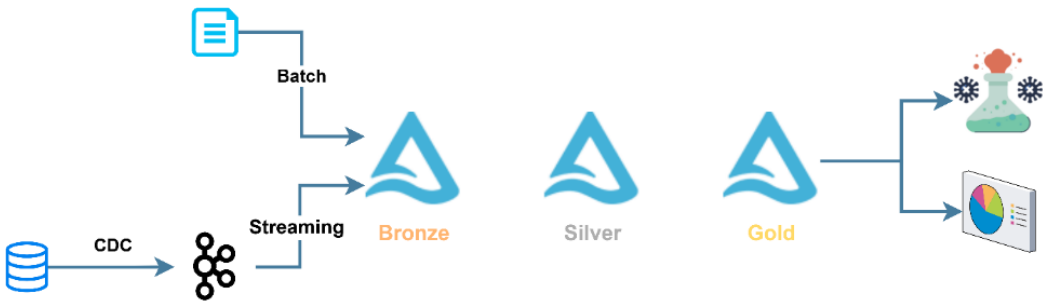



Figure 2-2 - Unified Batch and Streaming Analytics

This convergence of capabilities allows businesses to streamline their data infrastructure, reducing the need for complex ETL processes and multiple storage systems. With a Lakehouse, users can run real-time analytics, machine learning, and operational tasks in one place, providing a more cohesive and integrated data experience.

The approach to data management within the Lakehouse is called the Medallion Architecture (*Medallion Architecture*, n.d.). It is an approach to data organization and management that seeks to optimize the balance between performance and storage costs. At its core, it categorizes data into different tiers or "medallions" based on its value and access patterns.

- Bronze, the initial stage where raw data is ingested into the system. It's a landing zone for all types of data without any transformations. The data remains in its native format, ensuring no information is lost during ingestion.
- Silver, in this stage, data undergoes cleansing, enrichment, and some level of transformation. It's a transitional phase where data is prepared for analytics but might not yet be fully optimized for end-user consumption.

- Gold, the final stage where data is fully curated and optimized for consumption. It has undergone all necessary transformations, aggregations, and quality checks, preparing it for business intelligence tools and end-user queries.

As data progresses from raw to refined states (or bronze to gold), it moves through these medallions, Figure 2-3, each representing a different level of optimization and cost. This tiered structure ensures that frequently accessed or high-value data is readily available and optimized for performance, while less critical data is stored cost-effectively. (*Describe Medallion Architecture*, n.d.)


Figure 2-3 - Medallion Architecture from Bronze to Gold

The Lakehouse architecture requires a robust processing framework that ensures data is stored, effectively processed, and analyzed. Among these frameworks, Apache Spark is the preferred to bring the Lakehouse to life.

Apache Spark is an open-source, distributed computing system rapidly becoming the de facto standard for big data processing and analytics (Zaharia et al., 2010). Born out of the AMPLab at the University of California, Berkeley, Spark was designed to address the limitations of Hadoop's MapReduce computing model, offering a more dynamic and flexible approach (Zaharia et al., 2010).

One of Spark's standout features is its in-memory computation. By storing data in RAM across clusters, Spark drastically reduces the time to access data from storage, leading to significantly faster processing speeds (Zaharia et al., 2012).

A Spark cluster, represented in Figure 2-4, comprises the Driver nodes, where the main application resides and orchestrates tasks across the cluster, and Worker nodes, which carry out computations and store data in a cache for the application. The Worker nodes work in parallel, breaking down tasks for efficient processing.

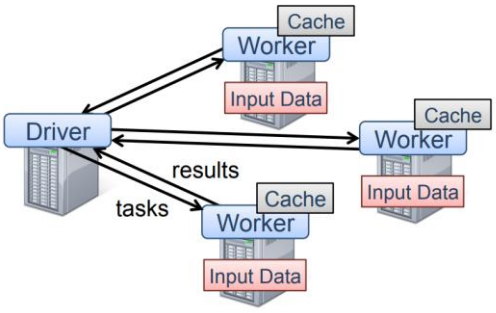



Figure 2-4 - Spark cluster configuration with 1 Driver and 3 workers (Zaharia et al., 2012)

Known for their fault tolerance and scalability (Han & Zhang, 2015), Spark clusters can distribute data processing tasks across multiple nodes. If a node fails, Spark's inherent fault tolerance ensures that tasks are rerouted to other nodes, guaranteeing uninterrupted processing. Moreover, as data demands grow, Spark clusters can be elastically expanded to accommodate the increased load, ensuring consistent performance.

It supports a wide range of workloads, from traditional batch processing to real-time data streaming. This adaptability ensures that the Lakehouse, powered by Spark, can cater to diverse data processing needs, whether periodic ETL jobs or real-time analytics.

Beyond its core processing capabilities, Spark boasts a rich ecosystem of built-in libraries:

- Spark SQL allows users to run SQL-like queries on large datasets.
- MLlib offers a comprehensive suite of machine learning algorithms, enabling data scientists to build models and derive insights directly within the Spark environment.
- GraphX caters to graph processing, allowing for analyzing complex networks and relationships.
- Spark Streaming facilitates real-time data processing, ensuring insights can be derived from data as it flows into the system.

The modular nature of Spark, combined with its scalability and performance, makes it a cornerstone of the Lakehouse architecture (Behm et al., 2022). By leveraging Spark, the Lakehouse can ensure that data, irrespective of its volume, velocity, or variety, is processed efficiently and is ready for analysis.

Additionally, since Delta is an open-source data format, any query engine can analyze data in the Lakehouse. Data is not locked in a proprietary format specific to each vendor of previous Data Warehousing technologies. Multiple query engines have added support for Delta; others are adding it, as depicted in Figure 2-5.

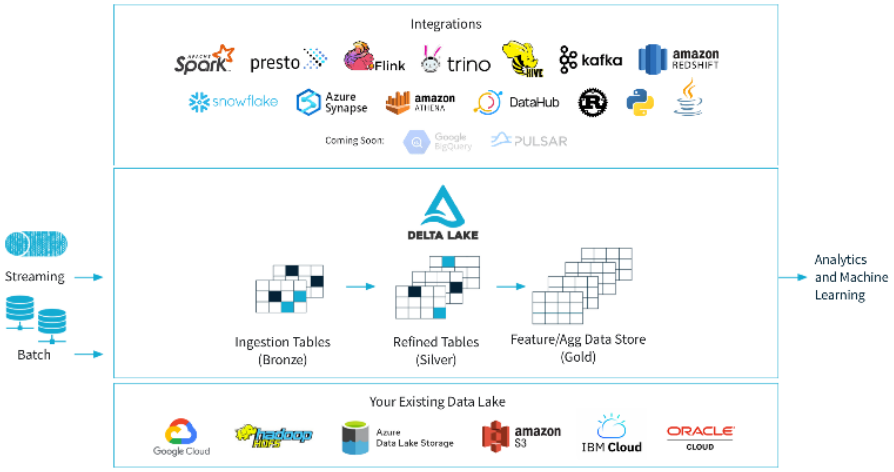



Figure 2-5 - Unified Batch Streaming into a Lakehouse with available integrations (*Delta Lake*, n.d.)

The Lakehouse architecture inherits the challenges of data fetch latency and I/O concerns from Data Lakes, which can hinder performance both in the processing and reporting layers (Miloslavskaya & Tolstoy, 2016). Indexing and caching become crucial to improve the latency of queries reading data from the data lake. Without indexes, the system would have to perform a full table scan every time a query is executed, which is highly time-consuming and resource-intensive. With the aid of indexes, query performance is dramatically enhanced, ensuring that data retrieval operations are rapid and smooth (Lehman & Carey, 1985).

Delta Lake introduces indexing in the Lakehouse by offering Z-Ordering, Figure 2-6, a method that maps multi-dimensional data onto a one-dimensional space, ensuring that data points that are close in the multi-dimensional space remain close in the one-dimensional space (Sugiura & Ishikawa, 2023). This technique is particularly beneficial for datasets with multiple attributes, as it ensures that related data points spanning across these attributes are stored in proximity. When executing range-based or multi-dimensional queries, Z-Ordering minimizes the data that needs to be scanned, leading to faster results. (Powers, n.d.)

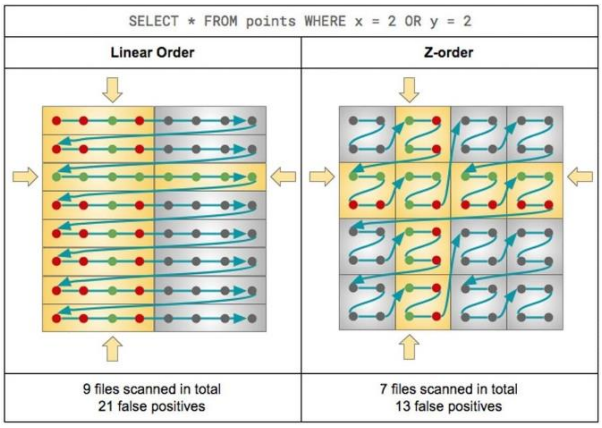



Figure 2-6 - Z-Ordering effect on the number of files scanned (Cheruvu, n.d.)

Caching mechanisms address the issue by storing frequently accessed data closer to the processing nodes of the Spark cluster, reducing the need to fetch data from remote storage, thereby mitigating latency repeatedly (Zhang et al., 2017). This ensures that popular datasets are readily available, boosting query performance.

During ETL, data is fetched from the remote storage and cached in the local compute nodes so it can be reused by multiple processes requiring the same raw data. The same applies to reporting, where query results are cached in local memory so they can be served to multiple users without requiring a fetch from remote storage and computing.

## 2.2. FRAMEWORK

### 2.2.1. F.A.I.R

Published in 2016, F.A.I.R. (Wilkinson et al., 2016) consists of a set of principles that define a framework for data management and stewardship, aiming to improve the capacity of computational systems to find, access, interoperate, and reuse data. These principles have gained support from academic researchers (Mons et al., 2017) and were embraced by the European Union in publishing EU-funded research data (Dutch Techcentre for Life Sciences, 2016).

To increase the usefulness of data and increase its value to organizations, F.A.I.R. introduces its four principles, Figure 2-7, Findable, Accessible, Interoperable and Reusable.

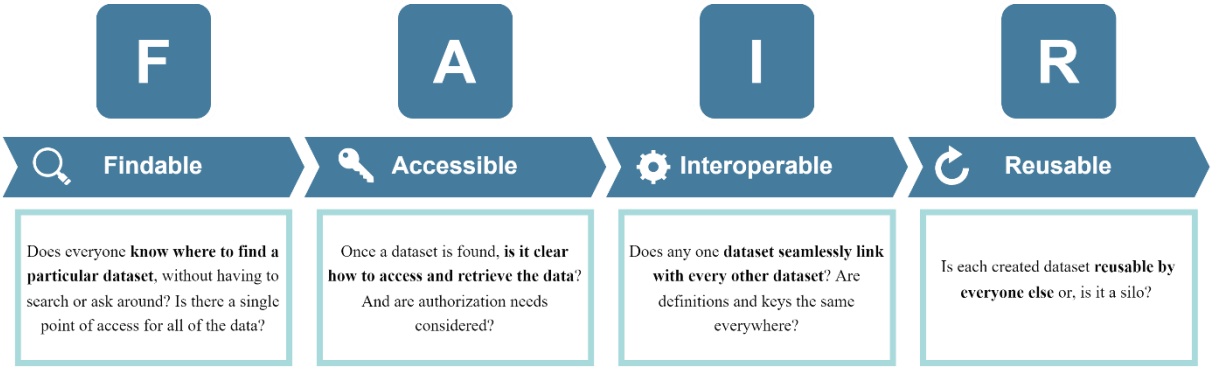



Figure 2-7 - The 4 F.A.I.R. Principles for data management and stewardship

The principle of "Findable" underscores the importance of data being readily locatable through precise, reliable metadata and distinct identifiers. This element is fundamental for ensuring that data can be found and used by all potential data stakeholders.

To enhance the findability of data, it is imperative to create metadata that is both precise and reliable, encompassing details such as the data's title, owner, creation date, and description. This metadata should be organized systematically and maintained uniformly across diverse data collections. Such organization and maintenance enable the straightforward identification

and retrieval of data in response to particular queries, which may include search terms or the names of data proprietors.

Furthermore, data should be consolidated within a central repository to broaden its reach. Centralizing data in this manner ensures that it can be readily found and accessed by individuals, including those unaware of alternative data stores or specialized data management facilities.

The principle of “Accessibility” emphasizes that data needs to be readily obtainable by both human users and computer systems, accompanied by explicit information on usage rights and licensing agreements. For human comprehension, it is critical to supply comprehensive documentation regarding the data's nature, configuration, and structure. This documentation must be straightforward and formatted in a manner conducive to human consumption. Regarding machine accessibility, offering data in formats and through Application Programming Interfaces (APIs) that facilitate seamless retrieval and assimilation with additional datasets and software systems is pivotal. This involves providing data in open, machine-friendly formats like CSV, JSON, Parquet, or Delta and the availability of APIs for automated access.

The principle of “Interoperability” highlights that data should be structured to facilitate its usage with other datasets and adherence to prevalent standards and protocols. Data interoperability is critical for the seamless interaction of distinct systems and technologies. It necessitates data to be curated in a manner that promotes its incorporation with other datasets and technological frameworks, necessitating the provision of explicit details regarding its format, construction, and meaning, along with adherence to established data standards. Adopting universally recognized data standards like those developed by ISO is a strategic approach to ensure data interoperability. Such standards offer a unified data description and representation methodology, enabling disparate systems and technologies to interpret and engage with the data uniformly.

The principle of “Reusability” details the necessity for data to be crafted with the future in mind, ensuring its utility extends beyond its initial application. This is facilitated through comprehensive documentation and a detailed record of its provenance. Reusability denotes the capacity of data to fulfill varied roles and cater to the needs of different users or groups, extending beyond its primary objective. To enhance data reusability, it is vital to equip it with ample documentation and a clear account of its lineage, encompassing aspects of its source, integrity, and the modifications it has undergone.

The lineage aspect encapsulates the history of the data, detailing the originator's identity, the time frame of creation, and the methodologies employed in its design. Such details are imperative for assessing the data's context and dependability, thereby serving as a foundational element that informs subsequent evaluations and applications by other researchers or analysts.

Adopting the FAIR principles is a transformative step towards enhancing the utility and longevity of data within the realm of research and beyond. By adhering to these principles, the data becomes not only more accessible and retrievable but also ensures its compatibility and integration with other datasets, thereby augmenting the collaborative potential among various stakeholders (Wilkinson et al., 2016). Collectively, these principles elevate the intrinsic value of data, establishing a standard for data stewardship within organizations.

### 2.2.2. Modern Data Platform

A data platform is a collection of technologies and tools used to manage, analyze, and utilize large amounts of data from various sources (Ferguson, 2012). It includes multiple components such as data storage, processing, integration, quality, governance, security, and visualization (Lewis et al., 2001). Its goal is to make it easy for organizations to access, analyze, and gain insights from their data to drive business decisions and outcomes. It also provides a way to manage data in a centralized and efficient manner and make it accessible to different teams and departments within an organization.

In the modern digital age, cloud computing has emerged as a dominant force, reshaping how businesses and individuals manage and store their data. Large and small organizations are increasingly transitioning their data platforms to the cloud. This shift is driven by the cloud's inherent scalability, flexibility, and cost-effectiveness (Demchenko et al., 2013). By leveraging cloud solutions, businesses can achieve more efficient operations and adapt more readily to changing technological landscapes (Serra, 2024).

Parallel to the rise of cloud computing is the explosive growth of big data. As data volume, variety, and velocity have surged, there has been a pressing need to develop robust frameworks capable of processing these vast amounts of information (Raguseo, 2018). These data processing frameworks have revolutionized how we handle and analyze big data, enabling swift and efficient processing. With these tools, businesses can glean valuable insights and make more informed decisions (Chen et al., 2012b).

