Implementing a Blockchain-Powered Metadata Catalog in Data Mesh Architecture

Anton Dolhopolov, Arnaud Castelltort, Anne Laurent

Implementing a Blockchain-Powered Metadata Catalog in Data Mesh Architecture

Anton Dolhopolov Arnaud Castelltort Anne Laurent

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

Abstract. This paper explores the implementation of a blockchain-powered metadata catalog in a data mesh architecture. The metadata catalog serves as a critical component in managing data at scale, allowing for efficient discovery, access, and governance. By integrating blockchain technology, the metadata catalog can provide federated control, immutability, and transparency in managing metadata across a distributed network of data domains. This paper discusses the benefits of using blockchain technology in the metadata catalog and provides a proof-of-concept implementation of a blockchain-powered metadata catalog in a data mesh architecture using HyperLedger Fabric. The paper also highlights some challenges and potential solutions for adopting this approach, including scalability, interoperability, and governance concerns. Overall, this paper presents a novel approach for implementing a secure and federated metadata catalog in data mesh architecture that can improve the efficiency, reliability, and transparency of data management.

Keywords: Blockchain · Data Mesh · Metadata Catalog · Data Governance

1 Introduction

Data mesh [4] is one of the most recent grounded theoretical developments in the data management field. Contrary to the previous approaches such as data warehouses [11] or data lakes [13], this new paradigm is inspired by the microservices arhictecure [18]. It advocates the logical decentralization of the data platform. It builds upon the 4 core principles: distributed domains data ownership, data-as-a-product, self-serve data platform, and federated computational (data) governance. In data mesh, the decentralization of the analytical platform architecture components aims to enable organizational scaling and close the gap between data generation and data analyses.

Metadata management is another important element of an efficient data platform. The research shows that it facilitates discoverability, accessibility, interoperability, semantic comprehension, and utilization of the data [9, 19]. However, most of the theoretical models and practical implementations focus on centralized metadata catalog architectures. Therefore, these works can not be easily adapted and employed in decentralized systems like data mesh.

On the other side, blockchain technology envisions the decentralization of information system actors at its core. The recent works show that it can be beneficially applied for cross-organizational collaboration [12] and metadata management [5,14]. The attractiveness of this technology includes such properties as a historical, secure, and immutable ledger, unified data control in form of smart contracts, and use of the consensus algorithms for providing data distribution.

In this paper, we claim that blockchain technology can profit the institutions that want to adapt the data mesh paradigm. First, we describe the challenges of data governance in the mesh. Then, we draw upon the previous theoretical work of metadata catalog distribution by providing a proof-of-concept implementation of the catalog based on the open-source Hyperledger Fabric platform. The paper concludes with discussions and further research plans.

2 Challenges of Data Governance in Data Mesh

According to [16], data governance "... is a collection of information-related processes, roles, policies, standards, and metrics oriented to maximize the effectiveness of deriving business value from data". Thus, data mesh should offer a system where technical engineers, business users, legal representatives, and other participants can easily define business processes, regulatory policies, data access roles, and other elements that will shape the platform's inner operations.

The centralized governance imposes the rules and standards to follow by each team which enables interoperability. But it has organizational scaling issues as it has been shown for data warehouse and data lake systems [15]. At the same time, the decentralized governance leaves a lot of freedom for each domain team which brings the risk of building incompatible data products, duplicating the development efforts, missing compliance procedures, etc.

In [4], the author of the data mesh concept, Zhamak Deghani, highlights the importance of federated computational data governance as the need to "maintain a dynamic equilibrium between domain autonomy and global interoperability". It means a striking balance of global policies (centralization) and local freedoms (decentralization) is required.

Federated governance systems should provide the tools that help to collaborate and ease the definition and access to data schemes, semantic knowledge, and lineage, but also to automatically enforce the security, transparency, and legal policies. However, the main challenge surfaces when constructing such a system is based on already available tools or existing research.

