International Journal of Internet, Broadcasting and Communication Vol.16 No.2 193-202 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.2.193

IJIBC 24-2-23

A Fabricator Design for Metadata CI/CD in Data Fabric

Chae-Yean Yun*, Seok-Jae Moon**

** The master's course, Graduate School of Smart Convergence, KwangWoon University, Seoul, Korea*
*** Professor, Graduate School of Smart Convergence, KwangWoon University, Seoul, Korea*
E-mail : {dbscodus, msj8086}@kw.ac.kr

Abstract

As companies specialize, they use more modern applications, but they still rely on legacy systems and data access is limited by data silos. In this paper, we propose the Fabricator system, a design system for metadata based on Data Fabric that plays a key role in the data orchestration layer consisting of three layers: Sources Engine, Workload Builder, and Data Fabric Ingestion, thereby achieving meaningful integration of data and information. Provides useful insights to users through conversion. This allows businesses to efficiently access and utilize data, overcoming the limitations of legacy systems.

Keywords: *Apache, BigData, Cloud, Database, Data Fabric, Data Integration, Metadata*

1. INTRODUCTION

Data Fabric architecture is an integrated framework for managing big data and data across various stages [1]. Data Fabric architecture effectively utilizes the latest technologies such as big data analysis and cloud computing for large enterprises [2]. The existing Data Fabric architecture consists of five core layers [3]. First, the Data Collection layer where data is loaded into the BigData store. Second, the Data Management and Intelligence layer, where data is governed, protected, managed and accessed and other related processes take place. Third, the Data Orchestration layer where data is integrated and converted into meaningful information that users can use. The fourth is the Data Discovery layer, which is the available data that users can view. Lastly, there is the Data Access layer, which is the interface through which users can access and obtain data to gain business insights. As companies expand to become more specialized, they utilize an increasing number of cutting-edge system supports meaningful integration of data and conversion into information, providing useful insights to users. Therefore, companies can quickly access and utilize data effectively, overcoming limitations caused by legacy systems. The structure of this paper is as follows. Section 2 describes related research, and Section 3 describes the Fabricator design system and Fabricator model configuration for Data Fabric metadata Continuous Integration/Continuous Deployment (CI/CD). Section 4 describes application cases of the system and comparative analysis with other systems, and finally, Section 5 describes conclusions and future research. applications. However, storage systems still rely on legacy systems, and data silos make

Manuscript Received: April. 5, 2024 / Revised: April. 18, 2024 / Accepted: April. 24, 2024
Corresponding Author: msj8086@kw.ac.kr
Tel: +82-2-940-8283, Fax: +82-2-940-5443
Author's affiliation: Professor, Graduate School of Smart Convergence, KwangWoon University, Seoul, Korea

approaches to data access stagnant and difficult over time. As a result, problems arise in terms of low productivity, efficiency, data accessibility, reliability of storage and security, and scalability [4, 5].

In this paper, we propose a design system for Data Fabric metadata, which plays a key role in the Data Orchestration layer. The proposed system consists of three layers: Sources Engine, Workload Builder, and Data Fabric Ingestion. This allows businesses to efficiently access and utilize data, overcoming the limitations of legacy systems.

2. PROPOSED SYSTEM

In this paper, we propose a Fabricator design system for metadata CI/CD of a new Data Fabric. The Fabricator Design System consists of a total of three layers.

