Applied Mathematics and Computational Intelligence
Volume 15, No. 2, 2026 [75-91]

Design and Implementation of Data Lakehouse Architecture for Self-Service Analytics

Soon Kien Yuan[1], Nor Azuana Ramli[2*], Mohd Zaid Waqiyuddin Mohd Zulkifli[3]

[1,2]Centre for Mathematical Science, Universiti Malaysia Pahang Al-Sultan Abdullah, Lebuh Persiaran Tun Khalil Yaakob, 26300 Kuantan, Pahang, Malaysia.
[3]Credence, 1 Jalan Damansara, Damansara Kim, 60000 W.P. Kuala Lumpur, Malaysia.

[*] Corresponding author : azuana@umpsa.edu.my

Received: 19 June 2025
Revised: 4 October 2025
Accepted: 6 October 2025

ABSTRACT

This paper focused on designing data lakehouse architecture for self-service analytics. The objectives include creating a collaborative analytics environment, streamlining the management of multiple extract, transform and load (ETL) processes, adopting cost-effective and non-proprietary architecture, integrating with business intelligence (BI) tools, ensuring high query performance for interactive visualization, enabling data warehousing capabilities, and offering a self-service data discovery and metadata platform. An iterative development methodology that involved requirement gathering and planning, design, implementation, testing, deployment, and maintenance phases was utilized in this research. The logical design comprises six layers: data ingestion, storage, catalog, semantics, processing, and consumption. For physical design, Dremio was used as the core component, while Apache Iceberg was used for data format and query processing. The case study presented in this paper adopted an Integrated Multi-Zone Analytics Framework to handle data tasks and workloads. As this paper concludes, it suggests future enhancements, such as considering the Dremio Enterprise Edition for advanced features, and exploring Databricks and MLflow if expecting extensive machine learning workloads. These enhancements can further improve the architecture and its outcomes.

Keywords: architecture, cloud, data analytics, data lakehouse, data management.

1 INTRODUCTION

The world has long benefited from data warehouses and data lakes, each offering distinct advantages but presenting certain drawbacks. As a result, data architects and end users have envisioned a powerful unified system for storing and processing data across various use cases, leading to the invention of the data lakehouse [1, 2]. Data warehouses have been in use for decades, dating back to the introduction of the data warehousing concept by International Business Machines Corporation (IBM) [3]. They provided a single source appearance for data residing in different locations, allowing end users to focus on decision-making rather than data retrieval and maintenance [4]. However, with the exponential growth of unstructured data, especially with the advent of the Internet, data warehouses faced limitations. They struggled to accommodate semi-structured or unstructured data,

needed more scalability and flexibility in managing large data volumes (petabytes), and incurred significant maintenance expenses for large IT projects [5].

In response to the increasing need for processing data in diverse formats, data lakes emerged as a promising platform capable of managing vast amounts of data in various formats. A data lake is a cost-effective data repository that improves an organization's ability to capture, refine, archive, and explore raw data while preserving it in its native format. Data lakes eliminate the need for complex pre-processing and transformations before loading data into data warehouses, resulting in reduced data ingestion costs. Anyone within the company can access and analyze it on a schema-on-read basis once data is loaded into the data lake. However, data lakes also present limitations and challenges. A data lake can deteriorate into a data swamp, impacting data quality, governance, and retrieval without adequate data curation procedures [6]. Data stewardship, governance, analytical skills, data quality, and data retrieval are some of the difficulties associated with data lakes. Data lakes need more transactional capabilities, struggle to ensure data quality, and face challenges in merging appends, reads, batch, and streaming operations due to a lack of consistency and isolation [7].

Many of the benefits promised by data lakes have yet to be fully realized, which has led to a loss of the advantages previously offered by data warehouses. To address the need for a powerful integrated system capable of storing and processing high volumes of data in various formats for multiple industrial applications, organizations often use multiple systems, including data lakes, data warehouses, and NoSQL databases [7]. However, such multiple-system data architectures are expensive and require continuous data synchronization, which undermines the goal of low-cost storage and prioritizes decision-making over data engineering [8]. A new solution to these limitations and challenges is emerging in the form of data lakehouse architecture, designed to cater to the needs of self-service analytics. According to Talend (now known as Qlik) [9], a data lakehouse is a unique data architecture that combines the data quality and governance requirements of a data warehouse with the capability to store all types of data in a data lake, including unstructured, semi-structured, and structured data. Oracle highlights that a data lakehouse eliminates the silo barriers between a data lake and a data warehouse, facilitating seamless data movement between the low-cost and flexible storage of a data lake and the schema and governance management tools of a data warehouse. Machine learning algorithms are frequently employed for data cleansing in this architecture [10].