In Section 1 we mentioned that in the research metadata management system often performs the functions of the data governance system. Nonetheless, the majority of scientific systems such as GOODS [8], AUDAL [20], DAMMS [21], Child et al. [3], Cherradi et al. [2], and industrial products as DataHub Project[1] and Apache Atlas[2] implement only the centralized approach. It means that the

[1] https://datahubproject.io/
[2] https://atlas.apache.org/

metadata is collected within the central repository, analyzed and linked in post-hoc fashion, and often made available to users via web portals.

Some recent works [7, 10] attempt to implement federated data governance by using semantic web technologies, but these are still immature systems.

Therefore, there is an open research need for designing and developing federated metadata management systems.

3 Introducing Blockchain-Powered Metadata Catalog

Section 2 describes the challenges associated with building efficient data governance in data mesh. In this section, we briefly describe 3 types of metadata systems that can be implemented by companies. Afterward, we proceed with a description of how Hyperledger Fabric can be used for developing an effective federated catalog that satisfies security, traceability, transparency, immutability, and interoperability demands.

3.1 Metadata Catalog Types

The theoretical work of authors defines 3 types of metadata catalogs in [6].

In Type I all metadata records are kept in a central repository. To enable the data access control, it presents the *visibility* and *access* functions that are associated with each user. The initial metadata is generated upon data product release or update and pushed to the repository that is available to the participants of the mesh (potentially through the web portal interface).

In Type II the repository of records is distributed. Instead of pushing the metadata about data products into a central repository, each domain of the mesh hosts an instance of the metadata catalog. The system is also composed of a data synchronization algorithm and standardized data access policies.

Type III catalog is seen as a complete system's decentralization with no single repository or standardized procedures. Each domain (or a subset of domains) is free to implement the storage and governing policies however it likes, but it is still required to provide the metadata exchange, discovery, and query interfaces.

3.2 Using Hyperledger Fabric for Metadata Management

Hyperledger Fabric (HLF) platform [1] is an open-source project hosted by the Linux Foundation. Its modular architecture allows using various computational components depending on the system requirements: pluggable consensus protocol, identity providers, and transaction endorsement policy. If there are additional security demands, it allows the exploitation of network segmentation via channels and private data exchange algorithms.

We describe how Hyperledger Fabric fits the Type II catalog thereafter.

Chaincode and Platform Governance. A smart contract (SM) is a universal way of enforcing some pre-determined, beforehand agreed procedure over the given asset during the parties' interaction. HLF defines a notion of a *chaincode* that is assembled from one or more smart contracts and policies. It is the smallest software module deployed as a Docker container. Chaincode policies determine how the underlying contracts should be executed. For instance, a policy can define which nodes should perform an endorsement of the proposed transaction.

On the other hand, chaincode itself can be seen as a transparent and interoperable governing tool for interacting with the ledger. It guarantees a standardized way of modifying the information and performs an ongoing metadata integrity verification which fits the implementation requirements of the Type II catalog.

Upon the chaincode execution, it is also possible to emit events. Therefore, the catalog users (or simply nodes) can listen to the specific types of ledger updates and discover the newly published information.

Another distinctive characteristic of the HLF is that developers can define smart contracts with general-purpose programming languages like Java, Go, or JavaScript/TypeScript (NodeJS-based contracts). This is contrary to such platforms as Ethereum which makes the technology adaption curve easier in general. In some way, it also offers interoperability capabilities since it is possible to develop a number of chaincode contracts using any supported language. The only requirement for using such a contract is the agreement of the network majority.

Overall, platform control is done through the policies management mechanisms. Policies of the HLF (as a part of a single chaincode package or of the global channel configuration) represent how members come to the agreement of accepting or rejecting changes to the network, channel, or smart contract.

Private Communication. Permisioned (private) blockchain provides more advantages for building the Type II metadata catalog compared to the permissionless (public) one. For example, domain or user identification allows to design more secure systems with access control management. For more details, we refer the reader to [6] which outlines the benefits of using a permisioned blockchain.

In its nature, HLF is a permisioned blockchain platform that runs the private network of uniquely identified components by using the membership service providers (MSP). We may imagine the MSP as a certificate authority (CA) extension that establishes the identity of each element of the network - organizations, nodes, applications, and policies. The whole network is structured as a number of non-overlapping communication *channels*. To give an analogy, the channel resembles a subnet of the CIDR-based network in OSI reference model[3].