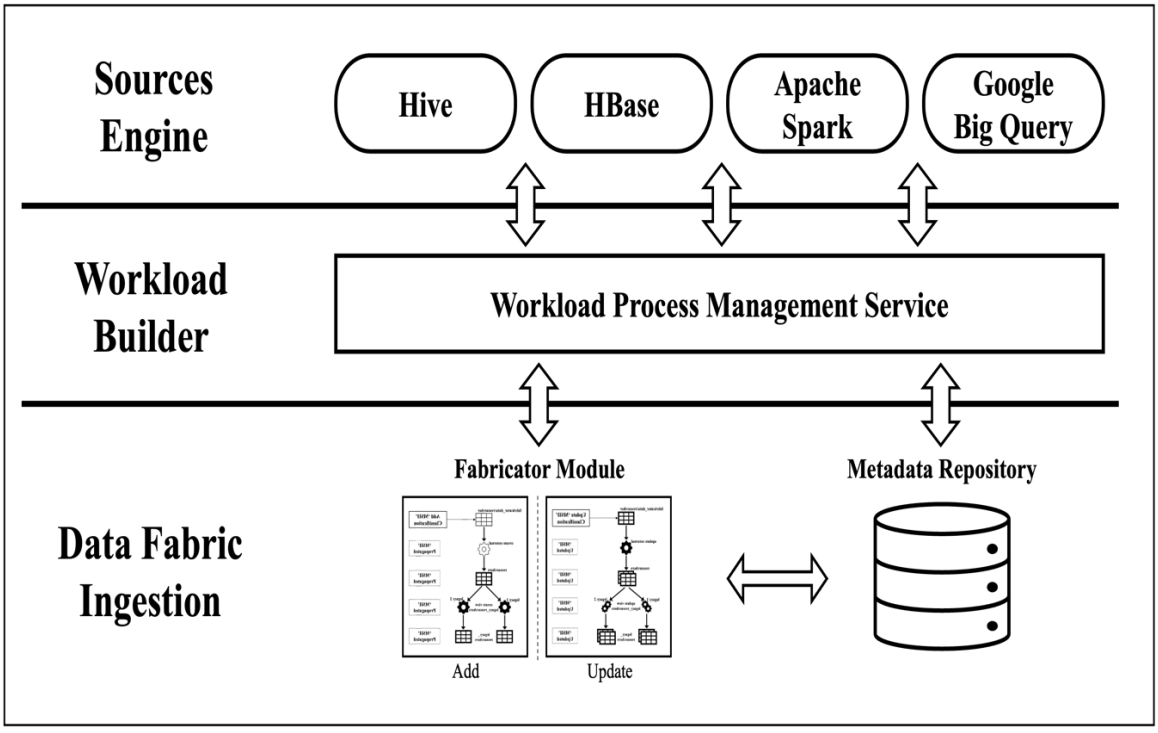


Figure 1. Fabricator Architecture

The first layer, the Sources Engine layer, includes databases such as Hive, HBase, Apache Spark, and Google Big Query, which are used as the starting point for data processing.

- Hive: It is a Data Warehouse that stores and manages large-scale data.
- HBase: It is a distributed NoSQL database that stores large amounts of unstructured data.
- Apache Spark: It is an open source distributed computing framework for big data processing.

• Google Big Query: This is a Data Warehouse service provided by Google Cloud.

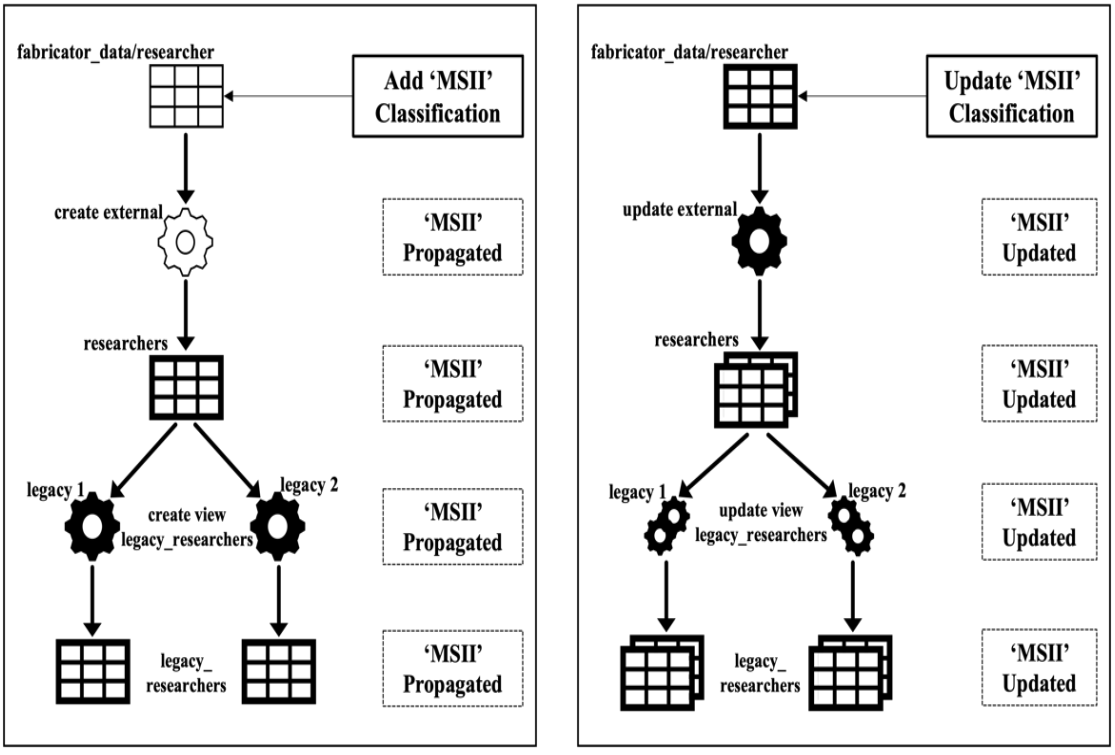
The second layer, the Workload Builder layer, is responsible for organizing and managing data processing tasks, and is the step to define and execute data processing tasks.

