Trick or Treat: Centralized Data Lake vs Decentralized Data Mesh

Anton Dolhopolov, Arnaud Castelltort, Anne Laurent

Trick or Treat: Centralized Data Lake vs Decentralized Data Mesh

Anton Dolhopolov Arnaud Castelltort Anne Laurent

LIRMM, Univ. Montpellier, CNRS, Montpellier, France
{firstname.lastname}@lirmm.fr

Abstract. Over the course of the last few years, the augmentation of processed data and an increase in the need for fast product release cycles led to the emergence of bottlenecks in information and knowledge flows within large organizations. Recent research works attempted to resolve these issues from several perspectives, which span from the data platform architectures to the storage technologies. In this positional paper, we start by comparing the well-established methods of designing analytical data platforms and make a review of existing problems inherent to them, namely centralization of storage and ownership. It continues by analyzing the principles of a data mesh proposal and by providing an examination of unresolved challenges, such as metadata centralization. We further consider the business domain dependencies and platform architecture of our running example. The final section presents our vision for solving the identified metadata management issues in large enterprises via data decentralization and offers potential directions for future work.

Keywords: Big Data Platforms · Data Mesh · Metadata Management.

1 Introduction

In the 21st century, the information technology community highly popularized the term *big data* [9]. This notion does not differ much from the original data definition. It is rather used to reflect properties such as the amount of data, processing speed, or heterogeneity. Commonly cited big data 5 V's classification [10] describes the following characteristics: the operated data volume; the processing velocity; the variety of structured (e.g., relational), semi-structured (e.g., XML, JSON), and unstructured (e.g., audio, video) data; the veracity - meaning truthfulness, correctness, or validity of data; and the value - the hidden insights or knowledge present in data. Though it is difficult to determine who coined the term, we can easily observe the relevant growth of both scientific research [1] and industrial solutions [2] in this domain.

Historically, *data warehouses* [8] were the first generation of solutions to deal with enterprise analytical data. While being oriented on managing the structured

[1] Analytics from app.dimensions.ai. Available at https://bit.ly/3I5p3N1
[2] Big data and analytics software revenue worldwide. https://bit.ly/2Gt8VFt

data, it provided tools for business schema enforcement on the collected data, user query processing engines, and analytical data processing. These solutions are still successfully applied in OLAP-related scenarios, but they struggle with other big data requirements, namely velocity and variety. To overcome the limitations, in [3], James Dixon proposed the second generation platform called *data lake*. Its core idea is to provide a schema-on-read functionality, which is opposite to schema-on-write in the data warehouse. The need to pre-process the data is delayed until the future when the analyses are done by business analysts, data scientists, or other competent users. This facilitates the velocity of batch and real-time acquisition directly from the data generating process (application logs, user operations, crawling) to the data variety support in the underlying platform thanks to the storage in a raw format. Some noteworthy data lake architectures are data reservoir [1], Constance [6], lambda [17], and AUDAL [15]. Although data lakes can resolve the initial big data challenges, it is still seen as a centralized solution. It brings such issues as data silos, modules inter-dependencies, and long product delivery cycles [2]. We believe that platform decentralization will better benefit the large organization in comparison to widespread lake platform centralization. We thus consider the data mesh concept.

In this position paper, we make a further review of the existing data lake problems, describe the four main principles of a data mesh, present our vision for improving this architecture by decentralizing metadata and product catalogs of a data platform, and conclude with the remaining open scientific questions.

2 Data Lakes: Current Issues

In general, the existing data lake architecture models are built around centralized data storage and data ownership [5, 14]. Modern cloud computing technologies provide physical data distribution and fault-tolerant access guarantees across different geographical regions. It is essential for providing highly available services to end users. However, from a logical point of view, data is still centralized and controlled by data engineering teams. Such centralization can create bottlenecks in large enterprises.

In the rest of this section, we are going to review such issues as data silos, team friction, and changing environments.