Private data exchange is another privacy-preserving technique that lets to share secret data only with intended parties. The channel segregation method limits the participating parties from access to the whole ledger. By contrast, the private exchange mechanism lets to monitor that data sharing has taken place without the data itself, for instance, when secretly passing the data product access credentials and recording it on the ledger.

[3] https://www.iso.org/standard/20269.html

Ledger and Consensus Algorithm. When implementing the metadata management system with the help of blockchain technologies, the ledger takes the role of the underlying metadata records storage medium. Its properties such as distribution, immutability (with historical information as a side effect), and cryptographic signatures suit well the required needs.

Data indexing improves the search performance in a big pile of records which is often the case for immutable ledgers. In HLF the equivalent indexing functionality is done via the ledger state database. This database keeps only the latest modification over the given asset identified by a unique key. It is implemented using either the key-value (LevelDB) or key-document (CouchDB) stores.

For maintaining the ledger in the synchronized state and to make the configuration upgrades possible in the network, HLF provides several options for implementing a consensus algorithm. For the extreme cases when the blockchain users operate in a trustless environment, it is possible to use the Byzantine Fault Tolerance (BFT) protocol. It might be well suited for the cross-organizational (meta)data exchange initiatives (like in healthcare or finance) when security and network poisoning fault-tolerance are of high importance.

By contrast, if the environment is partially or fully trusted (like within the same organization), it might be sensible to implement a more performant Crash Fault Tolerant (CFT) consensus algorithm. In fact, due to the modular architecture, it is possible to use distinct protocols for ledger or configuration updates.

In this section, we discussed 3 metadata catalog types: centralized, distributed, and decentralized. We also presented the advantages of employing a blockchain platform for building a distributed catalog. Next, we present our proof-of-concept catalog model and the implementation based on the Hyperledger Fabric.

4 Metadata Catalog Model and Implementation

Our research contribution is three-fold. First, we derive our model from the works on the federated data exchange [12] and provenance metadata management [5] by proposing novel asset structures to be used in the metadata catalog.

Second, we define a catalog architecture integrated with data mesh products and a smart contract that consists of several functions for using the ledger as the metadata information store.

Third, we provide 4 scenarios of how the proposed model and contract would be used for managing the data products metadata in the context of a data mesh.

4.1 Ledger Asset Structure

At the baseline, the metadata should describe the data product (DP) from different perspectives to facilitate its discovery, addressing, understanding, and manipulation. The standard functional classification includes operational (location, size), technical (format, type, schema), and business (notions, context, process lifecycle) metadata.

Our proposed catalog ledger structure is comprised of the following assets:

- DataProductAsset - describes the main information such as name, location
- MetadataAsset - describes the DP's metadata such as owner, state, lineage
- ConsumptionRequestAsset - defines a request to use the published DP
- ConsumptionResponseAsset - defines a response to the open DP's request
- ConsumptionResponseUpdateAsset - defines an update to the response

These assets (with detailed definitions available in Appendix A) enable the data mesh governance in the following aspects.

The Lineage field allows DP discovery and risk assessments. By traversing and studying the dependency graph users can project the product failure consequences, or they may decide to consume more coarse or refined products.

The Schema and SampleDataLocation fields help with DP understanding and interoperability. Schema may contain semantics, while samples represent the underlying data that is open to everyone. SampleDataLocation helps with the development of more refined products without administrative approval delays.

The asset ownership and integrity verification is implemented through the cryptographic hashes mechanism. The hash matching during the DP processing is essential for mitigating fraudulent activities.

Consumption-related assets (Request-Response-Update) are used for access control and usage tracking. It helps to implement security and regulation compliance measures in the first place.

4.2 Chaincode Operations and Catalog Architecture

As mentioned in Section 3.2, we consider the smart contract machinery, and the chaincode in particular, as an enforcement tool of the pre-agreed governance.