The technological landscape has been further enriched by advancements in machine learning and artificial intelligence, AI. These fields have ushered in a new era of data technologies, including deep learning and computer vision frameworks (Voulodimos et al., 2018). With these cutting-edge tools, it's now possible to construct more intricate and accurate models. The implications are vast, from enhancing business operations to making groundbreaking discoveries (Athey, 2018).

The digital realm is not just about static datasets; it's also about the continuous flow of information. Streaming data sourced from avenues like IoT devices, social media platforms, and system logs is becoming increasingly prevalent (Armbrust et al., 2018). To manage and harness the potential of this real-time data, new technologies have emerged that ensure that businesses can react in real time, optimizing operations and enhancing user experiences (Kolajo et al., 2019).

With the proliferation of data and the increasing reliance on digital platforms, there's a heightened emphasis on data governance and security by organizations (Delacroix & Montgomery, 2020). Big data and cloud computing challenges necessitate robust solutions to ensure data integrity, privacy, and security (Qlik, 2019). New technologies have been developed to address these concerns, including tools for data cataloguing, metadata management, data lineage tracking, encryption, and masking. These tools safeguard sensitive information and ensure that businesses remain compliant with evolving regulatory standards.

In data management and exchange, the proliferation of open data formats is a testament to the global push for transparency, interoperability, and collaboration. Their rise can be attributed to their myriad benefits, including fostering innovation, ensuring data consistency across platforms, and reducing integration costs (Arribas-Bel et al., 2021). By providing a standardized way to represent and share data, these formats eliminate the barriers often posed by proprietary systems, enabling seamless data exchange across diverse applications and organizations. As the digital landscape continues to evolve, the emphasis on open data formats underscores the collective vision of a more connected and transparent digital future.

Building a data platform requires a deep understanding of data management, architecture, data processing, and analytics (Pratsri & Nilsook, 2020). An extensive team of data engineers, data scientists, data analysts, and data architects must work together to design, implement, test, deploy, and maintain the data platform.

To keep up with these advancements, organizations often need to modify their existing data platforms to incorporate new technologies and tools (Nadkarni & Prügl, 2021). This can involve migrating data to new storage systems, integrating new data processing frameworks, and implementing new data governance and security technologies. It is challenging, but it's crucial for organizations to continuously monitor and evaluate the data platform and make improvements as needed to keep up with the changing business requirements and the latest technologies (Reinsel et al., 2018).

### 2.2.3. Data Mesh

Data Warehouses and Lakehouses are inherently centralized architectures, usually managed by a central IT team responsible for all operations and developments. The concept of a Data Mesh is a shift in this approach, moving traditionally centralized models to a more decentralized framework with four specific characteristics (Dehghani, 2019, 2020, 2022): Domain Ownership, Data as a Product, Self-Serve Data Infrastructure and Universal Governance.

- Autonomous Data Ownership, In conventional architectures, the central IT teams have the responsibility to manage all data assets. A data mesh distributes the responsibility by allowing individual teams full ownership of their data. This decentralization accelerates data processing and fosters a sense of responsibility and accountability.

- Product-Centric Data Presentation, Data is organized into domains, treated into products and used by use-cases which can be cross domain/data product. Curating data to be easily accessible, trustable, and adaptable to myriad use cases. When data is seen as a product, it enhances its value and usability across an organization.
- Automated Infrastructure, The emphasis is to reduce manual interventions and automating data processes. Automated provisioning ensures that resources are allocated efficiently, reducing lags and ensuring smoother operations.
- Universal Governance, With diverse teams handling data, it becomes paramount to have universal standards. Governance in a data mesh ensures that teams operating autonomously adhere to global standards, especially concerning security and data quality.

Figure 2-8 shows the logical representation of a Data Mesh applied to the concepts above of Data Domains, Data Products, and Use cases. In a Data Mesh, one or more data sources are prepared to create a self-sustained entity called a Data Product. Data Products are grouped into Data Domains and are consumed by Use cases, which can be cross-domain. These concepts are supported by a common, shared data infrastructure maintained by a central IT team.

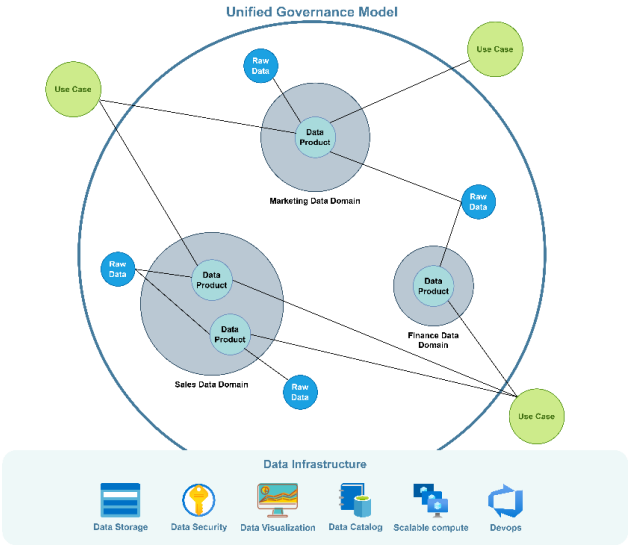



Figure 2-8 - Conceptual Diagram of a Data Mesh across domains, products, and use cases

A Data Mesh is not a technology but a concept designed to help drive an organizational and cultural shift within companies struggling to obtain value from their data (Serra, 2024). The technology used to build a data mesh could follow the Data Warehouse or Lakehouse architecture, or domains can even follow different architectures.

# 3. BUSINESS CONTEXT

## 3.1. COMPANY

This internship was executed in a multinational consumer goods manufacturer with over 200 years of history in the health, hygiene, and nutrition markets. Across the globe, more than 40,000 employees work together to create a reference in all the product categories where the company competes.

The company soon realized that data should be at the core of the business. As such, an integrated structure of professionals was created composed of data ambassadors in each relevant region of the globe, plus a central data team, as seen in Figure 3-1. These two entities work in synergy by making sure that the data products available in the company are adequately developed, maintained, and supported.

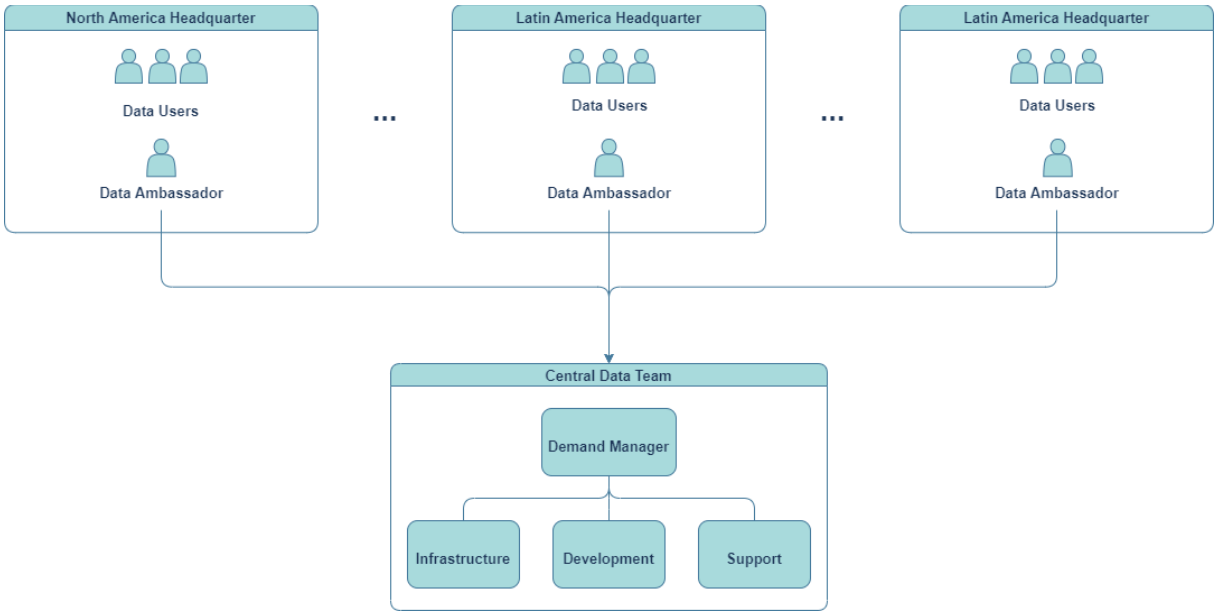



Figure 3-1 - Organization of the Data Team within the organization

The data ambassador sits with the central operational hub of a specific region, supporting the business in the day-to-day operations with data in the form of visualizations, local data management, and communicating with the central team about possible data issues or new features requested by the business.

The Central Data Team comprises multiple hubs across the globe, in countries like the Netherlands, Portugal, India, and Poland, where various engineers work together on different data products. The structure of this global team is split into four roles with distinct responsibilities:

- The Demand Manager acts as a gateway between all the development requests from the Data Ambassadors and the Development team, coordinating timelines and prioritizing all the work needed.
- Support is responsible for monitoring data processes throughout the day and helping the data ambassadors with requests.
- Infrastructure is responsible for deploying and maintaining all cloud resources used in the company.
- Development, composed of Data Engineers and Data Scientists, is responsible for maintaining the existing data products and developing new features/products requested by the business.

The work described in this report was executed within the context of the Data Engineering Team of the Portugal Hub, located in Lisbon. This hub is mainly focused on the company's main reporting platform, which is used across the organization for report building and as the data source for most of the data science work across the company.

## 3.2. The Data Architecture Journey

The wide adoption of data products within the organization created a strong mentality in the data team to always strive to improve the current products and keep exploring and bringing forward new emerging technologies to be used. As such, every new iteration of the data architecture, Figure 3-2, originated from this mentality.

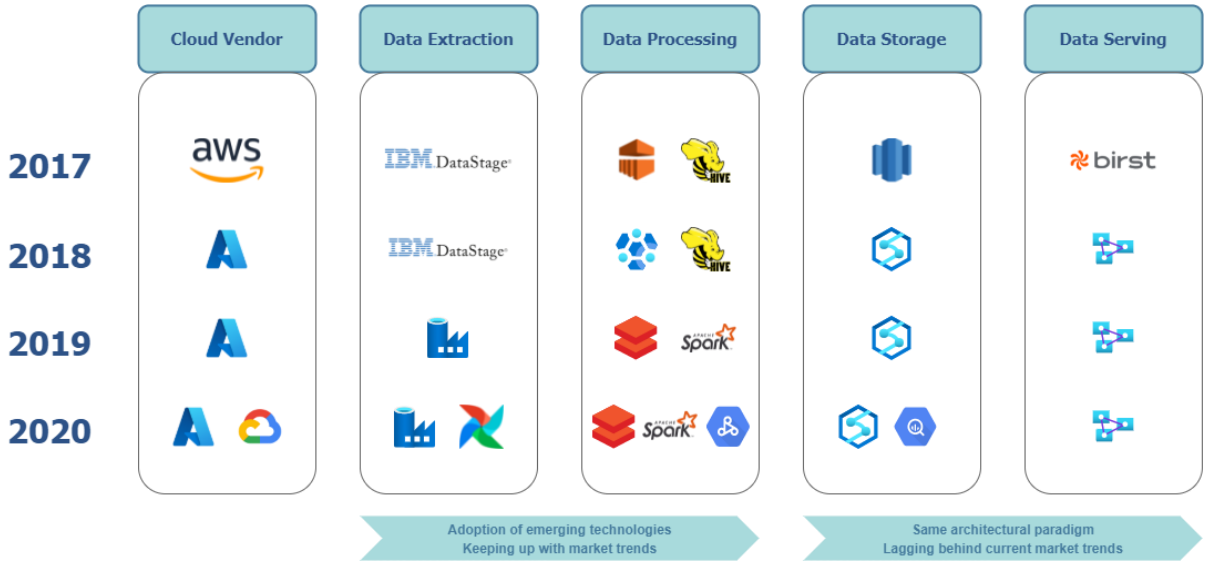



Figure 3-2 - Diagram of the data architecture journey of the company

In 2017, the data extraction from the company's Enterprise Resource Planning systems, using IBM DataStage, ran in a company server due to security policies. The extracted data was processed using Hive scripts running on an AWS EMR cluster and stored on an AWS Redshift data warehouse. To serve the data, the company used pre-built reports using Birst.

The main limitation of this architecture came from the serving layer where, by only providing pre-built reports, the ability to expose the data entirely was hindered. Across the company, the desire to expose data to users and allow them to be self-sufficient in report building led to the search for a platform to achieve this. Eventually, the platform chosen for this effect was Azure Analysis Services with PowerBI as the user-facing software, both developed by Microsoft, which, at the time, was the leading company in Business Intelligence according to the 2018 Gartner's Magic Quadrant for "Analytics and Business Intelligence Platforms" (Howson et al., 2018), in Figure 3-3.


Figure 3-3 - 2018 Gartner's Magic Quadrant for "Analytics and Business Intelligence Platforms" (Howson et al., 2018)

The choice of Microsoft-developed products naturally triggered a migration from AWS to Azure, and to achieve this, the Lisbon Hub was created in 2018. From a data architecture perspective, the Data Extraction layer remained the same. Still, the data was now uploaded to a different cloud, the Data Processing layer was running the same scripts but on HDInsight clusters, and the data was still stored in a data warehouse, Azure Synapses dedicated pool (Azure Data Warehouse at the time). But, with the change of architecture in the Serving Layer, users were now fully empowered to develop their reports, which were being served by a fast in-memory engine. Overall, the data consumption within the company increased, and the migration was considered a significant success.

Months after the new migration, two major components were identified as possible improvements: Data Ingestion and Data Processing.

The Data Processing component, using HDInsight, was becoming troublesome for the team to maintain mainly due to:

- The data storage format chosen during the initial development of scripts was RCFile, which was already outdated and greatly surpassed by newer formats like Parquet or ORC.
- Intermittent issues with cloud deployment of new on-demand clusters became the norm and not the exception, requiring significant effort from the Support Team to recover from the failure of processes.
- Overall, jobs took several hours to complete, leading to higher costs on infrastructure and complaints from the Data Ambassadors where local teams were left waiting for jobs to finish at the start of each working day.

The solution found by the team was two-fold. First, migrate the Hive scripts to Spark SQL and upgrade the data storage format to Parquet. Second, move all the processing to Azure Databricks, which, at the time, was considered a leader in the Data Science and Machine Learning Magic Quadrant by Gartner (Idoine, 2019), Figure 3-4.


Figure 3-4 - 2019 Gartner's Magic Quadrant for "Data Science and Machine Learning Platforms" (Idoine, 2019)

With this migration, the team was able to improve on all three aspects cited above, namely:

- With the data stored in Parquet, around 50% of storage costs within the platform were reduced.
- Databricks Cluster deployment and management were smooth, allowing the Support and infrastructure Teams to focus their time on other aspects of the platform.
- The team achieved a 50% decrease in job execution time with a 40% reduction in the total costs of each job.

The Data Ingestion component, at the time, was entirely handled by the DataStage software developed by IBM and was responsible for connecting to the company's internal servers,

pulling the data, and preparing it to the standard schema accepted by the data processing component. However, two problems existed:

- Monitoring of the software was done on an entirely different portal by the Support Team. It couldn't be integrated into the existing monitoring solution, requiring the Support Team to monitor running processes in 2 distinct tools.
- At the time, the primary use case for the software was the connection to the source systems with little use of other capabilities. As such, the license price paid for the software became too high for the usage given.

The solution to both problems came from using the Azure Cloud Stack to migrate all processes to the cloud as much as possible. To achieve it, the team used a Self-Hosted Integration Runtime, deployed within the Company's internal network (enabling connection to the internal data sources) paired with Azure Data Factory to manage and orchestrate the queries executed against the source systems. Rather than using DataStage to connect and harmonize the data, Azure Data Factory was used to extract the data from the systems, and Azure Databricks was used to harmonize the data and store it for consumption by the Data Processing component, as seen in Figure 3-5.