2.1 Hidden Data

A *data silo* is a good example that often happens in companies. Back in the days of using data warehouses, different departments would curate their master data sets and use them internally. Refining and improving data quality and the analytical pipelines within the business departments isn't something wrong. Nevertheless, in the modern days of advanced analytics, like machine learning, harvesting the value of data in big enterprises requires accessing all existing datasets. As soon as the matter comes to sharing the data, problems arise. Data lakes did not masterfully escape this fate too. Instead of putting the walls in

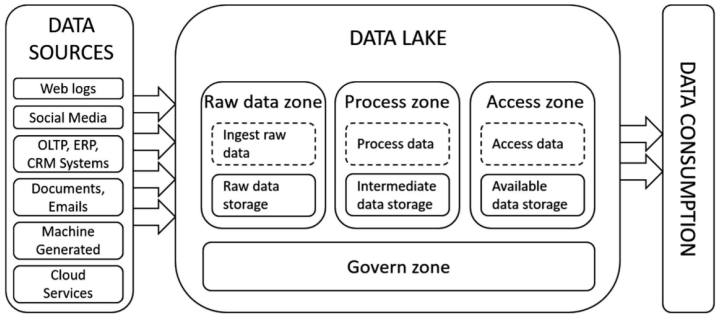

Fig. 1. Simplified Data Platform View from [13].

front of the analysts, the lake approach puts everything in one logical place for applying transformations later in time [11]. As a result, it can easily degrade into *data swamps* where it is impossible to understand and meaningfully process all of the available data [7]. A widely acknowledged solution for avoiding data garbage is a metadata management system. We shall provide a more detailed view of metadata in the section 3.3.

2.2 Teams Communication

Developing monolithic applications in big companies becomes evidently problematic over time [12]. Monolith applications create intolerably many cross-team dependencies. It means that the implementation and release of new functionality or a product from one team are in direct reliance on the work of another one. Thus the main issues arise in efficiently managing inter-team communications and speeding up the product release cycle.

Unfortunately, the same limitations are faced in data platforms. Figure 1 displays a typical data lake architecture taken from [13]. It contains three main data handling zones, namely "raw data zone" (no processing but data ingestion that can be stream or batch), "process zone" (transformed data), and "access zone" (self-service data consumption for data exploitation - querying, reporting, machine learning serving, etc). The governance zone is responsible for having a view of all the other zones and is in charge of ensuring data security, data quality, data life-cycle, data access, and metadata management.

Oftentimes data engineers (who build these zones) require close collaboration with application developers from the operational (data sources) plane, including, but not limited to data quality checks, new data attributes, or new source integration requests. The responsibility of managing and improving the data platform, in this case, is put on the shoulders of data engineers and data scientists who are detached from the day-to-day business operations of the company.

It means that engineers who work on the same product need to have high cohesion inside a single team.

2.3 Changing Organization Environment

Today, organizations rarely stay in a static environment, meaning that there are always new working requirements. The provided service or product is always subject to modifications or even cancellation. It forms a dynamic nature of all organizational processes, impacting IT teams.

To cope with it, the analytical teams should also be dynamic, agile, and flexible. The teams should fulfill the new requirements at an acceptable time without compromising the present functionalities and should adapt to changes arising from the operational plane.

In reality, it has been shown difficult to achieve in the current circumstances of large organizations. Highly specialized data platform engineers are grouped by skills and expertise and divided by functional modules at the same time. Their primary objective is to harvest and employ the value from data, but instead, they become a bottleneck in delivering innovations [3].

We have seen that the current state of analytical systems can not resolve all exigent challenges, such as data silos, team friction, and dynamic working environment. The following sections assess a Data Mesh architecture, which promises to deal with the aforementioned issues.

3 Towards Decentralized Data Platforms With Mesh Architecture

As was stated in the introduction, we advocate for the decentralization of data platforms. This section describes the concepts of the most recent works around platform decentralization and continues with its application on top of our running example. In the last part, we highlight the main ongoing difficulties that are not completely addressed.

3.1 Data Mesh 4 Pillars

Zhamak Dehghani proposed a novel architecture called *data mesh* [2]. It summons to make a paradigm shift in the way of building big data platforms. The shift is based on 4 main principles, and each one is essential on its own. Without proper implementation of all principles, it will only exacerbate the existing problems inside the organization.