Our proposed chaincode contracts allow the user to define and record in the metadata catalog new assets with various types of information:

- RegisterDataProduct function records metadata regarding new DP in the data mesh by creating DataProductAsset and MetadataAsset objects
- RequestDataConsumption function records information about DP access request by creating ConsumptionRequestAsset object
- ResolveConsumptionRequest function records the response to the DP access request by creating ConsumptionResponseAsset object
- UpdateDataProduct function records a new asset with updated metadata information and old asset references by creating new DataProductAsset and MetadataAsset objects
- UpdateConsumptionResponse function records a response update information by creating ConsumptionResponseUpdateAsset object (optional, but only relevant in case of a previous UpdateDataProduct function call)
- GetAllAssets function returns all available assets in the ledger

Classical Hyperledger Fabric network architecture includes a communication channel, blockchain nodes, ledger storage, and chaincode. The applications (or data products) and certificate authorities are considered outside of the network but they are still communicating with nodes.

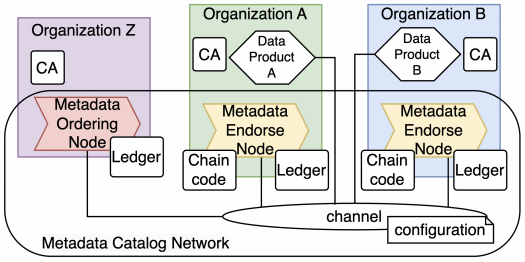

Fig. 1. Proof-of-concept metadata catalog architecture

We provide our proof-of-concept (PoC) catalog architecture in Figure 1. We use two organizations to demonstrate the data product registration and consumption interactions. It is a simplified plot since it does not contain all HLF platform elements (like Membership Service Provider or Private Data Store). However, it represents the most important parts of the system such as metadata nodes that own the chaincode and/or the ledger.

The endorsement node is used for validating the proposed transactions with new assets to be appended in the ledger. In the default policy setup, the majority of endorsing nodes is required meaning that both Organization A and Organization B should approve the transaction for accepting it. Organization Z holds only an ordering node that does not affect the approval or refusal decision, but it is solely responsible for ordering a number of transactions into a block that will be added to the ledger. It guarantees that blocks are identical across nodes.

In an example scenario, before connecting to the running metadata catalog network, a new organization would have to setup a CA infrastructure. This infrastructure is wholly owned by the organization and is necessary to generate identities for its domains, data products, feedback loops, etc. After intermediating the request to join the catalog, the MSP would assign the organization roles and rights based on the network configuration policies. The assigned roles and rights define how and who this organization can interact with.

4.3 Use-Cases for Distributed Metadata Catalog in Data Mesh

In this subsection, we present 4 use-cases for using the proposed distributed catalog in the context of a data mesh platform of the video-service company.

In the most basic case, a data product owner (DPO) from organization A wants to register a new DP in the catalog. For doing this, one has to use the RegisterDataProduct function by passing the DataProductAsset structure (a JSON that corresponds to the type fields) and some other parameters (name, description, consumed DPs). When the new asset transaction is validated, it will be recorded into the block and synchronized across the different nodes, and the new DP will become available to other domains. The sequence diagram of the whole process is demonstrated in Figure 2 which also shows the final ledger and DP PromoAd Costs states.

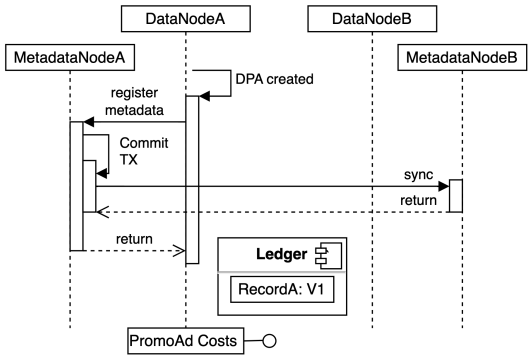

Fig. 2. Process of a metadata asset registration

In the second case that is shown in Figure 3, a DPO from organization B (DPO-B) wants to consume a new DP from organization A (DP-A) that was created before. For doing it, DPO-B has to use the RequestDataConsumption function by passing the desired product information (DP-A in this case) and the access rights. When the transaction is accepted it is reflected on the ledger and the request becomes available to the DPO-A for review. To approve or deny the request, DPO-A has to use the ResolveConsumptionRequest function by passing the request id and the rights to be granted. If the response was positive (e.g. "read" right was granted), DPO-B can setup the product consumption, and the lineage relationship is established between DP-A and DP-B.