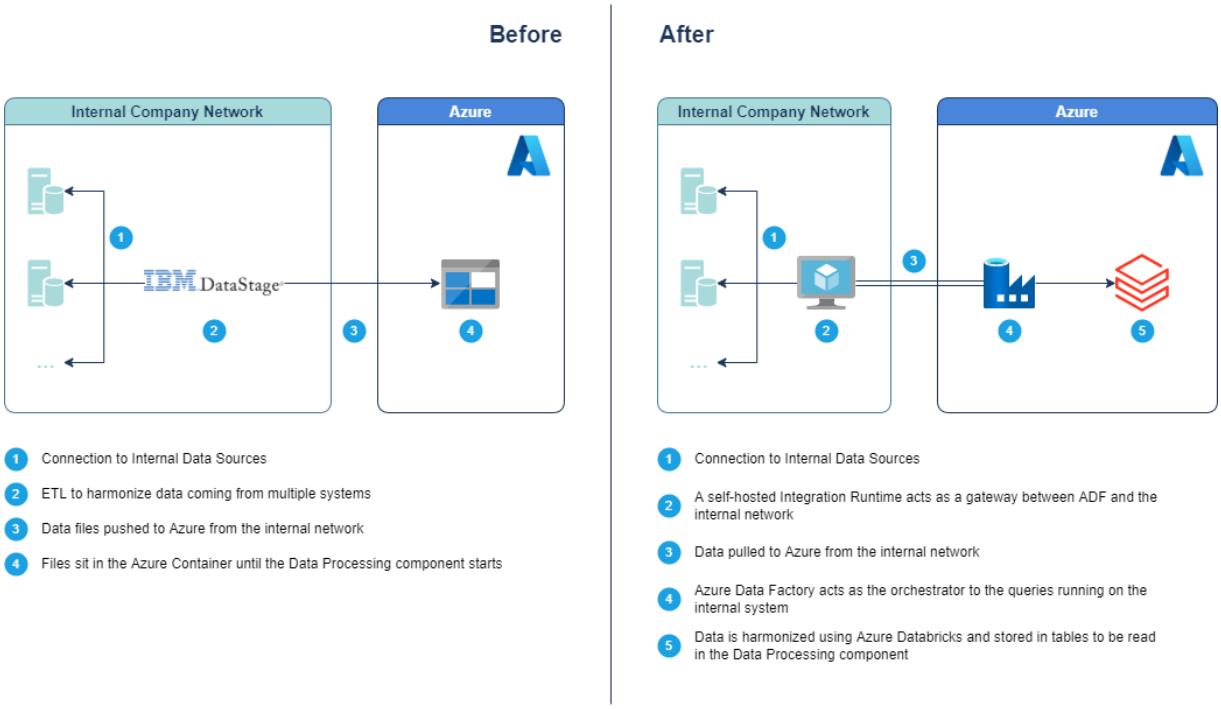



Figure 3-5 - Before and After of the Data Extraction component architecture

At the time, Data Science was still in its infancy within the company, and there was no clear vision or strategy. It mainly comprised small independent projects working on locally acquired datasets and data from the main data platform.

With a change in leadership and new funding, data science was added to the capabilities of the data team, and unique architectural necessities were created. Mainly, the initial push for

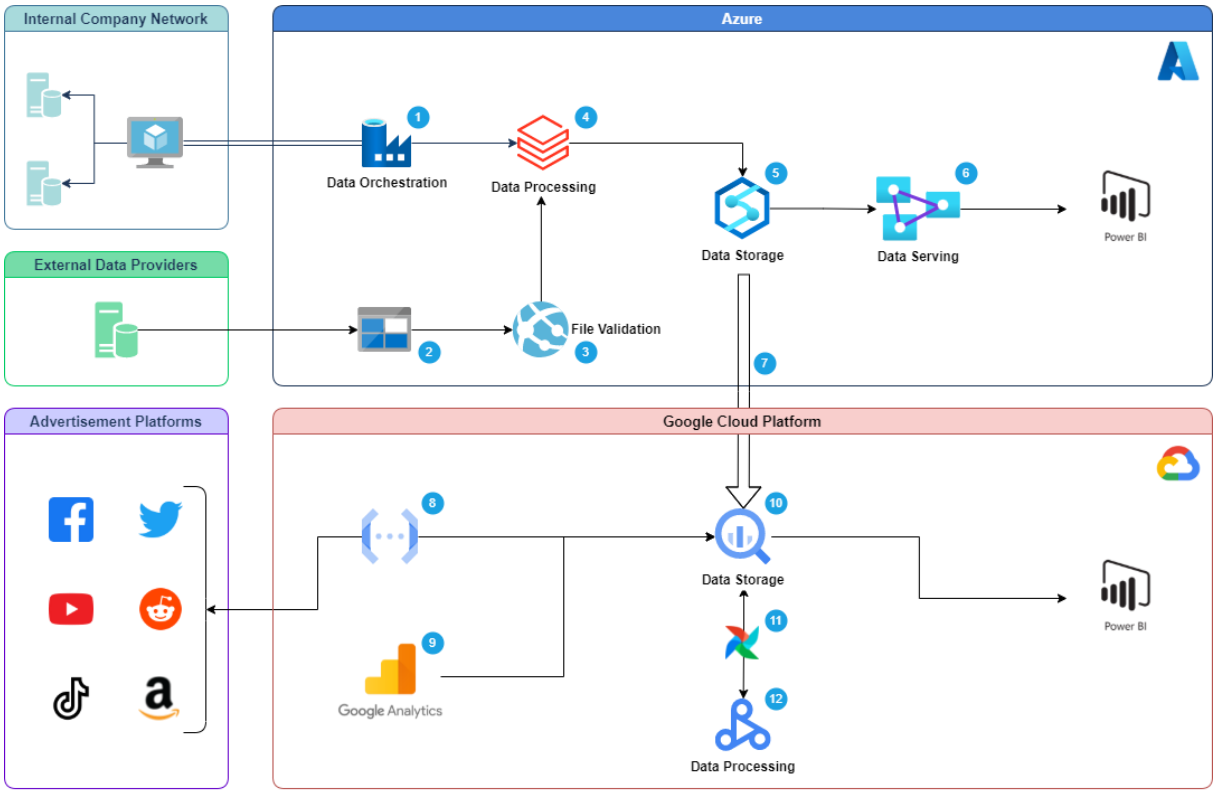
data science within the company was to analyze the return on investment being placed on media platforms, and this would require a new class of data, which the data platform was not accommodating. The significant bulk of this new data class was comprised of Google Analytics, and, at the time, there were two methods to obtain this data:

1. API, where data was pulled from Google on-demand.
2. Integration of Google Analytics with Google BigQuery in Google Cloud Platform, where data was pushed to the system.

The first method would allow the team to remain within the Azure Cloud Stack, but Google does not expose all the data via API, limiting the number of features that can be extracted. This was a significant constraint and triggered a multi-cloud approach by the team since the features were required for analysis by the data science team.

The current architecture, Figure 3-6, is then composed of a multi-cloud stack where two distinct data classes are processed and made available for reporting and data science use cases. On Azure, data is acquired by connecting to the internal source systems with Azure Data Factory or receiving text files from external data providers. Both sources are processed with Azure Databricks, stored in Azure Synapses' dedicated pool, and served in Azure Analysis Services. A weekly copy of selected datasets to Google BigQuery is performed to enable data science use cases.

In Google Cloud, advertisement data is captured using the respective APIs with Python scripts running on Google Cloud Functions, and Google Analytics data is refreshed automatically within BigQuery. All data processing and data science use cases are processed using Google CloudProc and orchestrated via Apache Airflow.




1. Connection to Internal Data Sources, where Azure Data Factory executes queries against the source system and pulls the data for further processing.
2. External Teams push text files to Azure Blob Storage in accordance to the standard schema defined.
3. Files received from External Teams go through a validation process running in an Azure Web App, where schema integrity, primary key validation and date formats are assessed.
4. Data is processed using Azure Databricks.
5. All historical data is stored for later consumption in Azure Synapses. It serves as the main data storage solution in Azure.
6. Azure Analysis Services cubes are refreshed with the latest data, giving users the ability to query the latest 2 years of data at any given time.
7. A weekly copy from Azure Synapses to Google BigQuery is performed for certain datasets required for Data Science purposes.
8. Scripts running on Google Cloud Functions connect to the API's of advertisement platforms to extract all the available data.
9. Data from Google Analytics is automatically ingested into Google BigQuery using the available connector.
10. Google BigQuery hosts all data for advertisement platforms and the model results from the Data Science projects.
11. Apache Airflow is used to orchestrate the execution of the Data Science processes.
12. Data is processed and models are trained using Google Cloudproc. Results are written back to Google BigQuery.

Figure 3-6 – Diagram of the current data architecture of the platform

## 3.3. LIMITATIONS OF CURRENT ARCHITECTURE

### 3.3.1. Regional Isolation

One of the requirements put in place by the leadership is to isolate the data to ensure that teams visualize only the data of their region. To accomplish this, in the Serving Layer, data is split across 13 different Azure Analysis Services servers (which correspond to 13 business-defined regions), and each regional team can only access their respective server, Figure 3-7.

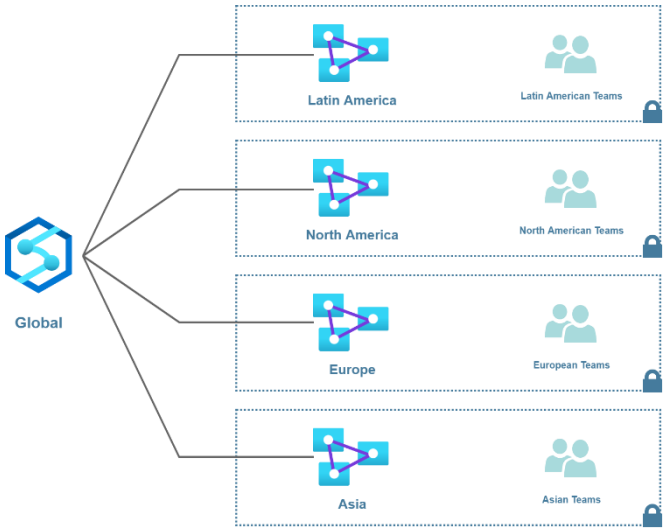



Figure 3-7 - Assignment of regional teams to separate regional Azure Analysis Services servers

However, over time, Data Ambassadors have requested to restrict this isolation further, where some members could not visualize data for certain countries within a specific regional team. Implementing such a feature would require the introduction of Row Level Security in the Azure Analysis Services server, in which previous tests performed revealed a lackluster performance. This feature is still not implemented despite the continuous push from the Data Ambassadors.

The second feature with high demand is being able to block users from using specific datasets within the server; one user might be able to visualize data for a particular dataset but not the other. Although possible by resorting to model perspectives within Azure Analysis Services, applying this feature for the hundreds of users across 13 different servers would be unmanageable.

Although this isolation successfully splits the users across different regional teams, there is a need to support reporting for the global leadership teams, which require data to be available for all countries, irrespective of the region. Due to the size of the data, this is only possible within specific critical datasets, and, as such, an Azure Analysis Services server was deployed containing data for all countries for a restricted list of datasets for a limited group of users.

### 3.3.2. Data Management Agility

Overall, there is a feeling that the architecture does not allow agility in both the development work and the local market's daily operations.

From the development perspective, a simple request as the inclusion of a new column in one table results in the following four main steps, as seen in Figure 3-8:

- Data Extraction, any new request from the market to introduce a new dataset and a change to an existing change needs to include a deep analysis on what the exact request entails for the rest of the datasets. As the data model is complex, a small

change can have profound implications. After the request is understood, a source system must be identified, and all the necessary extraction queries must be prepared to be introduced in the Azure Data Factory pipelines.

- Data Processing, during this step, the new schema from the previous step must be analyzed and incorporated into the ETL scripts. Tables within the Azure Databricks workspace must be modified, and the export to the Data Warehouse must be adapted to include the change.
- Data Storage, all the Stored Procedures which reference the table need to be altered to adapt to the new schema. The schema of the fact tables will need to be updated.
- Data Serving, as stated in section 3.3.1, 13 Azure Analysis services share the same model, and, as such, the model needs to be updated and re-deployed to the 13 servers.

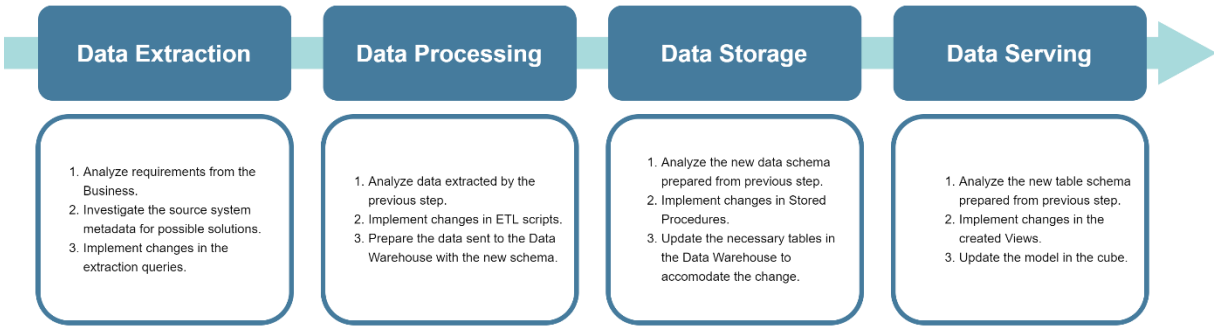



Figure 3-8 - Current development flow within the platform

Due to the sheer variety of services and the complexity of the data model, a simple request will almost always cascade to many changes in the platform with multiple breaking points, reducing the overall agility of the development team to introduce changes and extending development time for simple requests to weeks.

The entire platform was built on top of the expectation that all data would be available for processing before the start time of the pipeline and that the markets would have the refreshed data available for consumption at the start of the day. The daily flow of operations can be seen in Figure 3-9, where there is only one scheduled processing period per region. This creates issues when local markets acquire refreshed data from their local sources throughout the day but must wait for the following day's pipeline to have the data available.

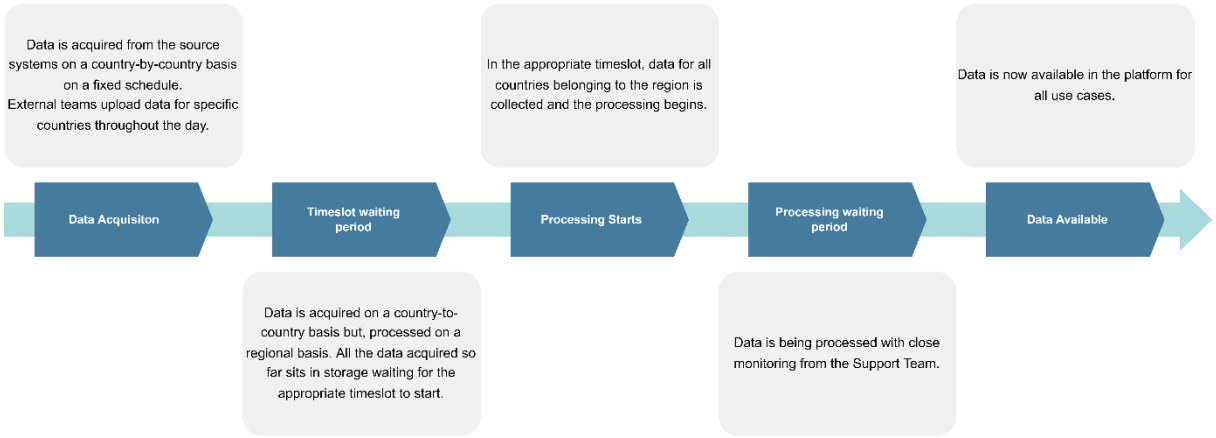



Figure 3-9 - Current daily operations flow

### 3.3.3. Read and Write Concurrency

In the dedicated SQL pool within Azure Synapse Analytics, all fact tables are strategically partitioned by calendar month (e.g., January 2022 is denoted as 202201) to enhance query performance. This partitioning scheme aligns with established querying patterns identified through prior analysis. Utilizing the partitioned structure, stored procedures in the dedicated SQL pool leverage the 'ALTER TABLE SWITCH PARTITION' command, as illustrated in Figure 3-10. This command facilitates the transition of data into a transient state before the targeted partition in the final table is seamlessly replaced with the prepared data, effectively superseding the existing data. This method proves to be more efficient than the conventional DELETE followed by INSERT operations because it is a metadata change without data movement; it does not incur high transactional log operations.