Distributed Data Domains The basic ideas for successfully implementing data mesh architecture take their origin from the domain-driven design (DDD) [4]. Similarly, way as the monolithic operational plane is broken up into small, independent components like microservices so that we can divide the analytical platform into self-contained business domains. *Distributed domains architecture*

[3] Break Through the Centralized Platform Bottlenecks with Data Mesh. Available at `https://thght.works/40YM6Sj`

represents the first principle of a mesh. Instead of forcing centralized data ownership onto the narrowly specialized teams, one can benefit from forming domains aware groups of interdisciplinary employees. Each group would contain application developers, designers, data engineers, data scientists, business analysts, and product owners. This way, groups would be focused on developing both operational and analytical parts of a single product.

Data Product Design As with the software products in domain-driven methodology, the systematic approach of building the *data-oriented products* within the domain is also required. Most of the time the users would include not only the payable clients but other teams of the same organization too. Providing a great service takes designing, delivering, measuring, and constantly improving the provided data, be it A/B testing results or machine learning models. It creates a new zone of responsibility for product owners since they also need to take care of how the data products perform. Guaranteeing SLAs, publishing trustworthy data, and helping the users to discover, understand and consume the product are some of the key indicators of a successful product.

Reusable Infrastructure To avoid re-implementation of similar or even identical data acquisition, processing, and serving functionality, one must benefit from a shared *self-serve infrastructure platform*. Its main goal would be a set of common and interchangeable tools like data source connectors and sinks, transformation jobs, automatic logs, data lifecycle configuration, and so on. Eliminating the basic day-to-day engineering operations behind an easy-to-use abstraction layer will help to unlock the fast release cycle.

Federated Computational Governance The importance of providing governing policies and mechanisms can not be overstated. It is the only way how a lot of independent, distributed, interconnected, and dynamic data domains will work efficiently together. Policies help to determine how the individual domain or product should behave in the mesh. For instance, interoperability standards would describe the data publishing formats (e.g., JSON, Protobufs), schema integrity checks, information security, and privacy rules. At the same time, enforcement mechanisms will allow leveraging the verification, monitoring, or alerting processes automatically. Defining and enforcing the governance elements is not easy, but it must not be centralized. The mechanisms must be incorporated directly into the infrastructure platform, and the policies could be defined by the committee of data domain representatives and other experts (e.g. legal team).

3.2 Running Example

We experimented with our running example by applying the mesh principles. We consider different departments in a video-streaming platform: Human Resources (HR), Sales, Finance, and content production Studios. In Figure 2 (left part) we

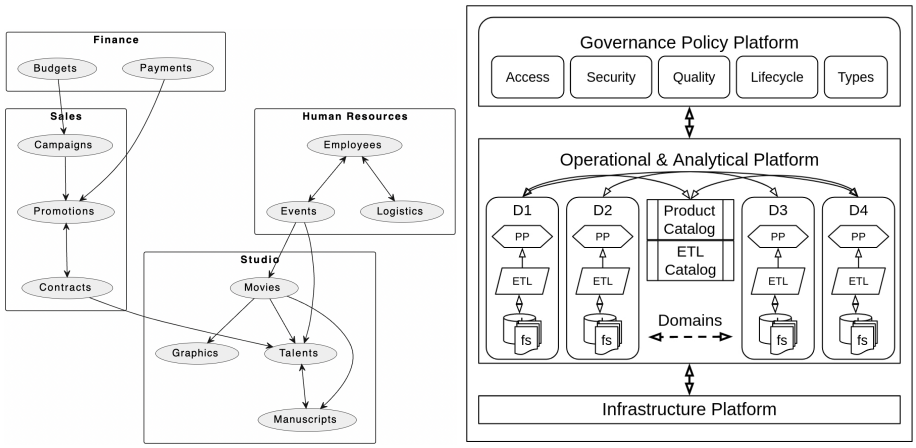

Fig. 2. Data mesh cross-domain dependencies (left) and functional architecture (right).

can easily notice that these departments inevitably have inter-dependencies, even when performing different day-to-day operations. For example, contracts issued by Sales would be related to creating new content and signing up partnerships with actors or directors (talents entity).