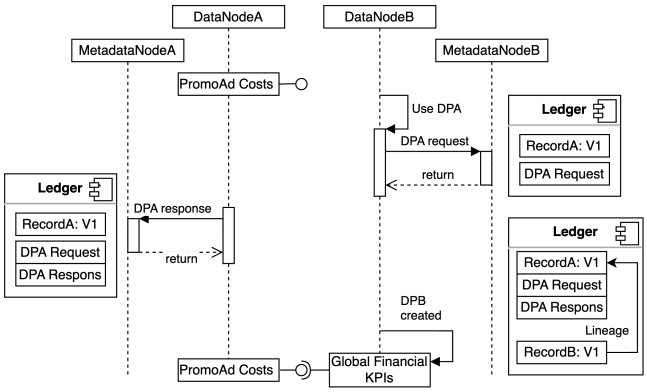

Fig. 3. Process of requesting a metadata asset and responding to the request

In the third case, we have a settled product consumption chain where the end users can see the movie promotional list. Then, a DPO-A decides to make an update of the existing DP-A. When the update is recorded, the versioning

relationship arises since the ledger will contain a new asset pointing to the old DP-A asset. As soon as the update is propagated across the network, Node B is notified of the update and it can automatically stop the DP-B if there is a breaking change of DP-A (as shown in Figure 4).

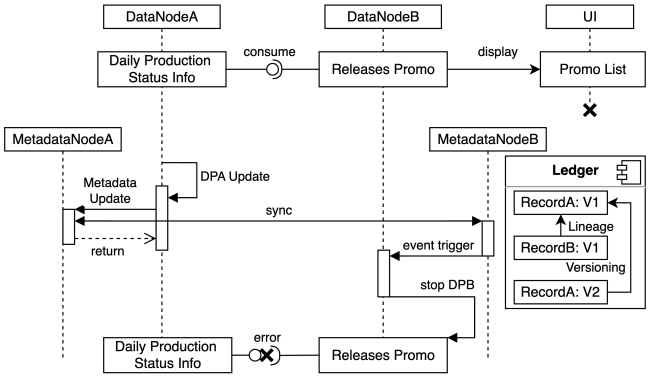

Fig. 4. Process of metadata asset update and the update event handling

In the last case shown in Figure 5, a DPO-A also decides to update its DP-A consumption responses. Such action is optional but it is useful in the sense that it brings automatic forward compatibility of the previously approved requests. When the response update is recorded and propagated, Node B is notified and proceeds with testing the new DP-A setup. If the consumption attempt is successful, then it can re-publish the updated version of the DP-B metadata reflecting the recent changes of the upstream dependency graph and re-establishing the data flow.

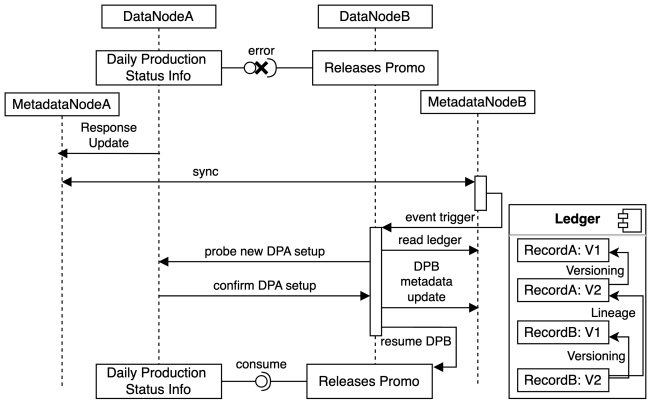

Fig. 5. Process of response asset update and automatic consumption recovering

5 Contribution Discussion

This section shows the advantages of the proposed model and its implementation, as well as missing requirements and potential solutions.

5.1 Model Pros and Cons

Sawadogo et al. [19] outlined 6 main requirements for the metadata systems in the data lake context: data indexing (DI), data versioning (DV), polymorphic data (PD), semantic enrichment (SE), link generation (LG), and usage tracking (UT). Nonetheless, these features are still relevant and necessary for constructing efficient metadata systems in the data mesh.