The third layer, the Data Fabric Ingestion layer, represents detailed processes including the Fabricator Module.

• Fabricator Module: This is an important step to process the data and transfer it to the metadata repository. This includes the process of extracting, converting, and loading data.

• Metadata Repository: Refers to a metadata repository where data is stored after data processing. This repository stores information related to processed data. Metadata includes information that describes the characteristics, structure, and relationships of data.

2.1. System Component



(A) Inserting **(B) Updating**

Figure 2. Inserting& Updating metadata schema identification information (MSII) repository propagation using Fabricator.

Figure 2-A visually illustrates the process of inserting metadata schema identification information (MSII) into the Fabricator data repository and shows how each step is connected.

- fabricator_data/researcher: This is the grid located at the top, and the “MSII” classification is added starting from this grid. “MSII” represents metadata schema identification information.

- create external: It is connected to the “fabricator_data/researcher” grid. It is responsible for creating the “MSII” classification externally.

- researchers: It is connected to the “create external” gear. This grid has a “MSII” classification.

- create view legacy_researchers: In the “researchers” grid, it is connected to the two gears “legacy 1” and “legacy 2”, each with the “MSII” classification, to create the “legacy_researchers” view. Views created through the ”create view” operation aggregate all “MSII”.

- legacy_researchers: Therefore, the “legacy_researchers” view integrates and manages all metadata schema identification information generated throughout the system. The “legacy_researchers” view created in this way plays an important role in the data processing and management process. This view allows you to efficiently manage and utilize metadata schema information.

Figure 2-B visually shows the process by which metadata schema identification information (MSII) is updated through Fabricator and shows how each step is connected.

- fabricator_data/researcher: It is the grid located at the top, and this grid is “the data source where MSII first occurs. It can be understood as a database or repository where data related to researchers is stored.

- update external: Connected to the “fabricator_data/researcher” grid, this step is responsible for updating the “MSII” classification through an external update mechanism. This is the stage where update information generated from outside is received and processed.

- researchers: It is connected to the “update external” gear. This grid reflects the updated “MSII” classification. It can be understood as a central location where the latest research data is stored.

- update view legacy_researchers: It is connected to the “researchers” grid. It is responsible for updating the view called “legacy_researchers”. In database terms, a view is a virtual table based on the result set of a SQL statement that can combine and filter data from “legacy 1” and “legacy 2” databases to provide a unified view of legacy researcher data.

- legacy_researchers: The final result, with all updates completed, includes an updated “MSII” classification and can provide refined information by integrating information from current researchers and legacy databases.

Algorithm 1 below is an algorithm that defines the process for managing MSII, and is intended to automate the process of creating or updating legacy views in the database based on given metadata and external information.

Algorithm 1. MSII - Metadata schema identification information

```
procedure start:
        fabricator_path: str fabpath;           // metadata path
        msii_classifcaiton: str msiiclass;      // metadata schema identification classification
        create_external: str external;          // external create path
        leacy_viewcreate: str viewcreate;               // creating legacy views

func_msiiPropated(external):                    // metadata schema identification information propagted
        result = "create view leacy_n(external.researchers) as
                    select leacy_n(external.researchers) as researchers _name, d.name as      department_name
                     from employees e
                     join departments d on e.department_id = d.id";
       return result;

func_msiiUpdated(external):                     // metadata schema identification information updated
        result = "create view leacy_n(external.researchers) as
                     update leacy_n(external.researchers) as researchers _name, d.name as
                     department_name from employees e
                      set employsees = external.researchers d
                  where on e.department_id = d.id";
       return result;

begin:
        str fabpath := input(os_path);
        str msiiclass:= input(add_metadata);
        external := fabpath, msiiclass

        if mclass == 'add_metadata': // creating legacy views type(add, update)
                str viewcreate := msiiPropagted(external)
        else if mclass == 'update_metadata':
                str viewcreate := msiiUpdated(external)
        end if;
        Integrated management of metadata schema identification information()
end:

procedure end;
```