If we would like to integrate some analytical pipelines, it would be natural to proceed with extending the existing departmental technical modules. The Studio representatives may be interested in planning the load schedule. To deliver such an estimation, the Studio engineers will have to consume the information on contracts that the directors and actors have signed (an external data source provided by the Sales team), as well as the movie schedules (an internal source).

An important aspect of building the analytics here is not to be inclined with just another warehousing solution. One has to remember to provide the data product, not merely the statistical calculations. Our Studio product would include the metadata about data freshness, running transformation history, and semantic info on parties, costs, and jurisdictions in case we would like to filter the data further. In plus, the product could be consumed via different means: asynchronous message queues, HTTP requests, or even direct disk mounting in case of large blobs and supported data fabric functionality.

Overall, these interactions form a net of dependencies on the analytical plane in a similar way as on the operational one. The net would directly affect the design of the functional platform architecture, as is shown in Figure 2 (right part), where different departments or domains pass the data to each other directly.

In addition, the specialized data (product) and extract-transform-load (ETL) catalogs can help in data discovery, development automation, etc., as is often done in the literature. Such catalogs can be implemented in metadata management systems by using, for instance, graph databases that provide efficient processing of graph structures (that naturally arise in our running example).

We shall consider the benefits of graphs in more detail in Section 4.2.

3.3 Challenges

It may not be obvious at first look, but the principal part of building the decentralized platform stems from changing the organizational hierarchy. Companies and institutions have to update their framework for creating engineering teams, zones of responsibility, and, finally, the rendered products. One of the data mesh paradigm objectives is to unite the once-divided operational and analytical planes together across different distributed data domains. Now we shall look into left open questions.

User Profile We have mentioned previously that users operating in a centralized environment have to understand the business domain processes. It is necessary to know how the data was generated, processed, and stored in the operational plane, as well as the meaning, semantics, and relationships between attributes, tables, datasets, etc. Since all of the data is available to the end user as a huge chunk of files in one place, it becomes challenging to analyze it. Commonly, the data lake user profile is a technically knowledgeable person who has to use specific software (e.g., Apache Spark, Flink, Beam) to get the utility from data. It constrains the adaption and digital transformation of the organization which tries to implement big data managing solutions.

When adopting a decentralized platform architecture, special care should be taken to support multiple consumer profiles. Needless to say that if the advertised domain data is a chunk of files, it won't generate much utility. It means that consumption interfaces should provide and publish the data in multiple formats (datasets, analytical models, reports, etc). There is a clear need to adapt the data product attitude to unlock the underlying value.

Such an approach will expand the list of end users - from data engineers and data scientists to analysts, marketing teams, or business owners. The latter would be able to find the required information via metadata and product catalog search or relationship traversal and plug it directly into visualization tools such as Grafana, Tableau, Data Studio, etc. The growth in the number of active data users will accelerate the transformation of the organization into a data company.

Technology State On one side, decentralized approaches are still maturing, and there are no settled models and tools for implementing them across the integrated planes. On the other side, we should briefly note that: a) the operational teams have already established practices of DDD via microservices architecture (and later via service mesh paradigm), use of distributed computational platforms (e.g., Docker, Kubernetes), configuration tools (e.g., Ansible, Chef, Terraform), continuous integration and delivery services (e.g., Jenkins, GitLab); b) analytical teams also have a wide range of tools which support distributed technologies, including Apache Spark and Apache Beam for real-time and batch data processing; HDFS, Apache HBase, Apache Cassandra for non-relational and MySQL, PostgreSQL for relational data storage; Grafana or Tableau for

data visualization. But mentioned data platform technologies provide the distribution only for scalability, fault-tolerance, and disaster recovery. It is quite complicated to apply it in terms of complete data ownership decentralization.