Dehghani [4] did not state any need for a metadata management system per se, but rather attributed all the mentioned properties as part of the data product design itself. Although, we find it conceptually easier to attribute these to a metadata catalog along with other federated governance requirements, including, but not limited to, standards and policies as computational blocks (PCB), system evaluation feedback loops (FL), and governance leverage points (LP) [17].

Since data mesh builds upon the micro-services architectures ideas, we also consider the necessary properties of being independently deployable (ID) and automatically testable (AT). A conclusive summary is displayed in Table 1.

Table 1. Supported metadata management system requirements

LP	FL	AT	LG	SE	PD	DI	DV	UT	PCB	ID
✗	✗	✗	∗	∗	∗	✓	✓	✓	✓	✓

✓ − supported ✗ − unsupported ∗ − partially supported

5.2 Potential solutions

Data product polymorphism (or polyglot data) can not be represented at once as a single asset definition with multiple dimensions (stream, batch, report, etc). But it is possible to register multiple DP metadata assets related to each output dimension. Semantic enrichment and link generation are not implemented automatically in the system. The user still has to pass such information as consumed DPs, schema, and description. It can be improved by extending the model for multiple product dimensions and by introducing AI tools for automatic metadata extraction, such as data similarity and schema linking.

Automated testing is a big challenge that developers still face when making upgrades in any large distributed system. Apparently, it is also inherent to our proposed catalog model and it is difficult to point to any viable solution. Still,

HLF has *channeling* support that can be used for testing any platform upgrades without affecting the main catalog network.

Feedback loops and leverage points are other important but highly challenging elements with no evident solution. Donella Meadows, the author of the concepts, mentions the use of global incentives that may profit from the implementation of FL and LP. Introducing such incentives component easily aligns with the nature of blockchain platforms in the first place. However, we think that it is also required to have well-integrated federated governance and self-serve infrastructure platforms which is not the case today.

6 Conclusions and Further Research

Overall, this paper presented a novel approach for implementing metadata catalog in data mesh architecture. At first, we inspected the challenges of federated governance and the limits of existing systems. Second, we described the benefits of using Hyperledger Fabric. Third, we proposed a new, blockchain-based metadata model, distributed system architecture, and demonstrated 4 relevant metadata catalog use cases for enabling computational governance in data mesh.

Our further work will focus on extending the proposed model and running experiments for estimating the model performance throughput. We will also investigate graph-based distributed ledger technologies for improving catalog functionalities and make a comparative study of the construction process of a Type III metadata catalog on top of property graphs.

References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth EuroSys conference. pp. 1–15 (2018)
2. Cherradi, M., El Haddadi, A., Routaib, H.: Data lake management based on dlds approach. In: Networking, Intelligent Systems and Security: Proceedings of NISS 2021. pp. 679–690. Springer (2022)
3. Child, A.W., Hinds, J., Sheneman, L., Buerki, S.: Centralized project-specific metadata platforms: toolkit provides new perspectives on open data management within multi-institution and multidisciplinary research projects. BMC Research Notes **15**(1), 106 (2022)
4. Dehghani, Z.: Data Mesh: Delivering Data-Driven Value at Scale. O'Reilly (2022)
5. Demichev, A., Kryukov, A., Prikhodko, N.: The approach to managing provenance metadata and data access rights in distributed storage using the hyperledger blockchain platform. In: Ivannikov Ispras Open Conference. IEEE (2018)
6. Dolhopolov, A., Castelltort, A., Laurent, A.: Exploring the benefits of blockchain-powered metadata catalogs in data mesh architecture. arXiv preprint (2023)
7. Driessen, S., Monsieur, G., van den Heuvel, W.J.: Data product metadata management: An industrial perspective. In: Service-Oriented Computing–ICSOC 2022 Workshops: ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, November 29–December 2, 2022 Proceedings. pp. 237–248. Springer (2023)