2.2. Sequence Diagram

This sequence diagram represents the workflow from the Sources Engine layer to the Data Fabric Ingestion layer.

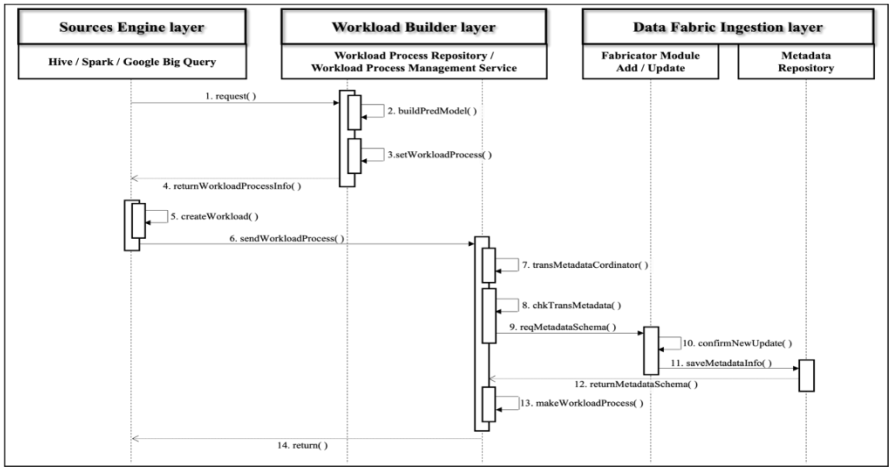


Figure 3. Sequence Diagram of Proposed System

1. request(): This is the step to start the work. It is called from the Sources Engine layer and initializes the work. Request metadata from a database or retrieve data from an external system.

2. buildPredModel(): It is called from the Workload Builder layer and plays a role in building a prediction model. Create a model that predicts specific patterns or trends through data analysis.

3. setWorkloadProcess(): This is the step to set up the work process and is called from the Workload Builder layer. Set or initialize parameters required for work.

4. returnWorkloadProcessInfo(): This is the step of returning the set work process information to the Sources Engine layer. Returns the status and configuration information of the work process.

5. createWorkload(): It is called from the Sources Engine layer and is responsible for creating the workload. Data is processed to create a workload.

6. sendWorkloadProcess(): This is the step to transmit workload process information back to the Workload Builder layer.

7. transMetadataCordinator(): It is called from the Data Fabric Ingestion layer and proceeds with the metadata schema registration process. Reconcile and verify metadata.

8. chkTransMetedata(): This is a step to check the consistency and consistency of metadata transmitted from the Data Fabric Ingestion layer, and confirms the integrity of the data.

- Check that the transmitted metadata is in the correct format.
- Check whether there is duplicate data.
- Check that all required fields are filled in.

9. reqMetadataSchema(): This is the step of requesting the metadata schema, The field structure of the metadata is checked.

- Retrieves schema information necessary to provide structured data representation.
- If necessary, use schema information to process or verify data.

10. confirmNewUpdate(): Called when newly updated information is confirmed, and all modifications are tracked and validated.

- Check whether new updates are reflected accurately.
- Ensures consistency and accuracy of data.

11. saveMetadataInfo(): All metadata-related information is safely stored and data integrity and availability are guaranteed.

- Store metadata information safely in the database.
- Maintain the integrity of data and make it searchable when necessary.

12. returnMetadataSchema(): This is the step of returning the stored metadata schema information and confirms the field structure of the metadata. It is also delivered to the Sources Engine layer as the final result.

- Returns the schema of the saved metadata.
- If necessary, process or verify data using structured data representation.

13. makeWorkloadProcess(): Work processes can be recreated or updated, and new updates can be reflected according to the latest information in stored metadata information.