Metadata Management There is a consensus in the research that a metadata system is an essential part of any big data platform [13, 14, 18]. Its main role is to prevent the formation of a data swamp. It is achieved by providing supplementary, descriptive information to the collected and processed data. For instance, having the data type annotations helps to avoid creating bugs in the code; marking the date and time stamps ensures the data freshness; directed acyclic graph of transformation jobs provides the data lineage, which is in turn important for failure detection. The author of the data mesh architecture puts metadata management as part of the federated computational governance but does not provide clear details on how to build it. The governance is described in a somewhat blurred state between centralization and decentralization.

4 Research Proposal

Deriving from our running example and challenges sections, we conclude to have a high impact of the mesh paradigm on the overall data platform transformation.

In the following section, we analyze and present our position regarding domain design and metadata management as part of federated governance.

4.1 Data Lake Architecture Layers

As a first attempt, we reconsider the architecture of data lakes in order to integrate a layer-oriented vision. Figure 3 shows this revision, where data sources are placed on the operational layer, while the analytic layer contains the three zones from [13], namely raw data, process, and access zones (presented in Section 2.2).

The governance layer contains a single metadata catalog organized in a flat or advanced manner and is responsible for having a view of all the other zones. It is in charge of ensuring data security, data quality, data life-cycle, data access, and metadata management.

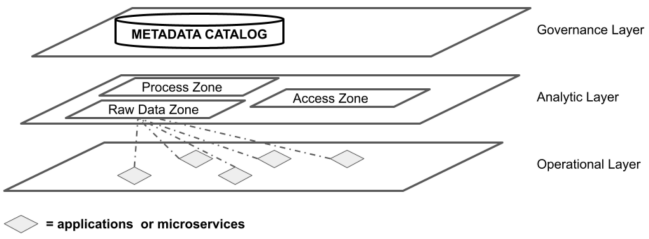

Fig. 3. Data Lake Architecture Layers revisited from [13]

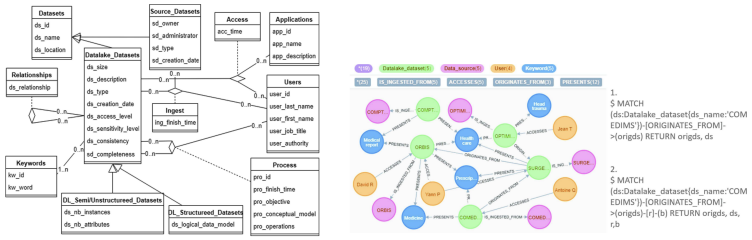

Fig. 4. Conceptual model (left) and Neo4j implementation (right) from [19]

As presented above, this metadata catalog is crucial and has attracted a lot of work in the literature. Also, it should be noted that such a catalog is more and more often implemented with property graphs. For instance, the DAMMS model [19] shown in Figure 4 uses Neo4j, which is an ACID-compliant transactional database with native graph storage and processing.

4.2 Data Mesh Architecture Layers

In this section, we focus on the role of metadata in the context of data mesh. In existing mesh architectures metadata are often stored in XML/JSON format.

We claim that metadata cannot be easily and efficiently modelized with flat or tree representations. It is due to the fundamental principle of how the data is stored and the limited ability to construct and process real-world graph structures (e.g. cross-domain relationships) when using these representations.

Therefore, we first propose more advanced models below.

Centralized Metadata Catalog In the Data Mesh architecture, the governance is seen as a federation. Within that federated governance, the metadata layer is the part where the metadata catalog is considered.

Figure 5 shows that applications from operational layers publish data that are then exposed by the data products. Those data products offer communication interfaces (e.g., REST API, Message Queue) to allow the consumption of data.