Nevertheless, this approach induces a read-and-write lock on the table, which precludes concurrent access to the table during the partition switch, regardless of the specific partition involved. Consequently, this can lead to operational challenges where intensive queries executed by other systems or users can significantly delay the execution of data pipelines. Moreover, while executing the 'ALTER TABLE SWITCH PARTITION' command, the table becomes inaccessible to users as it is locked.

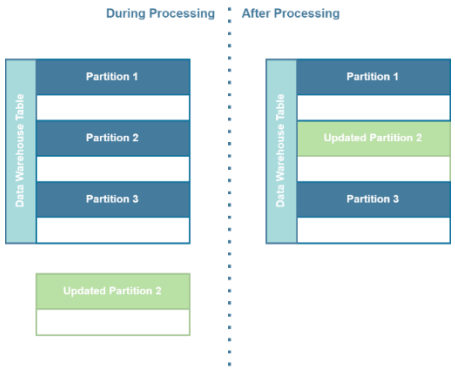



Figure 3-10 – The state of partitions within an Azure Synapses dedicated pool table during an ALTER TABLE SWITCH PARTITION statement

### 3.3.4. Source of Truth

A single source of truth is crucial in a data platform because it ensures that all users and systems are working with the same accurate information. Without it, different users and systems may have different versions of the same data, leading to inconsistencies and errors. A single source of truth makes it easier to understand and trust the data, as there is only one version to review and analyze.

The previous section renders the data warehouse ineffective as a single source of truth and incapable of serving multiple use cases throughout the organization. Over time, when new related projects that required data from datasets available in the platform started, the solution adopted was to create a static copy of the needed service with the required frequency. Besides making unnecessary data movement (and cost) in the platform, it also creates different sources of truth for different organization personas, Figure 3-11.

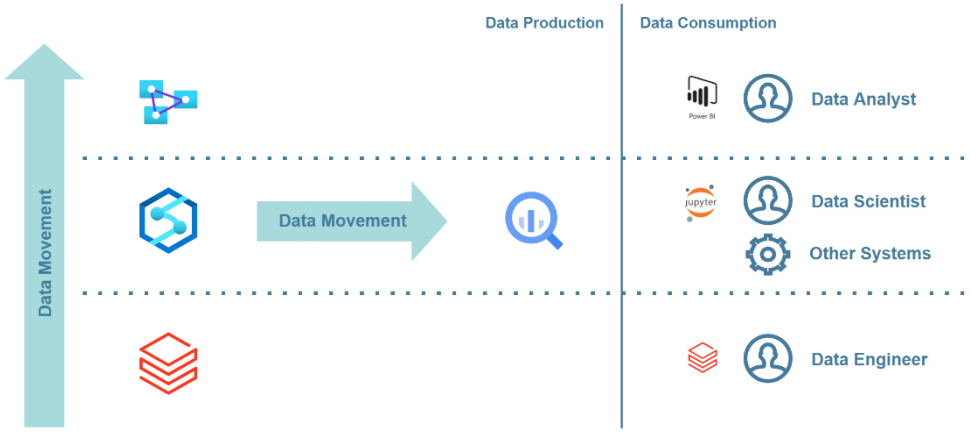



Figure 3-11 – The different tools and services used by different data teams

### 3.3.5. Fixed Compute

In the past, companies that needed infrastructure would have to plan and pay for the necessary capacity even if, in the end, resources might be underutilized. The cloud brought a "Pay as you go" approach where only the utilized resources are paid, and, as such, capacity planning becomes much more flexible. This allowed companies to shift from treating infrastructure from capital to operational expenses.

In the current architecture, the cloud deployment has three main computing components: Azure Databricks, Azure Synapse Analytics dedicated pool, and Azure Analysis Services.

In Azure Databricks, two mechanisms are used to capitalize on the "Pay as you go" approach: job clusters and auto-scale. When Job Clusters are utilized, one cluster will be created for a specific job execution and terminated upon its conclusion. Besides only paying for the exact duration of the job, this type of cluster is also considered a "Job Compute" by Databricks, and, as such, the cost per DBU (the hourly license price for a Databricks execution) is lower. Auto-

scale is a feature in Azure Databricks that allows a cluster to adjust the number of workers automatically based on the job necessity.

However, both Azure Synapses Analytics dedicated Pool and Azure Analysis Services use a pre-allocated set of compute characteristics without the ability to adjust it based on utilization dynamically. This requires the team to adapt the capacity of each service by the peak utilization during the day, which results in periods of significant underutilization.

### 3.3.6. Model Incompatibility and Over usage of Source Systems

Due to the scale and complexity of the organization, throughout time, multiple data products were developed and maintained by other teams separate from the central data team. These products are usually built to answer a specific business question and, if proven successful, will be transitioned to the data team for further development and maintenance. However, they suffer from two problems: their data model is built in such a way that renders them unsuitable for cross-reporting with the main reporting solution and, in some cases, if the sources of data are the same, will result in duplication of raw data being spread throughout the cloud environment and strain placed upon the source systems.

The most common root cause of model incompatibility in the manufacturing industry is the product codes used across multiple data products. Internally, the company uses an SKU tracking system that assigns a specific SKU to a product. However, different distributors and retailers will use other codes to represent the exact product externally. Figure 3-12 shows an example where, on the left, a simple star model is created to analyze what the company sells to a store and, on the right, what the store sells to a consumer.

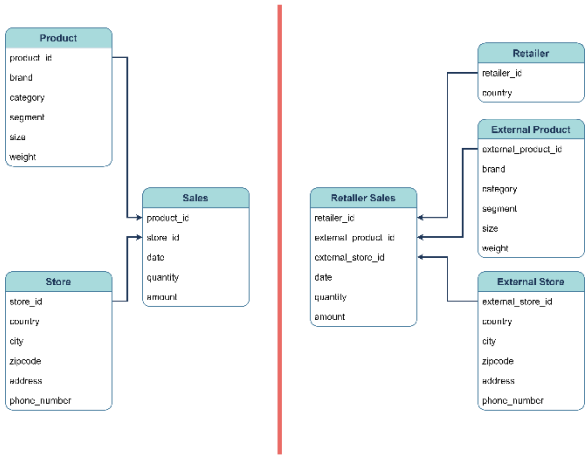



Figure 3-12 – Depiction of incompatible models between internal and external data

The most common mechanism of identifying products in the global market is the EAN/UPC; however, not every retailer/distributor will share this code when providing sales information by product. Maintaining a product code mapping without the proper tools for local markets will be expensive and complicated. Product code incompatibility is the most common, but the issue still stands for core domains like customer, brand, etc.

Currently, when multiple data teams require data from the source systems, they need to set up their own data acquisition and processing pipelines due to the inability of the main data platform to provide access to data in raw/unprocessed data. During the execution of the pipelines, this data is destroyed, and only the final processed data is kept for consumption.

### 3.3.7. Metadata and Lineage

Today, information is scattered between different engineering teams, long-standing members of the local teams, and outdated PowerPoint presentations, making it hard for new members to use the data quickly. With the current architecture, authorized users are given access to Azure Analysis Services, where dozens of tables and hundreds of columns exist without a single description. There is no central repository where metadata and lineage information are easily accessible by different teams. Hence, users resort to opening support tickets to the central team to answer simple questions like:

- How was this column transformed from source to the Analysis Services cube?
- How is this metric computed?
- What is the refresh period of this table?

# 4. METHODOLOGY

## 4.1. PURPOSE

The organization is transforming towards a digital-first approach, emphasizing direct consumer engagement through digital native products and leveraging vast amounts of data from internal and external sources. As the business landscape evolves, the company recognizes the imperative of staying ahead. This project is a testament to this vision, developing a cohesive data architecture to re-work all data sources, ensuring they are standardized, clean, and interoperable.

Understanding the challenges posed by data fragmentation, often scattered across legacy systems, this project aspires to be the solution where raw data transforms into actionable intelligence. Its core objective goes beyond mere data integration. It seeks to facilitate the swift creation of business-centric products that realize diverse use cases like demand forecasting, predictive maintenance, and revenue growth management. The goal is to unlock untapped business potential by placing the company at the forefront of innovation in the data space.

This initiative streamlines processes like data acquisition, cleansing, security, aggregation, and the integration of artificial intelligence. Such focused support ensures that teams can prioritize delivering unparalleled value to the business, free from the overhead of constructing and managing their platform infrastructure.

As described in section 2.2.1, the project underlines its commitment to creating a sustainable and future-ready data ecosystem by adhering to the FAIR data principles. Ensuring that data is Findable, Accessible, Interoperable, and Reusable paves the way for a culture where data-driven insights become the norm.

Moreover, the vision is to replace existing data & analytics solutions and reinforce its role as a cornerstone in the company’s digital evolution. While its primary mission amplifies the productivity of product teams, its overarching vision seeks to diminish operational costs significantly.

The project is a technological endeavor and a holistic approach to reshaping the company’s digital narrative. It embodies the company's dedication to innovation, efficiency, and delivering unmatched value to its consumers and stakeholders.

## 4.2. THE NEW ORGANIZATION

Following the paradigm of the Data Mesh, section 2.2.3, the organization created dedicated teams to support each data product and several central platform teams, which lay the foundations and tooling necessary for the entire platform. As seen in Figure 3-1, 5 teams were created, each with responsibilities within the platform.

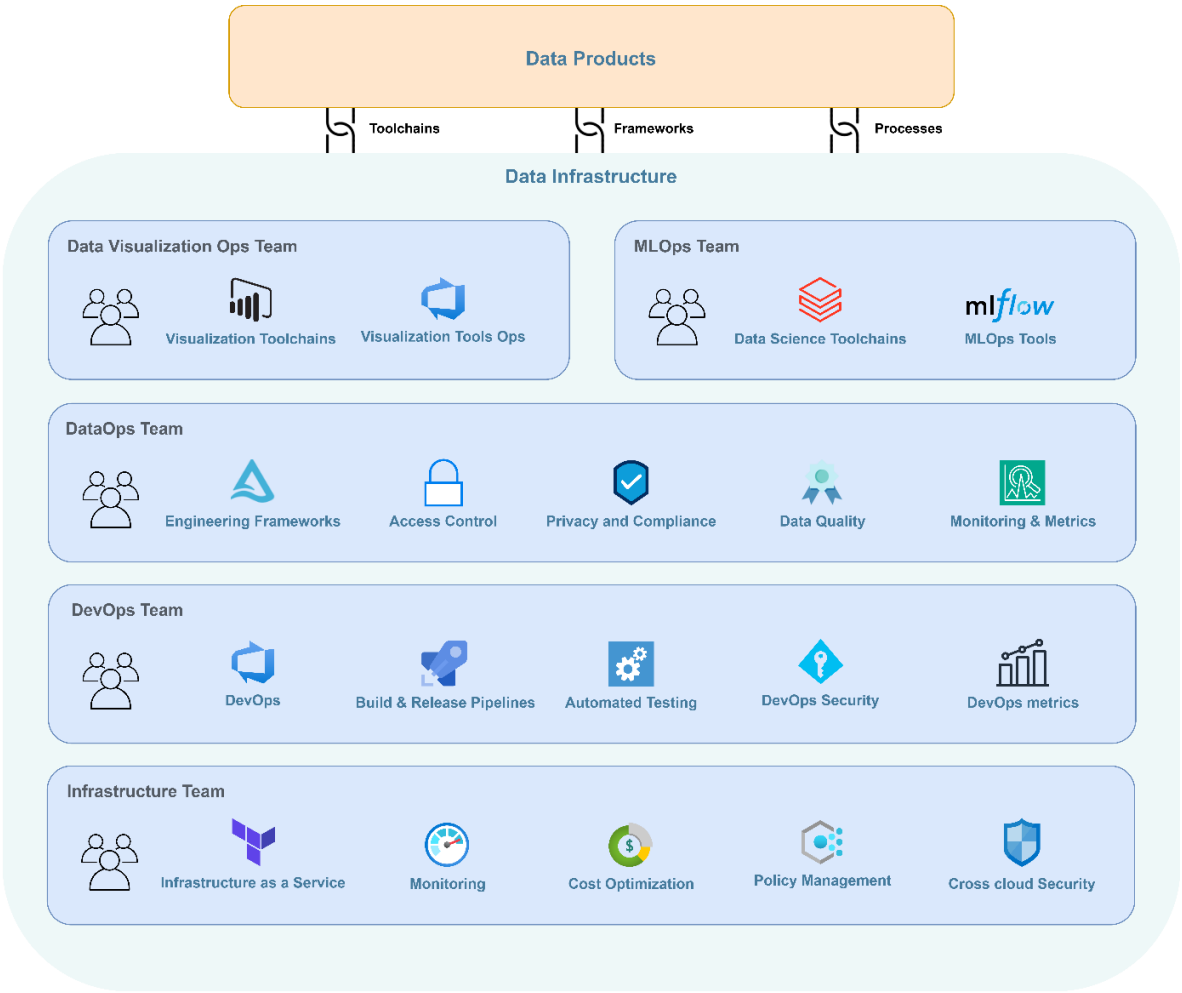



Figure 4-1 - Organization of the Data Platform team into sub-teams

Implementing a Data Mesh involves several specialized teams, each with distinct responsibilities.

Data Visualization Operations team is tasked with effectively managing visualization tools and resources. They handle the definition and management of PowerBI Workspaces, ensuring that the infrastructure for data reporting is robust and well-organized. Additionally, they monitor and optimize PowerBI Premium Capacities to maintain high performance and cost-efficiency. The team also establishes patterns for report creation, optimizing user queries for better performance and user experience. They are responsible for managing the refresh schedule of reports to ensure data is current and implementing Continuous Integration and Continuous Deployment (CI/CD) practices for reports to maintain the agility and reliability of reporting systems.

Machine Learning Operations team provides foundational support for the organization's Machine Learning initiatives. They offer a structured framework that includes the selection of approved libraries and tools necessary for developing machine learning models. This team is involved in the training and ongoing management of these models. They define the MLOps processes to ensure standardization and best practices across machine learning projects. They

are vigilant in monitoring production models to quickly detect and address data drift, ensuring model accuracy over time.

Data Operations team ensures the smooth operation of data pipelines. They monitor these pipelines, which acquire, conform, cleanse, and aggregate data, and communicate any failures to the product teams. The classification of incoming data into the platform and the security of sensitive data, particularly to meet regulatory requirements, fall under their purview. They enforce data retention periods, audit data access, control development team access, and synchronize access policies across technologies like Databricks, PowerBI, and Dremio. Moreover, they define the Data Quality framework, expose Data Quality metrics, and strive for cost, performance, and reliability optimizations within the data pipelines.

The DevOps Team maintains and optimizes the DevOps processes across the platform's infrastructure. This includes managing source control, builds, automated testing, and release pipelines. They ensure all teams, including proper resource access and the use of key vaults, uphold security best practices. Monitoring these DevOps processes aims to optimize the pipelines for performance, costs, reliability, and simplicity.

The Infrastructure Team provides critical support by delivering Infrastructure as a Service, IaaS, and Platform as a Service, PaaS, for Azure and Google Cloud Platform environments. They provision resources using Infrastructure as Code tools like Terraform and maintain Azure policies. Their responsibilities include ensuring all infrastructure resources comply with regulatory requirements, maintaining secure virtual networks, and managing Disaster Recovery protocols. They focus on optimizing the cost and performance of resources and provide reports on crucial infrastructure metrics such as availability, resource utilization, and time to restore services.

These five teams set the foundation to enable all of the Data Product and Use Case teams to operate without constraints in a performant, secure, and compliant fashion.

## 4.3. DATA DOMAINS

To provide a set of standardized models independent of any data source, 6 data domains were created (Figure 4-2): Marketing, Sales, Supply, Finance, Global Functions, and Master Data. Contrary to the paradigm of the Data Mesh, section 2.2.3, the Master Data domain was created. It will be maintained by the central IT team to create the interoperability defined in the FAIR principles, section 2.2.1. This requires extensive work in every single domain to adopt and model their data in accordance with the Master Data dimensions. Still, it will allow the re-utilization of data and the capability to easily mix and match data from different domains by utilizing the pre-established relationships.