- Update the work process based on information in stored metadata.
- If necessary, adjust work processes to reflect new updates.

14. return(): The entire process ends and control returns, at which point the workflow is complete.

- Notifies that the work process has ended normally.
- Returns the final result and indicates that the workflow is complete.

3. COMPARATIVE ANALYSIS

Table 1 compares the Proposed System with Data Warehouse, Data Lake, which are important concepts for data management and analysis, based on Data Integration method, Data Processing process, Data Structure, Data Access method, Scalability, and Application Field.

Table 1. Comparison of Systems

	Data Warehouse[6]	Data Lake[7]	Proposed System
Data Integration	Collect and integrate structured data from various sources	Store and integrate unstructured data in raw form	Integrate data in real time from data sources in various formats
Data Processing	Data processing through ETL process	Process stored data directly	Real-time processing without movement or transformation
Data Access	Unsuitable for real-time analytics as data must be processed before it becomes available for complex queries and reports	Optimized for processing large amounts of data, providing fast processing speed	Provides data across your organization's IT infrastructure

Expandability	Requires upfront design and ongoing maintenance	Capable of storing and processing large amounts of data	Capable of integrating large amounts of complex and diverse data
Application Field	Integration of historical data for in-depth analysis and reporting	Optimized field for big data processing	Provides real-time insights, Big data workload processing field

The first comparison item is data integration. Data Warehouse collects and integrates only structured data, which is structured data extracted from relational databases, and Data Lake stores and integrates unstructured data such as text, images, and video in raw form, but the proposed system integrates data in real time from data sources in various formats such as SQL and CSV. Through this, companies can maximize the flexibility and usability of data.

The second comparison item is data processing. Data Warehouse processes data through an ETL (Extract, Transform, Load) process, and Data Lake uses a method of directly processing stored data, but the proposed system processes data in real time without separate movement or transformation.

The third comparison item is data access. Data Warehouse is not suitable for real-time analysis because data must be processed before it becomes available for complex queries or reports, while Data Lake is optimized for processing large amounts of big data and provides fast processing speed, and the proposed system provides data across an organization's IT infrastructure and is also suitable for real-time analysis.

The fourth comparison item is scalability. Data Warehouse requires advance design and ongoing maintenance, and Data Lake can store and process large amounts of data, but the proposed system can integrate large amounts of complex and diverse data.

The last comparison item is the field of application. Data Warehouse is used for in-depth analysis and reporting and is mainly applied in the field of historical data integration, Data Lake is optimized in the field of big data processing, and the proposed system is applicable to providing real-time insights and processing big data workloads. In summary, these are various tools for data management and analysis, each with their own characteristics and advantages. Data Warehouse focuses on structured data, Data Lake focuses on unstructured data, and Proposed System integrates various data sources to provide flexible data management.

Figure 4 is a graph comparing Data Warehouse, Data Lake, and Proposed System by performance. The resource specifications in Table 2 apply. The performance evaluation in this paper was tested in a virtual environment under the same conditions with data sizes of 20, 40, 60, 80, and 100 GB.

Table 2. Resource Specification

Item	Details
Processor	Intel® Xeon® CPU E5-2697 v3 @ 2.60GHz – 8 Virtual CPUs (4 sockets with 2 cores per socket)
Memory	32 GB

Storage	512 GB HDD
Network	1 Gbit/s network card
OS	Ubuntu 16.04 xenial

Figure 4. Comparative evaluation by performance

As shown in the Query response time graph, the query response time of the Data Warehouse increases as the data size increases to approximately 3.11, 5.25, 7.05, 9.01, and 11.07 MB/s, and the data loading speed per unit of speed is approximately 41.23, 51.34, and 60.34 MB/s, and data movement and conversion times increased by 100.54, 90.34, 80.34, and 70.94 MB/h, but in particular, the time for 100GB data increased significantly to 90.34 MB/h. On the other hand, the query response times for Data Lake increased as the data size increased, measuring approximately 5.23, 8.01, 10.02, 12.29, and 15.54 MB/s. Additionally, the data loading speed per unit decreased as the size increased, measuring approximately 85.23, 70.78, 60.12, 50.25, and 47.23 MB/s. The time taken for data movement and transformation increased with the data size, measuring approximately 2.3, 4.43, 6.56, and 8.67 MB/h, However, beyond 100GB of data, it sharply increased to 20.23 MB/h. The query response times for our proposed system are approximately 4.31, 7.54, 9.34, 11.89, and 14.23 MB/s, while the data read/write speeds decreased as the data size increased, measuring approximately 95.98, 82.67, 70.89, 60.67, and 51.34 MB/s. However, the time taken for data movement and transformation increased with the data size, measuring approximately 1.02, 2.25, 3.34, 4.91, and 5.45 MB/h. Interestingly, there wasn't a significant increase beyond 100GB compared to other systems. As a result, as data size increases, query response time and data read/write speed tend to decrease across all data technologies. Data warehouses have relatively high query response times and data read/write speeds, but do not provide data movement and transformation times. Data lakes tend to have very long data movement and conversion times, and Data Fabric shows a tendency for query response times and data movement and conversion times to increase, but data read and write speeds are relatively stable.