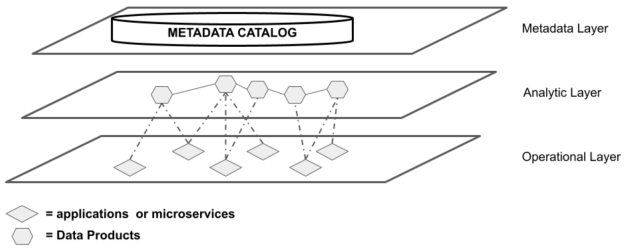

Fig. 5. Centralized Metadata Catalog

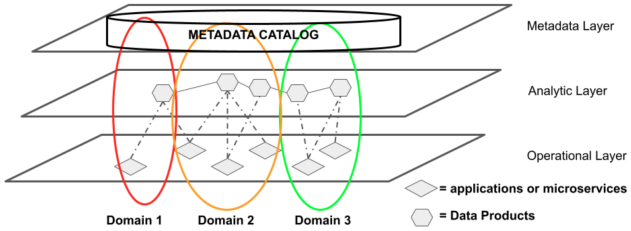

Fig. 6. Domains Perspective in Data Mesh

Every time a data product changes, it publishes its metadata to the centralized metadata repository from which clients can discover the available data.

However, this architecture has some drawbacks. Indeed, the data mesh can deliver data products that are inconsistent, and the data can be seen as non-reproducible (e.g. phantom reads).

If we take the data mesh architecture from the perspectives of business domains as shown in Figure 6, we can see those data products operate as a vertical view to serve the end users' needs. For instance, such a domain can be dedicated to HR, Sales, Finance, etc., as presented in our running example in Section 3.2.

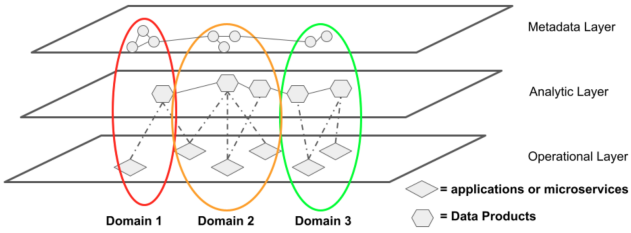

Fig. 7. Disjoint Metadata Graphs

In this architecture domains are disjoint and this is reflected in the metadata (which are represented as graphs) as shown by Figure 7.

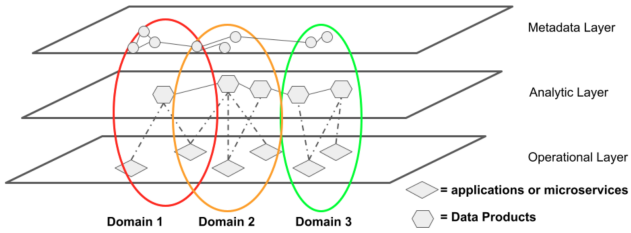

Fig. 8. Metadata Overlapping

On the opposite, Figure 8 shows the case where domains and metadata overlap. In such a case, metadata entities are shared between several domains.

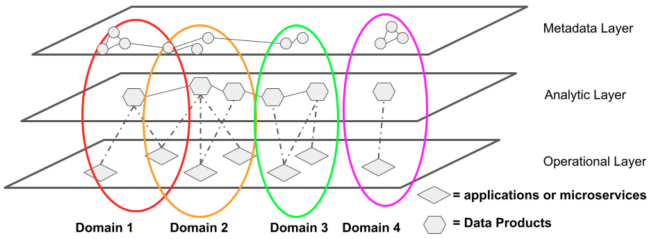

Fig. 9. Overlapping and Disjoint Metadata Graphs

As for disjoint domains, there is no obligation that all metadata graphs are connected, as in the example of Domain 4 from Figure 9.

Given a data mesh composed of n_s data sources and n_d data domains, we consider that the metadata can be formalized as:

- a global graph $\mathcal{G}$ such that;
- $\mathcal{G} = \bigcup_{i=1,\ldots,n_d} G_i$ where G_i is metadata graph associated with Domain D_i;
- There may exist $i, j \in [1, n_d]$ where $G_i \bigcap G_j = \emptyset$.

Decentralized Metadata Catalog Proposal The previous section highlights that the metadata layer can be seen as a graph and that each data product should provide metadata where entities lie in one or several business domains.

Generally, data mesh has a federation tier where centralized governance is meant to determine what semantic to use, who defines new vocabulary, and also to have a centralized catalog to allow the discovery of all metadata in one place.

In this era of decentralization, there are some pros and cons of this approach, for instance, the fact that centralized architectures lead to single points of failure.