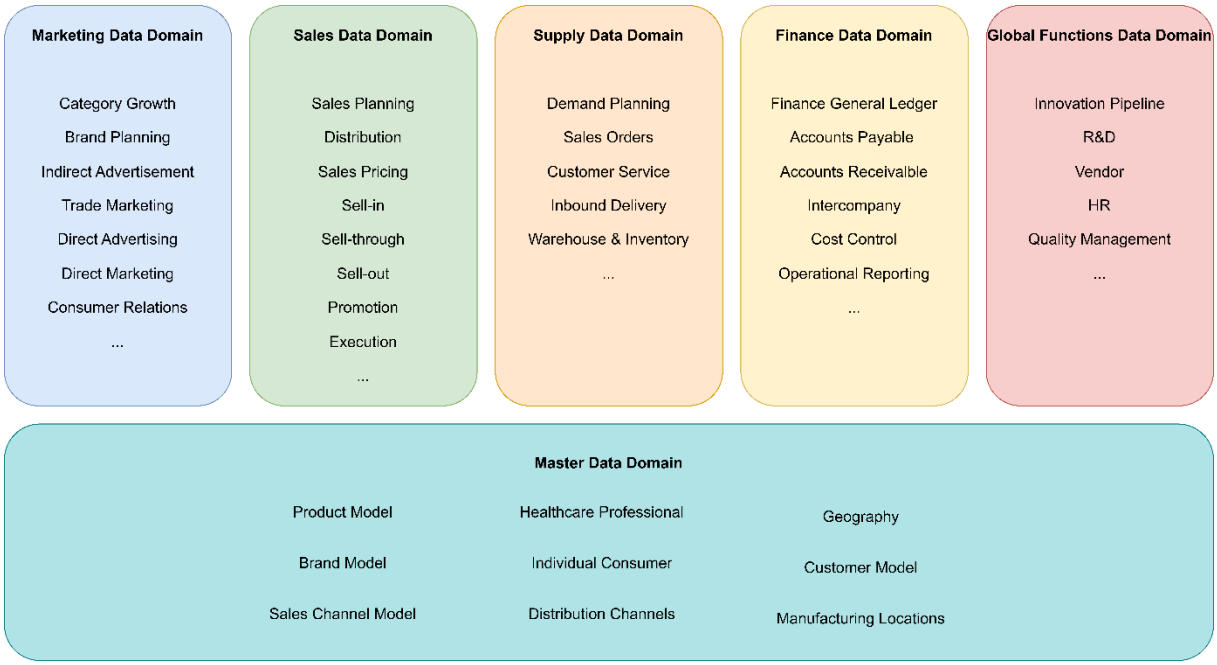



Figure 4-2 - Data platform domains

## 4.4. HIGH-LEVEL ARCHITECTURE

A comprehensive high-level architecture has been developed in response to the challenges delineated in the preceding sections. This framework aims to outline the selection of technologies for pivotal components, as depicted in Figure 4-3. These components include the data architecture, data processing engine, data storage format, data catalog, storage solutions, cloud service provider, querying engine, and visualization tools.

With the advantages mentioned in Section 2.1.1, our data strategy has adopted the Lakehouse architecture. Herein, data will be retained within a data lake in Delta format, leveraging the benefits outlined in section 2.1.1.

The data platform employs two cloud services to manage different datasets: Microsoft Azure is utilized for customer-centric data, while Google Cloud Platform is designated for consumer-centric data. This bifurcated cloud strategy will be maintained to minimize migration efforts for various teams across the organization.

Since Databricks is a familiar environment for all teams and is supported by Azure and Google Cloud, it has been selected as our data processing engine. Databricks offers rapid data ingestion from diverse sources and is intricately linked with the Delta storage format, enhancing its utility for the platform's needs.

Dremio (Dremio | The Easy and Open Data Lakehouse Platform, n.d.) has been chosen as the querying engine for the reporting and exploration layer to facilitate user access to data across different cloud services. Dremio is capable of federating queries between cloud services, thereby establishing a cloud-agnostic layer that allows users to access data without the need

to understand its physical location. To reduce costs with cloud egress, Dremio (deployed in Microsoft Azure) is capable of caching frequently run datasets/queries as reflections (*Accelerating Queries with Reflections | Dremio Documentation*, n.d.), allowing for data in GCP to be fetched less extensively. It thus serves as the organization's unified source of truth, regardless of individual user roles.

PowerBI has been entrenched as the organization's standard visualization tool due to its widespread user adoption over several years. In light of Dremio's native integration with Power BI (*Power BI + Dremio = A Seamless BI Experience for Cloud Data Lakes | Microsoft Power BI Blog | Microsoft Power BI*, n.d.), it will continue as the visualization tool of choice.

At the time of this report, the selection of a data catalog is still under evaluation, and negotiations are ongoing with various providers. For this document, Alation will be referenced as the prospective catalog solution.

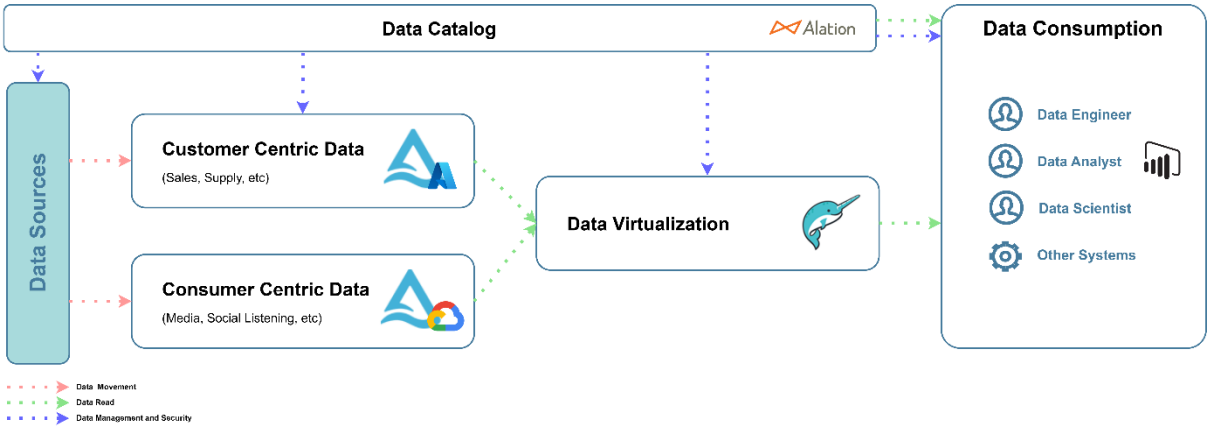



Figure 4-3 - High-Level Architecture of the data platform

## 4.5. LOGICAL ARCHITECTURE

Within the scope of the Data Mesh paradigm, section 2.2.3, the new data platform architecture seen in Figure 4-4, is designed with three critical and distinct layers: the Bronze Archive, Domain Models, and Use Cases. Each layer is tasked with specific roles that collectively strive to streamline data management, enforce security, bolster resiliency, reduce redundancy, and optimize costs.

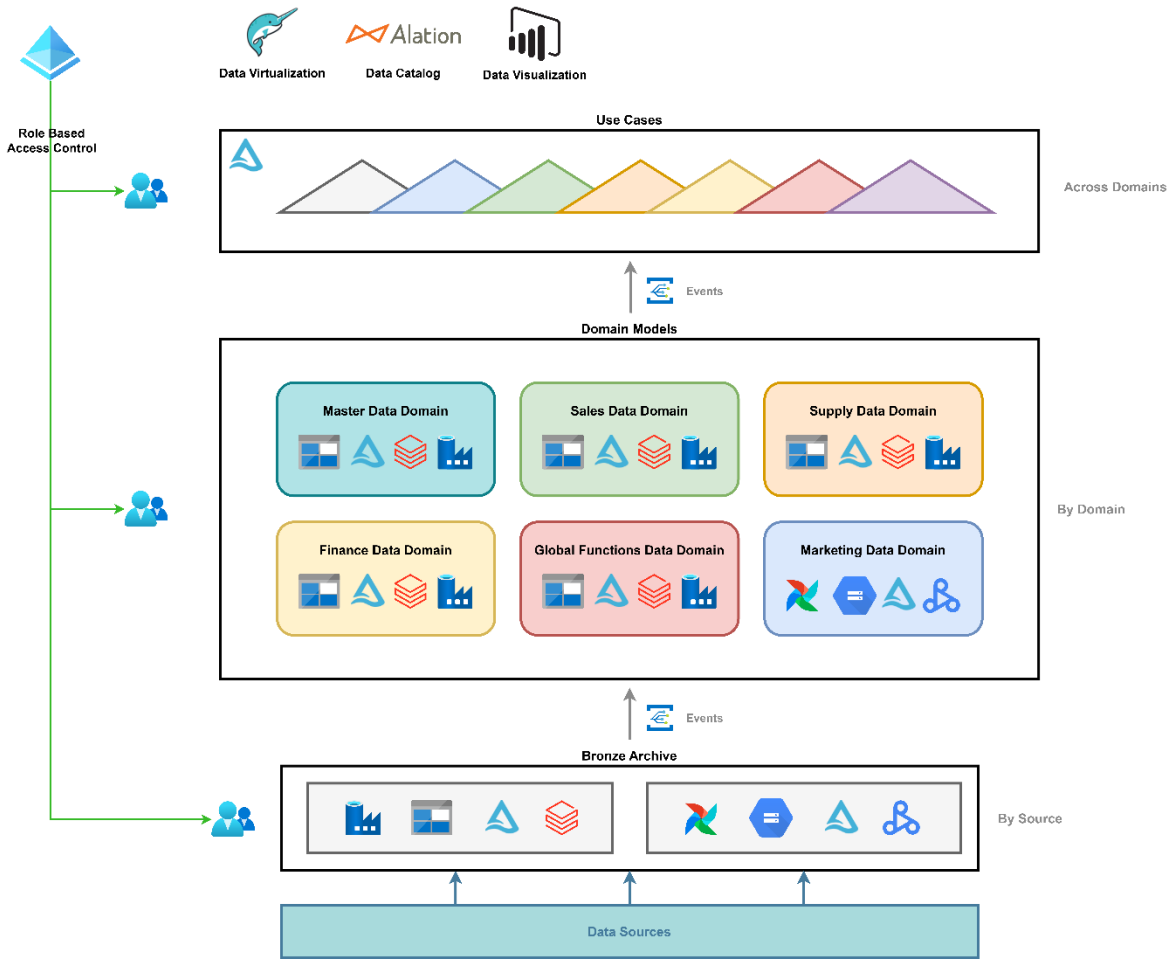



Figure 4-4 - Logical Architecture of the data platform

Each layer works as its own entity and is managed by different teams. To propagate the data loads across layers and maintain each team's independence, a notification system was developed using Azure Event Grid, where a notification is sent to the grid when each process is complete. These notifications can be detected within each team infrastructure and trigger new processes.

The layered architecture of the data platform is a carefully orchestrated arrangement, with each layer serving as a step toward achieving a cohesive, secure, and cost-effective data management system. By addressing data redundancy, security, and access management in a structured manner, the architecture not only meets the current needs of the business but also stands ready to adapt to future demands, thus empowering our teams with a resilient and efficient data solution.

### 4.5.1. Bronze Archive

At the foundation, the Bronze Archive is the bedrock of data consolidation. It acts as a gateway for all other pipelines/processes to read data from and creates a single source for all the Data Domains to operate from. In the past, each team would recreate data ingestion pipelines to fetch data from the sources, which caused excessive data redundancy and wasted computing

resources. The archive preserves the data in its original form, conforming to the source's table and column structures, allowing all teams to reuse it.

Furthermore, the Bronze Archive keeps an exhaustive record of current and historical data, organized within a partition structure of country, year, and month for effortless historical data retrieval. The security of these data sources is paramount, entrusted to the robustness of Azure Active Directory, AAD, for identity management and Azure Role Based Access Control, RBAC, roles for authorization, ensuring that only permitted users and processes have the necessary access.

Cost optimization is achieved through employing the Delta storage format, with high compression rates, and creating a single process within the company to fetch and store source data, eliminating several of the same processes across different projects.

### 4.5.2. Domain Models

The Domain Models layer hosts a set of standardized data models across various business domains. These models stand independent of specific data sources, allowing use-case products developed in one country to adapt to other markets seamlessly. The models are interconnected by a common Master Data Domain, thus facilitating complex data products that can draw data from multiple domains without additional manipulation, focusing instead on value addition.

This layer is instrumental in granting business data stewards the capability to easily maintain and evolve data relationships, aligning with future use cases. It strikes a balance, providing centralized control over the Master Data Domain to the central data community while maintaining agility within the Business Domains.

Domain models are locked down with Azure Active Directory and RBAC roles, with direct access restricted to Azure Compute Resources and data engineers, ensuring granular security measures are in place.

### 4.5.3. Use Cases

The Use Cases layer, where analytical models are developed, encapsulates the intricacies of thousands of domain tables and relationships into standardized analytical tables primed for reporting. This layer reduces the domain complexity into a more manageable form for the business, creating views and aggregates engineered to be easily understood by end users.

Complex business logic is embedded within Databricks notebooks and executed as part of the analytical pipeline processing of each use case. This hides the inherent complexity from end users and provides the advanced analytics that the business requires.

The security protocol at this layer mirrors the previous layers, with end-user access through Dremio, where row-level security prevails, backed by Azure AD and RBAC roles for compute resources and engineering access.

## 4.6. UNIFIED TRANSFORMATION LAYER

Throughout the multi-year development process involving various teams and employing a range of technologies, our codebase associated with all datasets has been organized into three distinct logical layers:

- Transformation layer primarily consists of pySpark/SparkSQL code executing within Databricks, tasked with refining raw data into a format suitable for consumption by the data warehouse.
- Storage layer involves stored procedures within the data warehouse that ensure data is stored accurately and efficiently.
- Serving layer, this layer comprises SQL views established atop the data warehouse tables to facilitate data retrieval.

Historically, as teams have developed familiarity with specific platform segments, the boundaries between these three layers have become increasingly blurred. On occasion, processes ideally handled within the transformation layer have been erroneously managed within the serving layer. For instance, there have been instances where teams have modified data during the storage phase, bypassing the designated transformation procedures.

Such practices have necessitated significant efforts to consolidate these layers into a cohesive framework for data transformation and preparation, as illustrated in Figure 4-5. This consolidation is imperative for the analysis and revision of scripts and codes written in a variety of programming languages.

The advantages of this consolidation are evident. Streamlining the three layers into a single, coherent layer will enhance the interpretability of transformation steps. This, in turn, simplifies the tracing of data lineage to the data catalog, improving overall data governance. Furthermore, it expedites the implementation of modifications within the platform, as it minimizes the extent of alterations required to incorporate new features. This streamlined approach is a strategic move to enhance the efficiency, agility, and reliability of our data management processes.

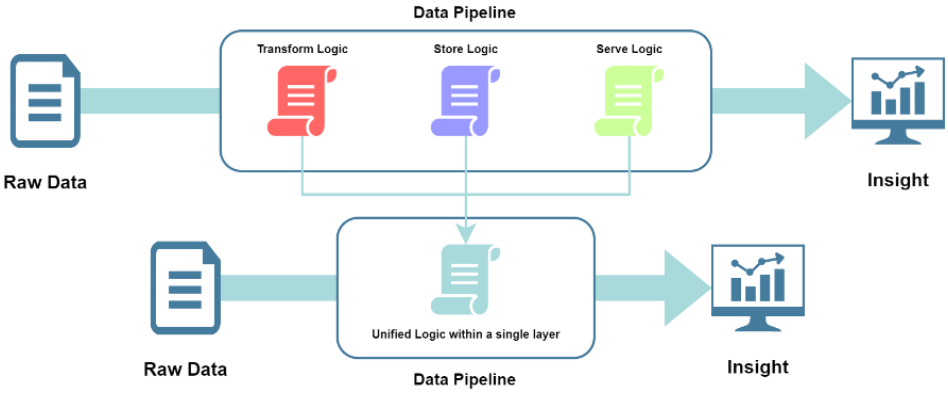



Figure 4-5 - Unified transform layer

## 4.7. Event Based Notifications

A system-wide notification system was designed to create isolation between all the layers of the data platform. Using Azure Event Grid (*Overview - Azure Event Grid | Microsoft Learn*, n.d.), a message will be sent between each layer, signaling that the data from the previous layer is ready to be consumed. Any relevant data product or use cases will be able (with the necessary RBAC permissions) to subscribe to a specific topic and execute pipelines/processes when said message is sent. The event contains a message containing the source, country, period updated, and any other relevant information necessary for subsequent executions.