8. Halevy, A.Y., Korn, F., Noy, N.F., Olston, C., Polyzotis, N., Roy, S., Whang, S.E.: Managing google's data lake: an overview of the goods system. IEEE Data Eng. Bull. **39**(3), 5–14 (2016)
9. Hillmann, D.I., Marker, R., Brady, C.: Metadata standards and applications. The Serials Librarian **54**(1-2), 7–21 (2008)
10. Hooshmand, Y., Resch, J., Wischnewski, P., Patil, P.: From a monolithic plm landscape to a federated domain and data mesh. Proceedings of the Design Society **2**, 713–722 (2022)
11. Inmon, W., Strauss, D., Neushloss, G.: DW 2.0: The architecture for the next generation of data warehousing. Elsevier (2010)
12. Koscina, M., Manset, D., Negri-Ribalta, C., Perez, O.: Enabling trust in healthcare data exchange with a federated blockchain-based architecture. In: International Conference on Web Intelligence-Companion Volume (2019)
13. Laurent, A., Laurent, D., Madera, C.: Data Lakes. John Wiley & Sons (2020)
14. Liu, L., Li, X., Au, M.H., Fan, Z., Meng, X.: Metadata privacy preservation for blockchain-based healthcare systems. In: Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part I. pp. 404–412. Springer (2022)
15. Machado, I.A., Costa, C., Santos, M.Y.: Data mesh: concepts and principles of a paradigm shift in data architectures. Procedia Computer Science **196**, 263–271 (2022)
16. Majchrzak, J., Balnojan, S., Siwiak, M., Sieraczkiewicz, M.: Data Mesh in Action. Manning Publishing (2022)
17. Meadows, D.H.: Leverage points: Places to intervene in a system (1999)
18. Newman, S.: Building microservices. O'Reilly Media, Inc. (2015)
19. Sawadogo, P., Darmont, J.: On data lake architectures and metadata management. Journal of Intelligent Information Systems **56**(1), 97–120 (2021)
20. Sawadogo, P., Darmont, J., Noûs, C.: 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. pp. 88–101. Springer (2021)
21. Zhao, Y.: Metadata Management for Data Lake Governance. Ph.D. thesis, Univ. Toulouse 1 (2021)

A Metadata Catalog Assets

```
type DataProductAsset struct {
    Name                string   `json:"Name"`
    DataHash            string   `json:"DataHash"`
    DataLocation        string   `json:"DataLocation"`
    SampleDataLocation  string   `json:"SampleDataLocation"`
    Schema              string   `json:"Schema"`
    Version             string   `json:"Version"`    }
type MetadataAsset struct {
    ID          string              `json:"ID"`
    IsDeleted   bool                `json:"IsDeleted"`
    CreateTime  time.Time           `json:"CreateTime"`
    Description string              `json:"Description"`
    Name        string              `json:"Name"`
    Owner       string              `json:"Owner"`
    OldAssetId  string              `json:"OldMetadata"`
    Product     *DataProductAsset   `json:"Product"`
    Lineage     []MetadataAsset     `json:"Lineage"` }
type ConsumptionRequestAsset struct {
    ID              string              `json:"ID"`
    Name            string              `json:"Name"`
    RequestTime     time.Time           `json:"RequestTime"`
    Owner           string              `json:"Owner"`
    RequestedAsset  *MetadataAsset      `json:"RequestedAsset"`
    Rights          []Right             `json:"Rights"` }
type ConsumptionResponseAsset struct {
    ID           string                      `json:"ID"`
    Name         string                      `json:"Name"`
    Owner        string                      `json:"Owner"`
    ResponseTime time.Time                   `json:"ResponseTime"`
    Request      *ConsumptionRequestAsset    `json:"Request"`
    ProductInfo  *MetadataAsset              `json:"ProductInfo"`
    GrantedRights []Right                    `json:"GrantedRights"` }
type ConsumptionResponseUpdateAsset struct {
    ID          string                          `json:"ID"`
    AutoApprove bool                            `json:"AutoApprove"`
    Name        string                          `json:"Name"`
    Owner       string                          `json:"Owner"`
    ProductInfo  *MetadataAsset                 `json:"ProductInfo"`
    Responses   []ConsumptionResponseAsset      `json:"Request"`
    UpdateTime  time.Time                       `json:"UpdateTime"` }
```