Nevertheless, we would like to highlight the consequences of the centralized metadata repository on the delivery process. Indeed, one of the advantages of data mesh is that data products are not beholden to a single release cycle to make the improvements needed. This is improving time-to-market, end-user services, agility, etc. A data product does not have to wait to deploy everything together. Data or features can be added or transformed without waiting for the next release cycle, as it would be with most data lake architectures.

We claim that a centralized metadata catalog can break this assumption. For instance, updating a centralized metadata catalog should be added to the deployment process of each data product to avoid the side effects of an eventual consistency system. Also, if a Data Product needs to publish data about a term that has not previously been defined in the federation governance, then the release may be stalled.

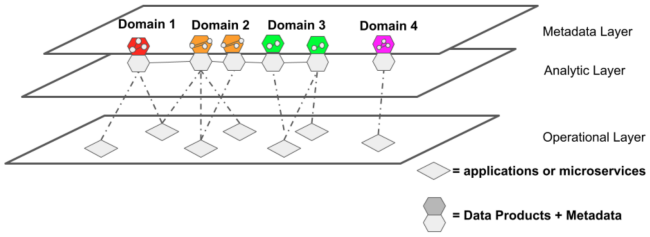

Fig. 10. Data Products with Metadata

Thus, our proposal is to treat the metadata catalog as a decentralized and distributed system. Each data product should provide its own metadata (business metadata, technical metadata, operational metadata, physical schema, semantic metadata, local and full lineage, and, quality metrics). Figure 10 illustrates that data products should be both in analytical and metadata layers.

In our proposal, we go beyond and consider that there is no more centralized metadata catalog where clients can browse the data, but a distributed and decentralized catalog accessible via discoverability, just as in the modern Web.

To go further, we consider that our metadata graph can be seen as a union of Labeled Property Graphs (LPG), each of them being hosted on a data product and representing its metadata as a graph and not as a tree or document (such as in XML or JSON format).

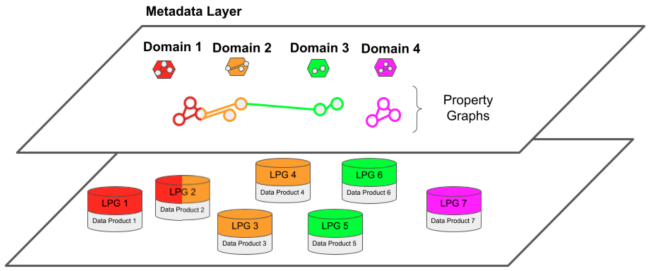

Fig. 11. Integrating Labeled Property Graphs

Figure 11 illustrates our proposition to have for each data product an LPG. An LPG can be defined as an extension of a property graph where 0 to N labels can be applied to nodes (or vertices). A property graph, as presented in [16], can be defined as a tuple $PG = \{V, E, S, P, h_e, t_e, l_v, l_e, p_v, p_e\}$, where:

- V is a non-empty set of vertices;
- E is a set of edges;
- S is a set of strings;
- P contains all $p = \{k, v\}$ key-value pairs describing properties;

- $h_e : E \to V$ is a function that yields the source of each edge (head);
- $t_e : E \to V$ is a function that yields the target of each edge (tail),
- $l_v : V \to S$ is a function mapping each vertex to label;
- $l_e : E \to S$ is a function mapping each edge to label;
- $p_v : V \to 2^P$ is a function used to assign vertices to their multiple properties;
- $p_e : E \to 2^P$ is a function used to assign edges to their multiple properties.

Given a data mesh composed of n_d data domains, we consider that the metadata can be formalized as:

- a global labeled property graph $\mathcal{LPG}$ such that
- $\mathcal{LPG} = \bigcup_{i=1,\ldots,n_d} LPG_i$ with LPG_i the labelled property graph of meta data associated with Domain D_i.
- There may exist $i, j \in [1, n_d]$ where $LPG_i \bigcap LPG_j = \emptyset$

This approach solves the issues of eventual consistency and non-reproducible environments (e.g., phantom reads), but of course, it generates new challenges to address, such as labeled-property graphs partitioning, discoverability and querying, networking issues (routing), etc., that will be discussed in further works.