This process brings flexibility to the platform since it allows the different teams to schedule their own processes without relying on previous layers' teams to include invocations in their processes. Within Azure, Event Grid integrates natively with several services (*Azure Event Grid Event Handlers - Azure Event Grid | Microsoft Learn*, n.d.), allowing for automatic execution of Azure Functions (*Azure Functions – Serverless Functions in Computing | Microsoft Azure*, n.d.) (which, being a serverless code solution enables the teams to to trigger any job) and Azure Data Factory Pipelines, as seen in Figure 4-6.

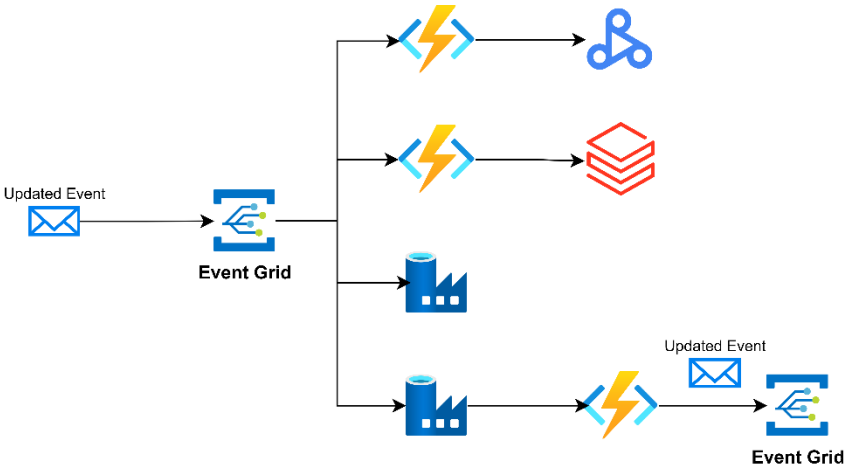



Figure 4-6 - Propagation of events throughout the platform

## 4.8. BRONZE DATA INGESTION

Within the data platform, two types of sources will exist: internal, where the platform manages the extraction process, and external, where files are dropped into cloud storage by third parties for further processing.

For internal datasets, data extraction is managed by Azure Data Factory, so there is no need to perform validations against the integrity of the files being ingested.

To simplify the ingestion of new data sources, internal or external, a bronze loading framework was designed to serve as a guide to ingest the data in the desired format in the Bronze layer. Data can be extracted from different sources, be in different formats, with different schemas, different primary keys, mechanisms to handle duplications, and received as full or in increments, but, in the end, the goal is to create a bronze layer which is stored within the data lake in Delta format and, which contains the entire history of the dataset.

The framework is an application built in pySpark, running in Databricks, which makes use of JSON configuration files for each of the datasets and loads the data into the target table using Databricks Auto Loader feature (*What Is Auto Loader? - Azure Databricks | Microsoft Learn*, n.d.). The JSON template can be seen in Figure 4-7.

```json
{<entry_name>: {
    "data_type": <string>,
    "primary_keys": <jarray>,
    "not_nulls": <jarray>,
    "read_method": {
        "folder_location": <string>,
        "sep": <string>,
        "read_type": <string>,
        "header": <boolean>,
        "columns": <jarray>
    },
    "write_method": {
        "type": <string>,
        "database_name": <string>,
        "table_name": <string>,
        "details": {
            "table_partitions": <jarray>,
            "replace_columns": <jarray>
        }
    }
  }
}
```

Figure 4-7 - Bronze Framework configuration JSON template

For each entity in the JSON configuration, the description and logic are presented below:

- *entry_name*, this entity defines the name of the dataset
    - *data_type*, used to define which type of dataset is being loaded. The possible values: *fact* or *dimension*.
    - *primary_keys*, an array containing the column names defined as primary keys. This value is used to perform the primary key validation on the dataset. If the dataset does not contain primary keys, this entry should be marked as *null*.
    - *not_nulls*, an array containing column names defined as *not null* constraints. This value is used to identify *null* values on specific dataset columns. If the

dataset does not contain *null* value constraints, this entry should be regarded as *null*.

- *read_method*, an object containing the following entries:
  - *folder_location*, folder location on the data lake where the raw data is located.
  - *sep*, separator character to be used when reading text files.
  - *header*, boolean value to indicate if header should be read.
  - *columns*, array to indicate if columns should be renamed or if schema on read is allowed.
  - *read_type*, method to be used when reading the file from the data lake. The possible values are *folder* (all files within the folder will be read) or *files* (specified files sent to the application as parameters will be read).
- *write_method*, an object containing the following entries:
  - *type* indicates which type of write method should be used when writing the data on the data lake. The possible values are *overwrite*, *merge_keys* or *replaceWhere.*
    - *overwrite*, dataset will be replaced with the new data.
    - *merge_keys*, new records will be inserted, and existing records will be updated based on the primary keys.
    - *replaceWhere*, records matched by the columns defined on the *replace_columns* entry will be replaced (*Selectively Overwrite Data with Delta Lake - Azure Databricks | Microsoft Learn*, n.d.).
  - *details*, this entry is only valid if the type of method is *replaceWhere*. If the method is *overwrite* or *merge_keys,* this entry should be *null*.
    - *table_partitions*, an array containing the columns to partition the data on the data lake.
    - *replace_columns*, an array containing the columns to build the *replaceWhere* method.

For external data, there is a need to validate every file received to guarantee that further processing will occur without issues (either by corrupted files, which will cause an error in the process, or by invalid data, which will cause data quality issues). To achieve this, a validation system was created where, for each file received, a series of steps are executed to guarantee its validity.

In Figure 4-8, this system is represented as a diagram in which the process starts with the upload of a file into an Azure Blob Storage and, making usage of the Azure Functions bindings, an API is used to validate the name of the file against a series of pre-approved regular expressions. Using regular expressions and with the correct format followed, this step will identify the dataset, country, and period for the specific file. If the file is not identified, it is dropped into quarantine storage, and the process is interrupted.

Upon the identification completed, the Azure Function will trigger the Validation API, which is an application written in C# running in an Azure Web Application (Web App Service | Microsoft Azure, n.d.); the file is read and validated against a list of specified rules. The rules applied will vary by each dataset but can include validations against valid primary keys, duplication of primary keys, schema, encoding, file format, and date formats. If the file violates any of the specified rules, the file will be dropped into quarantine, an email will be sent to the relevant stakeholders, and the process will be interrupted.

Upon successful validation, the file is sent for further processing by the bronze pipeline. When the data load is completed, a notification is sent to Azure Event Grid to notify downstream processes that data is ready to be used.

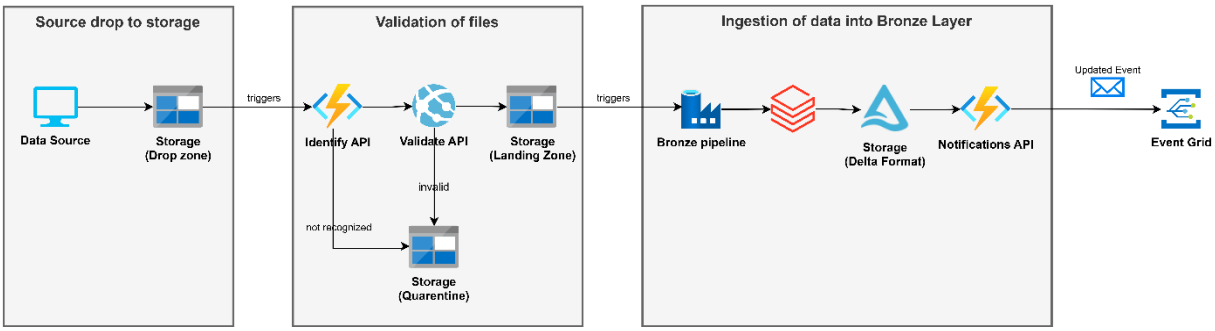



Figure 4-8 - Bronze data ingestion for files received from external providers

# 5. USE CASES

## 5.1. FINANCIAL GENERAL LEDGER

A financial ledger is a time-ordered log of a company's financial dealings, methodically capturing every transaction. This organized tally includes the ins and outs of the company's finances, like earnings, expenses, and what it owns. It is into specific accounts, helping keep a close watch on various financial activities.

The ledger isn't just a record-keeping exercise; it's pivotal for financial navigation and strategy. It lays out all the transactions in detail, making it easier to spot any inconsistencies. Plus, it's the go-to source for putting together the key financial snapshots—balance sheets, income statements, and cash flow statements—that give stakeholders, from investors to executives, the company's financial status.

The financial ledger provides a comprehensive look at a company's financial transactions, tracking everything of value that the company deals with. It’s indispensable for informed financial management and decision-making, offering crucial insights to those with vested interests. More than just a record, it ensures the financial data's precision and clarity, painting a reliable picture of the company’s economic health.

### 5.1.1. Current Limitations

As discussed in section 3.2 of this report, in 2019, the significant improvement introduced to the platform was the migration of IBM’s DataStage to a cloud-first approach with Azure Data Factory as the primary extraction tool. However, the tool's purpose was not only the extraction of the data from the source systems but also the preparation and final upload of the files to an Azure Blob Location from where the main data platform would ingest the files upon schedule.

However, there is a difference between the granularity at which both processes operate. Data extraction from the source systems and pre-processing occurs at a country level, while the data processing pipeline runs at a regional level (a collection of countries).

This introduced the technical challenge of designing this feature while also causing minimal impact on the processing jobs of the main data platform. As such, the final design of this feature was to introduce an initial processing step running in HDInsight, replicating the old ETL jobs within DataStage, which would save the data in table format in the same Hive Metastore as the main data platform with country granularity. The impact was minimal since the only relevant modification was related to the data reading, not the processing.

Another recurring concern among the team is the translation between the accounts and the financial performance indexes. At each end of the quarter, this mapping is subject to change, and since the processing of the data is ephemeral, no data is saved in the system besides the

final consumption layer. This results in massive data reload for the entirety of the reporting period (usually last two fiscal years plus year to date). This process is expensive in both cost and time, requiring coordination from the Support and teams responsible for the internal data systems.

From a reporting perspective, the above constraint of having only the processed data available also blocks users from reporting at the account level. This prevents users from performing analytics on the type of account contributing the highest to a specific financial performance index and underperforming.

The general ledger, in essence, shows the company's financial health and, as such, is a tightly controlled dataset security-wise. Regular audits are performed to assess the users with access to view this dataset since no user with access is allowed to exchange company shares in the stock market. As section 3.3.1 shows, the current security model in the platform is subpar and is usually a stress point in these audits since users from one country can view and use data from other countries, provided they are within the same region.

### 5.1.2. Implemented Design

As stated in section 5.1.1, the three challenges regarding the Financial Ledger dataset are pipeline cohesion, ephemeral processing, and security.

The lack of pipeline cohesion originates from the constant evolutions of the platform with different technologies used in each stage of the transformation from raw data to reports, where each technology would require its own component in the pipeline with its own business logic applied.

The process followed was analyzing every single component of the current pipeline, understanding the necessary transformations and business logic applied to create a holistic view of the requirements for the source raw data to make it available as-is in the reporting layer.

This created the opportunity to integrate the current component of the pipeline, which replaced DataStage into a single, unified pipeline containing all components: preparation for processing, processing, and data warehousing with the same country granularity across all components, as seen in Figure 5-1.

Applying the Medallion Architecture, the concept at which the team arrived is of a bronze layer containing all of the data coming from the ERP system, a silver layer (the data product in the Domain Models) with cleaned data composed of all the accounts in the organization and, a gold layer containing the FPI’s applying the mapping table currently available, as seen in Figure 5-2.

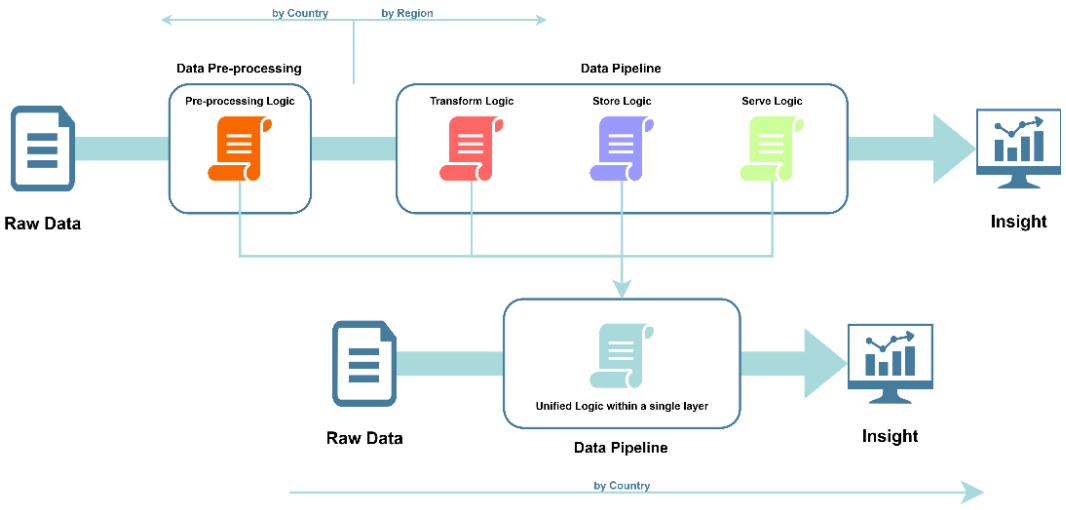



Figure 5-1 - New unified design for the Financial Ledge dataset

In these three layers, the history is preserved, and data is merged into the tables, allowing future requests to recompute the performance metrics with new mapping tables by recreating the gold layer and re-using all the data in the silver layer. Another possible approach was to use views in Dremio, which would compute the performance metrics on the fly, but the performance penalty imposed by this approach would negate any benefit compared to every quarter recomputing just the final layer. Furthermore, allowing Dremio to read data from the silver layer will enable users to create reports based on accounts.

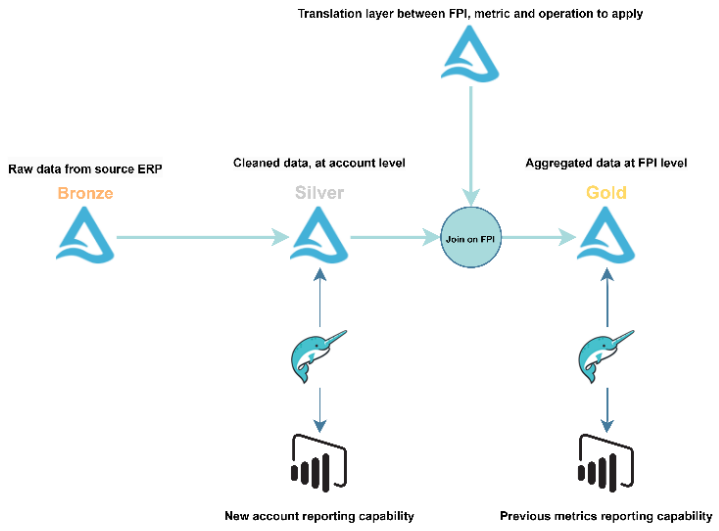



Figure 5-2 - Medallion Architecture applied to the Financial Ledger dataset

Adopting the Lakehouse architecture and storing all of the data within a single data lake requires a deeply integrated security mechanism to grant access to the data. While Azure Analysis Services cubes would not allow row-level security at a country level without impacting performance, users were still limited by data within their geographical region. On the contrary, within the Lakehouse, users would be granted to the data of all regions.