4. CONCLISION

In this paper, we propose a design system for metadata of Data Fabric, which plays a key role in the data orchestration layer. DW is used for in-depth analysis and reporting based on structured data, and is mainly used to integrate historical data. DL focuses on unstructured data and is optimized for processing large amounts of data, so it provides fast processing speed and is suitable for big data processing. On the other hand, the proposed Fabircator architecture can maximize data flexibility and usability by integrating various types of data sources in real time. In addition, it can be applied to various big data workload processing fields by

providing real-time insight. Therefore, companies should be able to choose a data management and analysis system that suits their own needs and goals. These choices have a significant impact on the success and growth of an organization, and selecting an appropriate system can promote data-based decision-making and innovation. As mentioned earlier, in Figure 2, Fabircator provides flexibility by integrating data in various formats in real time. Future research requires technical improvements to provide faster processing speed and real-time insights.

ACKNOWLEDGMENT

※ This paper was supported by the Kwangwoon University Research Grant of 2024.

REFERENCES

[1] X. Li, M. Yang, X. Xia, K. Zhang, and K. Liu, “A Distributed Data Fabric Architecture based on Metadate Knowledge Graph,” 2022 5th International Conference on Data Science and Information Technology (DSIT). IEEE, Jul. 22, 2022
DOI: https://doi.org/10.1109/DSIT55514.2022.9943831

[2] A. Abu Rumman and L. Al-Abbadi, “Structural equation modeling for impact of Data Fabric Framework on business decision-making and risk management,” Cogent Business & Management, vol. 10, no. 2. Informa UK Limited, May 21, 2023
DOI: https://doi.org/10.1080/23311975.2023.2215060

[3] N. G. Kuftinova, O. I. Maksimychev, M. Yu. Karelina, A. V. Ostroukh, and M. I. Ismoilov, “Data Fabric Digital Array Processing in Road Transport Systems,” 2022 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex (TIRVED). IEEE, Nov. 10, 2022.
DOI: https://doi.org/10.1109/TIRVED56496.2022.9965462

[4] A. Flizikowski, E. Alkhovik, M. Munjure Mowla, and M. Arifur Rahman, “Data Handling Mechanisms and Collection Framework for 5G vRAN in Edge Networks,” 2022 IEEE Conference on Standards for Communications and Networking (CSCN). IEEE, Nov. 28, 2022
DOI: https://doi.org/10.1109/CSCN57023.2022.10051118

[5] N. G. Kuftinova, O. I. Maksimychev, A. V. Ostroukh, A. V. Volosova, and E. N. Matukhina, “Data Fabric as an Effective Method of Data Management in Traffic and Road Systems,” 2022 Systems of Signals Generating and Processing in the Field of on Board Communications. IEEE, Mar. 15, 2022
DOI: https://doi.org/10.1109/IEEECONF53456.2022.9744402

[6] J. Praful Bharadiya, “A Comparative Study of Business Intelligence and Artificial Intelligence with Big Data Analytics,” American Journal of Artificial Intelligence. Science Publishing Group, Jun. 27, 2023
DOI: https://doi.org/10.11648/j.ajai.20230701.14

[7] R. Hai, C. Koutras, C. Quix, and M. Jarke, “Data Lakes: A Survey of Functions and Systems,” IEEE Transactions on Knowledge and Data Engineering. Institute of Electrical and Electronics Engineers (IEEE), pp. 1–20, 2023
DOI: https://doi.org/10.1109/TKDE.2023.3270101