5 Conclusions

The state-of-the-art centralized data lake architectures do not solve all modern big data platform challenges. Organizational scaling becomes an innovation bottleneck in companies with segregated operational and analytical platforms. To address these issues, some promising paradigms like data mesh were proposed.

However, new difficulties arise. Available metadata management systems which attempt to prevent the creation of data swamps are also designed as a centralized remedy. The mesh proposal does not offer any explicit guidelines for distributed domain implementation or evolution within the institutions. Moreover, the available technologies are still maturing and do not support the complete data decentralization capabilities for both operational and analytical planes.

Therefore, there is an open demand for developing the technology for building decentralized and interoperable data mesh with distributed metadata system.

In our research proposal, we introduce the main goals necessary to achieve for building a successful big data platform based on the data mesh principles, and we present and discuss centralized and decentralized architectures for disjoint and overlapping data domains.

Our future work will focus on refining our new metadata model and defining a logical structure based on property graphs and its decentralized technology implementation.

References

1. Mandy Chessell, Nigel L Jones, Jay Limburn, David Radley, and Kevin Shan. Designing and operating a data reservoir. In *IBM Redbooks*, 2015.

2. Zhamak Dehghani. *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, Inc., 2022.
3. James Dixon. Pentaho, hadoop, and data lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes, 2010. Accessed: 14.08.2022.
4. Eric Evans and Eric J Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.
5. Corinna Giebler, Christoph Gröger, Eva Hoos, Rebecca Eichler, Holger Schwarz, and Bernhard Mitschang. The data lake architecture framework. *BTW 2021*, 2021.
6. Rihan Hai, Sandra Geisler, and Christoph Quix. Constance: An intelligent data lake system. In *Proceedings of the 2016 international conference on management of data*, pages 2097–2100, 2016.
7. Bill Inmon. *Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump*. Technics publications, 2016.
8. William Inmon, Derek Strauss, and Genia Neushloss. *DW 2.0: The architecture for the next generation of data warehousing*. Elsevier, 2010.
9. Stefanowski J., Krawiec K., and Wrembel R. Exploring complex and big data. In *International Journal of Applied Mathematics and Computer Science*, 2017.
10. Anne Laurent, Dominique Laurent, and Cedrine Madera. *Data Lakes, Chapter 1, Introduction to Data Lakes: Definitions and Discussions*, volume 2. STE Ltd and John Wiley & Sons, Inc, 2020.
11. Natalia Miloslavskaya and Alexander Tolstoy. Big data, fast data and data lake concepts. In *7th annual international conference on biologically inspired cognitive architectures (BICA 2016)*. Procedia Computer Science, 2016.
12. Sam Newman. *Building microservices*. O'Reilly Media, Inc., 2015.
13. Franck Ravat and Yan Zhao. Data lakes: Trends and perspectives. In *International Conference on Database and Expert Systems Applications*, pages 304–313. Springer, 2019.
14. Pegdwende Sawadogo and Jerome Darmont. On data lake architectures and metadata management. In *Journal of Intelligent Information Systems*. Springer, 2020.
15. Pegdwendé Sawadogo, Jérôme Darmont, and Camille Noûs. Joint management and analysis of textual documents and tabular data within the audal data lake. In *European Conference on Advances in Databases and Information Systems*, pages 88–101. Springer, 2021.
16. Dominik Tomaszuk. Rdf data in property graph model. In *Research Conference on Metadata and Semantics Research*, pages 104–115. Springer, 2016.
17. John Tomcy and Pankaj Misra. *Data Lake for Enterprises Leveraging Lambda Architecture for building Enterprise Data Lake*. Packt Publishing Ltd, 2017.
18. Asma Zgolli, Christine Collet, and Cedrine Madera. *Data Lakes, Chapter 4, Metadata in Data Lake Ecosystems*, volume 2. STE Ltd and John Wiley & Sons, Inc, 2020.
19. Yan Zhao. *Metadata Management for Data Lake Governance*. PhD thesis, Toulouse 1, 2021.