Dremio does not support row-level security as an out-of-the-box feature; as such, a method was applied where the user-exposed views are coupled with filters that identify if the user is a member of a specific Azure Active Directory security group (which are created on a by dataset and by country basis). Using the built-in IS_MEMBER function (*IS_MEMBER | Dremio Documentation*, n.d.), a user querying this view will be identified and will only have access to data of the granted countries. The following DDL contains the sample logic to create the user-exposed views where, for each row of the source table, the user is validated against the specific security group of the country.

```
CREATE VIEW finance_ledger
AS
SELECT
        country,
        year,
        fiscal_month,
        date,
        metric_1,
        metric_2,
        metric_3
FROM dataplatform.gold_finance_ledger
WHERE IS_MEMBER( CONCAT ( 'DataPlatform_' , country ) )
```

Another concern was the need to expose all datasets of a particular country within the Azure Analysis Services cubes, even if the user only used a specific dataset. Dremio supports object-level security coupled with Azure Active Directory security groups.

As such, users who require access to a specific dataset for their country are inserted into two security groups: the dataset-specific one, which will present the dataset in Dremio to the user, and the country-specific one, which will allow the user to see data within the dataset for their country, as seen in Figure 5-3.

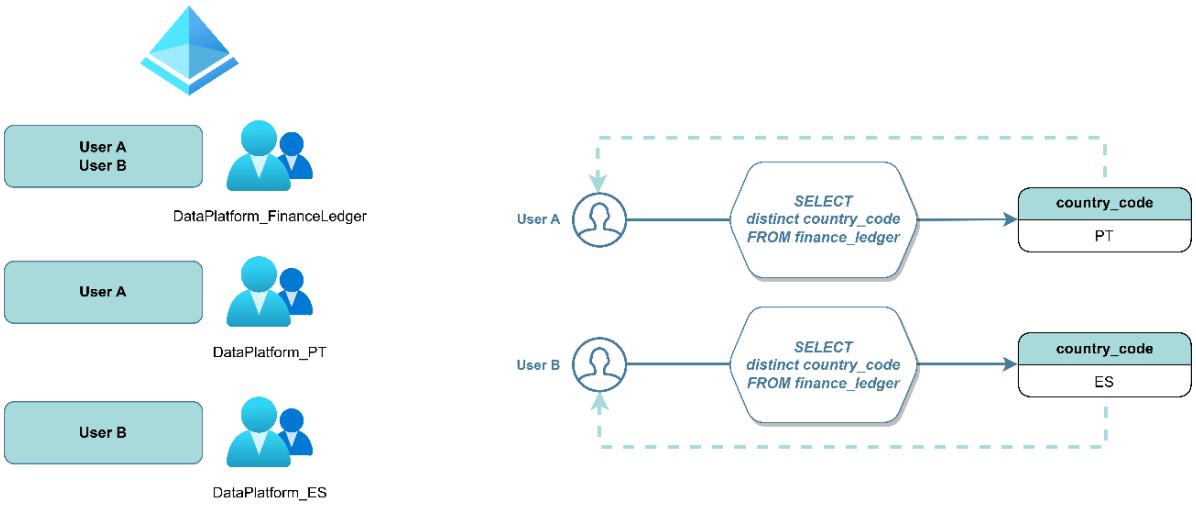



Figure 5-3 - Security model applied to the Finance Ledger dataset

### 5.1.3. Results

This dataset was the first chosen by the leadership and the team because of the opportunity to generate the most significant impact with fewer changes applied. As a simple but essential dataset, the flexibility offered by a Lakehouse approach was critical in the overall project's success.

From an operations perspective, simplifying the entire pipeline led to streamlined processes with fewer moving parts, which enabled our support team to debug better and identify issues faster while also reducing the execution time of the entire pipeline by around 34%. At the same time, the new re-computation of the financial metrics process transformed into an activity of a few hours from an activity that would require a few days and cross-team collaboration to achieve.

From the perspective of the users, all of their reports kept operating as expected when switching the source from the previous Azure Analysis Services to the new Dremio connection, and they are now enabled to perform more advanced analytics at an account level in a more secure manner via the new security process in place.

As a result, the migration was a success and fueled the future migrations into a Lakehouse architecture.

## 5.2. SELL-OUT

The product's journey to the customer is tracked within the company via three different sale types: Sell-in, Sell-through, and Sellout. As seen in Figure 5-4, the product's journey starts at the warehouse, where it is sold to a distributor and represented by a Sell-in sale. This distributor will distribute the products along a range of stores and is represented by a Sell-through sale. When the user visits a store and buys a product, this is considered a Sell-out sale.

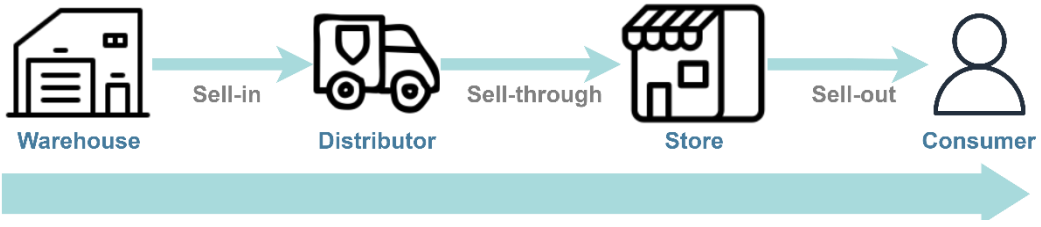



Figure 5-4 – The product journey until it reaches the customer is tracked in sales numbers

The Sell-out can be used to analyze consumer behavior and make informed decisions about inventory management, pricing strategy, new category development, and marketing efforts. It can also be used to identify trends in sales over time, such as seasonal fluctuations or changes in consumer preferences. Additionally, by analyzing the sales data in combination with other data, such as customer demographics or marketing campaign results, the organization may gain insights into which strategies are most effective in driving sales. Overall,

Sell-out is a valuable tool for making data-driven decisions and improving the business's performance.

### 5.2.1. Current Limitations

To understand the current limitations of the Sell-out dataset, four topics need consideration: standardization, expandability, availability, and duplication.

Regarding standardization, the further away the product is from the warehouse, the harder it is to acquire clean and standardized data. Sell-in data is registered and made available to the data platform from the internal ERP systems, resulting in a standard template across all markets. Sell-through data acquisition varies by market; specific markets have distributors owned by the company, but in others, distributors are external companies that will stock stores and share the data accordingly in different templates. However, due to the rarity of this situation, Sell-through is considered a standard template across markets. In contrast, Sell-out data is acquired independently from every retailer where the company sells products, and each will expose their data in different formats and templates. This creates the challenge of defining a standard template to be used upon data ingestion. Currently, one local market is acquiring and locally processing their data before exporting it for ingestion. Every other market is buying this service from a service provider, which, besides being costly to the company, has a long delay in data sharing.

The standardization issue will lead us to the topic of expandability. Data coming from the retailers varies in how the same data is represented and the number of metrics and attributes that vary by retailer, which can, have, and will evolve in the future. In the past, retailers were cautious with the data shared. They would mostly stick to metrics like amount/quantity sold per day and store, but, currently, there are retailers exposing hundreds of new metrics like inventory (which is helpful to analyze which stores will need product in the future), average basket size, number of page views per day (in online retailers), etc. The current dataset format in the data platform started becoming a bottleneck in the business aspirations to consume and analyze all possible metrics.

The current template, Table 5.1, defines the amount and quantity columns as standard metrics and gives some flexibility to include generic named metrics where each country will populate with different fields from different retailers. Today, the template allows for 20 generic measures, but this number is already an increase from the past (which allowed only 5), and requests to increase it occur frequently. With the rate of new metrics provided by retailers, this limitation and the workaround of increasing the number of metrics as columns is not sustainable.

Table 5.1 – Schema of the current Sell-out dataset

| Column | Data Type |
|---|---|
| date | date |
| country_code | string |
| retailer_id | string |
| store_id | string |
| product_id | string |
| attribute_1 | string |
| … | … |
| attribute_5 | string |
| quantity | integer |
| amount | decimal |
| measure_1 | decimal |
| … | … |
| measure_20 | decimal |

Today, there is a fixed daily schedule to execute the data pipeline for the Sell-out dataset for each country. Unfortunately, the data is available at different intervals across retailers, and, for the same retailer, the data is not delivered at the same time consistently. This problem is exacerbated when dealing with markets that contain multiple time zones. Essentially, the team needs coordination with all the countries to select a timeslot for them where the highest number of retailers have data available so that the data refresh will occur in time for business users to start their day with fresh (but incomplete) data available. This leads to data from retailers being made available after the process begins. Still, due to the high execution time of the pipeline, data sits in waiting to be loaded the following day, causing availability issues.

As described in section 3.3.6, incompatible product codes are the most common root cause of model incompatibility across data products, and the Sell-out dataset is no exception. In dealing with retailer data, the organization is subject to the product codes provided by each retailer, and, as such, a translation layer needs to exist between what the retailer sends and what is exposed for consumption. Today, this translation exists as part of local markets' daily operations, where reports generated by the platform identify new unmapped products/stores, and a new revised mapping file is sent to the platform. However, as described in section 3.3.2, markets are limited to 1 pipeline execution per day, which will cause unmapped data to remain as is for at least an extra day, resulting in late access to new information.

The final topic that needs to be considered is duplication. Due to the importance of the Sell-out dataset, many analytics projects within the organization require access to this data, but

due to the inability of Azure Synapses dedicated pool to provide high concurrency, section 3.3.3, the only option available to the team is to create a copy of this dataset across several resources (including BigQuery). This makes multiple copies of the same data spread across several environments, violating the principle of the Reusability of data, section 2.2.1.

The entire data flow of the Sell-out dataset can be seen in Figure 5-5, where data flows through different platform layers once per day per region and takes, on average, 3.5 hours to process all data per region before data is ready for consumption.

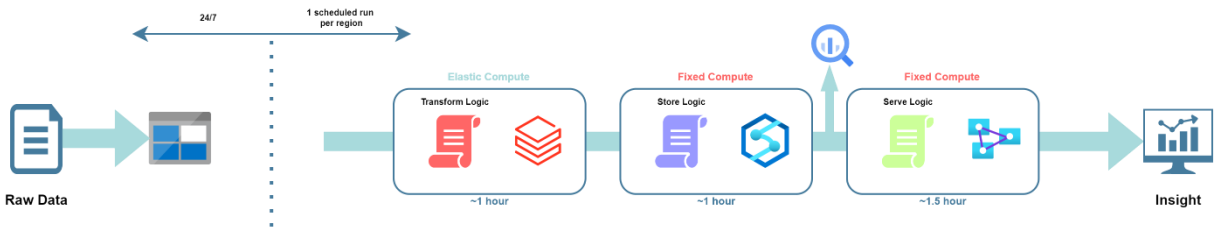



Figure 5-5 – Current data flow for the Sell-out dataset

### 5.2.2. Implemented Design

As stated in section 5.2.1, the four challenges regarding the Sell-out dataset are expandability, standardization, availability, model incompatibility, and duplication.

Today, from example files from retailers, the regional teams have dozens of different metrics that need to be included, but how can the team design such a template that will enable standardization across all markets and create the necessary expandability for the future? Traditionally, such kind of dataset will make use of NoSQL databases where there is the capability of extending the schema as needed to accommodate the additional metrics, but the regional markets would not have the necessary technical skills to fulfill their reporting needs quickly and would require the assistance of the central data team to accomplish their goals. So, the final template design combines traditional relational schemas and a new breed of data types available in Spark. As can be seen in Table 5.2, the template possesses the same core columns of the previous template (to allow an easier migration for the regional teams), but instead of relying on a fixed number of attributes and measures, regional teams can upload as many metrics as needed into the measures field, which will store all data in a key-value pair. This means that, no matter which new metrics or attributes the regional teams need to onboard into the platform, they are free to do it without constraints in this field.

Although all retailers differ in the format and schema of files sent, these specifications are somewhat constant in daily operations, meaning that, for a given retailer, the files sent are of the same format each day with minimal changes. This allows the creation and reuse of a translation from the retailer file to the defined template in section **Error! Reference source not found.**.

Table 5.2 – Schema of the implemented Sell-out dataset

| Column | Data Type |
|---|---|
| date | date |
| country_code | string |
| retailer_id | string |
| store_id | string |
| product_id | string |
| attributes | map<string, string> |
| quantity | integer |
| amount | decimal |
| measures | map<string, decimal> |

As such, a web-based application was created where business users all over the globe can define their translations and reuse them when a new file needs to be updated. When new attributes/measures are made available by the retailer, users update their translations, inserting the new fields in the respective map object within the template and re-uploading the necessary files. This allows for all the current measures/attributes to be uploaded to the platform at will, and with ease, users can expand their current use cases without requiring the assistance of the data engineering team to create additional columns in the database.

Any new file submission from the same retailer will reuse the previously defined translation and issue a new Azure Databricks job, which loads the data in all the layers, making it available for consumption for all users via the Dremio consumption layer.

Data will have two main paths, the web application path and the traditional scheduled pipeline, which will load data for markets unwilling to adopt the web application. Since this approach resembles a mini-batch processing method and optimizations are still possible with the current code, the target for ingestion time is 15 minutes. This will allow markets to quickly upload their data throughout the day and have it readily available for reporting.

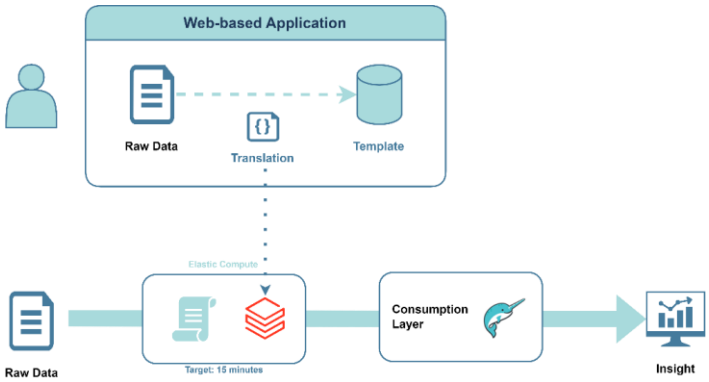



Figure 5-6 – Implementation of the Sell-out architecture within the new platform

Working with the new paradigm of the Lakehouse architecture and its ability to support both batch and streaming data sets and the capability of Dremio to connect directly to the Data Lake without the need to import data into our reporting solution, the solution to the problem came in the form of a web application. Two goals were defined for this application: immediate visibility in the reporting tools and recommendations of which internal product codes to assign.

As such, the workflow of the application is as follows:

1. When the user enters the application and selects the specific retailer to be worked on, the application issues a query to the data to retrieve the unassigned products/stores.
2. For the unassigned products/stores, use the metadata from the retailer, such as size, description, brand for products, and address, zip, and city for stores, to perform a keyword search on the internal product/stores catalog.
3. Present the recommendations to the user, who will select an internal product code/store code from the recommendations or, if none match, input the correct match.
4. When the user is finished, the web application will take the current session's matches and automatically merge the selections into the table of mappings within the data lake.

Since Dremio is querying the data lake directly, when the process is finished, users will have the data automatically in sync with the reporting tool without a need to wait for the subsequent data load.

By design, this dataset will not be duplicated since, within the new architecture, all data is persisted in the data lake and made available for consumption via a single layer. All analytics projects that require this dataset connect to Dremio via API and acquire the necessary data irrespective of which cloud and resources they interact with.

### 5.2.3. Results

Although much work went into fully designing and implementing the revised architecture for the sell-out data, this transformation has greatly impacted the organization and the local teams. The outcome of this revision can be analyzed in 5 fronts:

1. Are the markets using the web application to ingest data independently?
2. Is the measures field being used to grow the number of metrics available continuously?
3. Was the 15-minute ingestion time target goal achieved?
4. Does the mapping tool impact the overall number of products/stores mapped?
5. What are the improvements for projects which consume this data?

The web applications have revolutionized the daily operations of 27 distinct local teams across the globe. With the proper training, the teams could independently create around 2100 translation files, increasing to 250,000 stores covered globally, a 30% increase. Since no cost restrictions for external parties were in the equation, as soon as local teams acquired data for a new retailer, they could quickly create a translation file and start reporting on the data. Entire teams were created within the organization whose sole responsibility is the acquisition (with purposefully built web manipulation bots) and data ingestion into the tool. The assignment of internal codes using the web application lead to some markets, at the moment, having a 100% coverage of all internal codes mapped while spending far less time in this process due to the automatic recommendations.

The expectation regarding the measures field was conservative since, at first, the goal for local teams was to properly transition to using the tool rather than quickly expanding the reported measures. However, two local markets have already started using the capabilities provided and are reporting, in some retailers, around 60 measures.

The ingestion time differs based on the path of the data. When ingesting data through the web application, the processing times, on average, take between 10-15 minutes due to the almost mini-batch approach ingesting a relatively small amount of data. In the traditional scheduled pipeline, the ingestion time was reduced from 1 hour to around 30 minutes in the worst-case scenario (the markets with the highest volume of data). This was achieved by fully migrating the table storage format to Delta (instead of simply parquet), allowing the built-in optimizations and the extensive rework done on the code to be more efficient in both the operations performed and the resources utilized.

So far, two analytics projects that consume sell-out data have fully migrated into using the new cloud-based platform, one in GCP and another in Azure. Both projects have seen benefits, particularly in eliminating staleness in the data. Since then, they have been directly connected to the data lake to retrieve the most recent data and not to a version of it created at a specific time. However, since the data platform is built on top of Azure when data flows to GCP, there is an egress cost that was previously smaller due to the frequency.

# 6. CONCLUSIONS AND FUTURE WORKS

In conclusion, this thesis has elucidated the transformative power and comprehensive potential of the Lakehouse Data Architecture, a hybrid model that combines the strengths of data lakes and data warehouses. This innovative architecture has proven to be a highly effective solution in managing vast, diverse data sets, streamlining data management processes, advancing analytics capabilities, and supporting agile, data-driven decision-making within organizations.

By implementing cutting-edge technologies such as Microsoft Azure, Google Cloud Platform, Databricks, Apache Spark, Delta Lake, and Dremio, the Lakehouse architecture has addressed pivotal challenges in model incompatibility, data duplication, and agility. It has markedly optimized the performance of data pipelines, reduced redundancy, and enhanced the efficiency and security of data operations. The division of the data platform team into specialized sub-teams has further ensured the smooth functioning and reliability of data services.

The thesis has also illuminated the strategic implications of adopting the FAIR data principles, which advocate for data to be Findable, Accessible, Interoperable, and Reusable. Fostering a culture of data excellence and operationalizing the Data Mesh concept presents a paradigm shift toward a decentralized data management approach, potentially revolutionizing the efficiency and agility of organizational data platforms.

Providing valuable insights into the architectural components and operational mechanisms, this thesis is a foundational guide for organizations aiming to navigate the complexities of becoming data-centric. It underscores the pivotal role of the Lakehouse architecture in harnessing data for strategic innovation and positions the model as an adaptable and robust framework for future technological advancements in AI and machine learning.

As organizations continue to confront an ever-evolving digital ecosystem, this exploration contributes to the field of data management. It offers a forward-looking perspective on the ongoing journey towards leveraging data as a cornerstone of business success and transformation.

# BIBLIOGRAPHY


*Accelerating Queries with Reflections | Dremio Documentation*. (n.d.). Retrieved October 3, 2023, from https://docs.dremio.com/current/sonar/reflections/

Alaimo, C., & Kallinikos, J. (2022). Organizations Decentered: Data Objects, Technology and Knowledge. *Organization Science*, *33*(1), 19–37. https://doi.org/10.1287/orsc.2021.1552

Andany, J., Léonard, M., & Palisser, C. (1991). Management Of Schema Evolution In Databases. *VLDB*, *91*, 161–170.

Arif, M., & Mujtaba, G. (2015). A Survey: Data Warehouse Architecture. *International Journal of Hybrid Information Technology*, *8*, 349–356. https://api.semanticscholar.org/CorpusID:44205510

Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., Torres, J., van Hovell, H., Ionescu, A., \L{}uszczak, A., undefinedwitakowski, M., Szafrański, M., Li, X., Ueshin, T., Mokhtar, M., Boncz, P., Ghodsi, A., Paranjpye, S., Senster, P., … Zaharia, M. (2020). Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. *Proc. VLDB Endow.*, *13*(12), 3411–3424. https://doi.org/10.14778/3415478.3415560

Armbrust, M., Das, T., Torres, J., Yavuz, B., Zhu, S., Xin, R., Ghodsi, A., Stoica, I., & Zaharia, M. (2018). Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark. *Proceedings of the 2018 International Conference on Management of Data*, 601–613. https://doi.org/10.1145/3183713.3190664

Arribas-Bel, D., Green, M., Rowe, F., & Singleton, A. (2021). Open data products-A framework for creating valuable analysis ready data. *Journal of Geographical Systems*, *23*(4), 497–514. https://doi.org/10.1007/s10109-021-00363-5

Athey, S. (2018). The impact of machine learning on economics. In *The economics of artificial intelligence: An agenda* (pp. 507–547). University of Chicago Press.

*Azure Event Grid event handlers - Azure Event Grid | Microsoft Learn*. (n.d.). Retrieved October 2, 2023, from https://learn.microsoft.com/en-us/azure/event-grid/event-handlers

*Azure Functions – Serverless Functions in Computing | Microsoft Azure*. (n.d.). Retrieved October 2, 2023, from https://azure.microsoft.com/en-us/products/functions/

Behm, A., Palkar, S., Agarwal, U., Armstrong, T., Cashman, D., Dave, A., Greenstein, T., Hovsepian, S., Johnson, R., Sai Krishnan, A., Leventis, P., Luszczak, A., Menon, P., Mokhtar, M., Pang, G., Paranjpye, S., Rahn, G., Samwel, B., van Bussel, T., … Zaharia, M. (2022). Photon: A Fast Query Engine for Lakehouse Systems. *Proceedings of the 2022 International Conference on Management of Data*, 2326–2339. https://doi.org/10.1145/3514221.3526054

Belov, V., Tatarintsev, A., & Nikulchev, E. (2021). Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark. *Symmetry*, *13*(2). https://doi.org/10.3390/sym13020195

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM Sigmod Record*, *26*(1), 65–74.

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012a). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, 1165–1188.

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012b). Business intelligence and analytics: From big data to big impact. *MIS Quarterly: Management Information Systems*, *36*(4), 1165–1188. https://doi.org/10.2307/41703503

Cheruvu, C. (n.d.). *What is Z-Order on Databricks?* Retrieved October 24, 2023, from https://www.linkedin.com/pulse/what-z-order-databricks-saikrishna-cheruvu/

*Databricks Launches Delta to Combine the Best of Data Lakes, Data Warehouses and Streaming Systems*. (2017). https://www.databricks.com/company/newsroom/press-releases/databricks-launches-delta-combine-best-data-lakes-data-warehouses-streaming-systems

Dehghani, Z. (2019, May 20). *How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh*. https://martinfowler.com/articles/data-monolith-to-mesh.html

Dehghani, Z. (2020, December 3). *Data Mesh Principles and Logical Architecture*. https://martinfowler.com/articles/data-mesh-principles.html

Dehghani, Z. (2022). *Data Mesh* (1st ed.). O'Reilly Media, Inc.

Delacroix, S., & Montgomery, J. (2020). From Research Data Ethics Principles to Practice: Data Trusts as a Governance Tool. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3736090

*Delta Lake*. (n.d.). Retrieved September 23, 2023, from https://delta.io/

*Delta vs. Lambda: Why Simplicity Trumps Complexity for Data Pipelines*. (n.d.). Retrieved September 24, 2023, from https://www.databricks.com/blog/2020/11/20/delta-vs-lambda-why-simplicity-trumps-complexity-for-data-pipelines.html

Demchenko, Y., Ngo, C., & Membrey, P. (2013). *Architecture Framework and Components for the Big Data Ecosystem*.

*Describe medallion architecture*. (n.d.). Retrieved September 24, 2023, from https://learn.microsoft.com/en-us/training/modules/describe-medallion-architecture/2-describe-medallion-architecture

*Dremio | The Easy and Open Data Lakehouse Platform*. (n.d.). Retrieved October 3, 2023, from https://www.dremio.com/

Dutch Techcentre for Life Sciences. (2016). *European Commission embraces the FAIR principles*. https://www.dtls.nl/2016/04/20/european-commission-allocates-e2-billion-to-make-research-data-fair/

Ferguson, M. (2012). Architecting a big data platform for analytics. *A Whitepaper Prepared for IBM*, *30*.

Garani, G., Chernov, A., Savvas, I., & Butakova, M. (2019). A Data Warehouse Approach for Business Intelligence. *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 70–75. https://doi.org/10.1109/WETICE.2019.00022

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google File System. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 20–43.

Haerder, T., & Reuter, A. (1983). Principles of Transaction-Oriented Database Recovery. *ACM Comput. Surv.*, *15*(4), 287–317. https://doi.org/10.1145/289.291

Han, Z., & Zhang, Y. (2015). Spark: A Big Data Processing Platform Based on Memory Computing. *2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 172–176. https://doi.org/10.1109/PAAP.2015.41

Howson, C., Sallam, R., Richardson, J. T., apadinhas, J., Idoine, C., & Woodward, A. (2018). *Magic Quadrant for Analytics and Business Intelligence Platforms*. https://www.gartner.com/en/documents/3861464

Idoine, C. K. P. B. E. L. A. (2019). *Gartner Magic Quadrant for Data Science and Machine Learning Platforms*. https://www.gartner.com/en/documents/3899464

*IS_MEMBER | Dremio Documentation*. (n.d.). Retrieved October 28, 2023, from https://docs.dremio.com/current/reference/sql/sql-functions/functions/IS_MEMBER/

Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (2002). *Fundamentals of data warehouses*. Springer Science & Business Media.

Khine Pwint Phyu, & Wang Zhao Shun. (2018). Data lake: a new ideology in big data era. *ITM Web Conf.*, *17*, 3025. https://doi.org/10.1051/itmconf/20181703025

Kiran, M., Murphy, P., Monga, I., Dugan, J., & Baveja, S. S. (2015). Lambda architecture for cost-effective batch and speed big data processing. *2015 IEEE International Conference on Big Data (Big Data)*, 2785–2792.

Knight, M. (2018). *Data lake vs. data swamp: Leveraging enterprise data*. https://www.dataversity.net/data-lake-vs-data-swamp-leveraging-enterprise-data/

Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: a systematic literature review. *Journal of Big Data*, *6*(1), 47. https://doi.org/10.1186/s40537-019-0210-7

Le Dem, J., & Li nong, N. (2014). *Efficient Data Storage for Analytics with Apache Parquet 2.0*. https://www.slideshare.net/cloudera/hadoop-summit-36479635

Lehman, T. J., & Carey, M. J. (1985). *A study of index structures for main memory database management systems*.

Lewis, G. A., Comella-Dorda, S., Place, P., Plakosh, D., & Seacord, R. C. (2001). *An enterprise information system data architecture guide*. Carnegie Mellon University Pittsburgh, PA.

March, S. T., & Hevner, A. R. (2007). Integrated decision support systems: A data warehousing perspective. *Decision Support Systems*, *43*(3), 1031–1043. https://doi.org/https://doi.org/10.1016/j.dss.2005.05.029

*Medallion Architecture*. (n.d.). Retrieved September 24, 2023, from https://www.databricks.com/glossary/medallion-architecture

Miloslavskaya, N., & Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science*, *88*, 300–305. https://doi.org/10.1016/J.PROCS.2016.07.439

Mons, B., Neylon, C., Velterop, J., Dumontier, M., da Silva Santos, L. O. B., & Wilkinson, M. D. (2017). Cloudy, increasingly FAIR; revisiting the FAIR Data guiding principles for the European Open Science Cloud. *Information Services & Use*, *37*, 49–56. https://doi.org/10.3233/ISU-170824

Nadkarni, S., & Prügl, R. (2021). Digital transformation: a review, synthesis and opportunities for future research. *Management Review Quarterly*, *71*(2), 233–341. https://doi.org/10.1007/s11301-020-00185-7

*Overview - Azure Event Grid | Microsoft Learn*. (n.d.). Retrieved October 2, 2023, from https://learn.microsoft.com/en-us/azure/event-grid/overview

*Power BI + Dremio = A Seamless BI Experience for Cloud Data Lakes | Microsoft Power BI Blog | Microsoft Power BI*. (n.d.). Retrieved October 3, 2023, from https://powerbi.microsoft.com/en-us/blog/power-bi-dremio-a-seamless-bi-experience-for-cloud-data-lakes/

Powers, M. (n.d.). *Delta Lake Z Order*. Retrieved October 24, 2023, from https://delta.io/blog/2023-06-03-delta-lake-z-order/

Pratsri, S., & Nilsook, P. (2020). Design on Big Data Platform-Based in Higher Education Institute. *Higher Education Studies*, *10*(4), 36–43.

Qlik. (2019). *A comprehensive guide to data governance*. https://www.qlik.com/us/-/media/files/resource-library/global-us/register/whitepapers/wp-comprehensive-guide-data-governance-en.pdf?rev=02fa67c99f624265be0de0ec08a1ba86

Raguseo, E. (2018). Big data technologies: An empirical investigation on their adoption, benefits and risks for companies. *International Journal of Information Management*, *38*(1), 187–195. https://doi.org/10.1016/J.IJINFOMGT.2017.07.008

Reinsel, D., Gantz, J., & Rydning, J. (2018). *The digitization of the world from edge to core*. IDC. https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

*Selectively overwrite data with Delta Lake - Azure Databricks | Microsoft Learn*. (n.d.). Retrieved October 2, 2023, from https://learn.microsoft.com/en-gb/azure/databricks/delta/selective-overwrite

Serra, J. (2024). *Deciphering Data Architectures*. https://learning.oreilly.com/library/view/deciphering-data-architectures/9781098150754/

Subramanian, A., Douglas Smith, L., Nelson, A. C., Campbell, J. F., & Bird, D. A. (1997). Strategic planning for data warehousing. *Information & Management*, *33*(2), 99–113. https://doi.org/https://doi.org/10.1016/S0378-7206(97)00040-2

Sugiura, K., & Ishikawa, Y. (2023). *Z-ordered Range Refinement for Multi-dimensional Range Queries*.

*The Delta Lake Project Turns to Linux Foundation to Become the Open Standard for Data Lakes*. (2019). https://www.linuxfoundation.org/press/press-release/the-delta-lake-project-turns-to-linux-foundation-to-become-the-open-standard-for-data-lakes

The Economist. (2017, May 6). *The world's most valuable resource is no longer oil, but data*. 423(9039). https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

Thein, K. M. M. (2014). Apache kafka: Next generation distributed messaging system. *International Journal of Scientific Engineering and Technology Research*, *3*(47), 9478–9483.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., & others. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, *2018*.

*Web App Service | Microsoft Azure*. (n.d.). Retrieved October 2, 2023, from https://azure.microsoft.com/en-us/products/app-service/web

*What is Auto Loader? - Azure Databricks | Microsoft Learn*. (n.d.). Retrieved October 2, 2023, from https://learn.microsoft.com/en-us/azure/databricks/ingestion/auto-loader/

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., … Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*(1), 160018. https://doi.org/10.1038/sdata.2016.18

Zaharia, M., 0002, A. G., Xin, R., & Armbrust, M. (2021). Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012). Resilient Distributed Datasets: A Fault-Tolerant Abstraction for in-Memory Cluster Computing. *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 2.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster Computing with Working Sets. *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 10.

Zhang, K., Tanimura, Y., Nakada, H., & Ogawa, H. (2017). Understanding and improving disk-based intermediate data caching in Spark. *2017 IEEE International Conference on Big Data (Big Data)*, 2508–2517. https://doi.org/10.1109/BigData.2017.8258209


**NOVA Information Management School**
**Instituto Superior de Estatística e Gestão de Informação**

Universidade Nova de Lisboa