In the context of self-service analytics, data lakehouse architecture provides a solid foundation for empowering users to perform advanced analytics on diverse data sets while ensuring data quality, governance, and flexibility. It bridges the gap between data lakes and data warehouses, offering an integrated solution that supports efficient decision-making and reduces the complexities associated with traditional multiple-system data architectures [11]. In this paper, a data lakehouse architecture that fosters a collaborative analytics environment while streamlining the cumbersome process of managing multiple ETLs was constructed using the Dremio software. The whole methodology will be discussed in the next section before the finalized architecture is presented in the results and discussion section. Finally, this paper will reach its conclusion in the subsequent section.

2 MATERIAL AND METHODS

The concepts from the system and software development life cycles were borrowed and used for the methodology. Rapid Application Development (RAD) was used as the research methodology because the data lakehouse architecture for self-service analytics required continuous improvement over

time and the incorporation of valuable user feedback to align with the rapid changes in demand. Hence, one of the RAD applications called "iterative development" was applied in this research. Figure 1 illustrates the iterative development adopted in this study. There are six phases for the research methodology: requirement gathering and planning, design, implementation, testing, deployment, and maintenance.

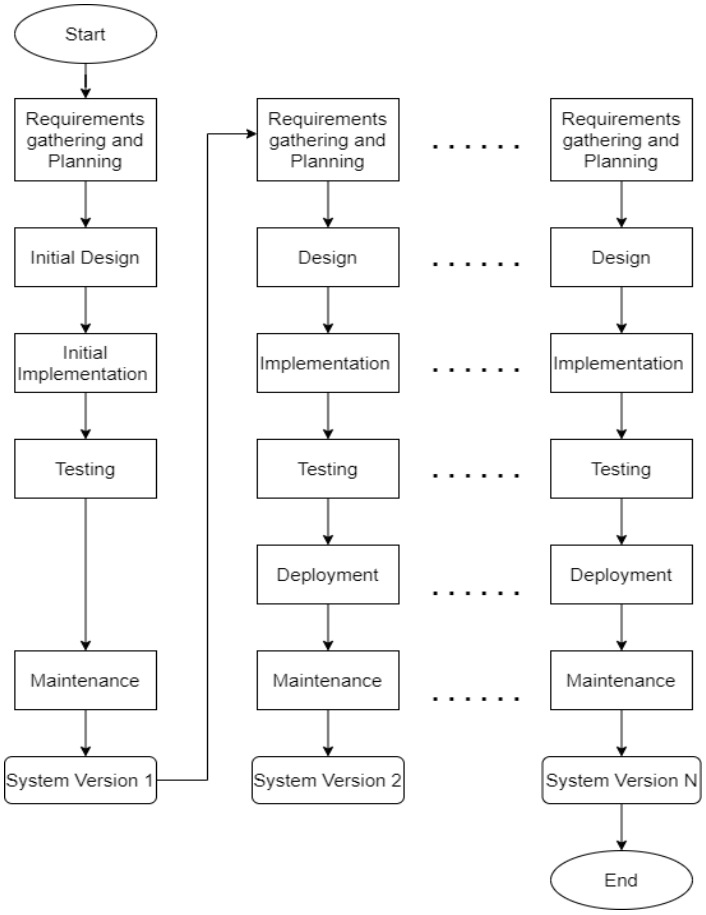

Figure 1 : Iterative development

2.1 Requirement gathering and planning

The first phase of iterative development involves gathering and planning. As the title suggests, the lakehouse architecture is designed for self-service data analytics. Therefore, self-service data analytics is the primary objective, with a user-friendly interface and interactive graphical buttons enabling users to perform analytics with fewer clicks and less code. After identifying the purpose of the proposed architecture, the next step is to identify the project requirements through communication with teams, stakeholders, and other relevant parties. The team can effectively communicate the project's scope, functionality, and performance requirements for the proposed data lakehouse architecture. [12] stated that a project's activities and concerns can be classified through size, cost, purpose, length, risk, scope, and economic value.

2.2 Design

In the second phase, which is the design phase, the gathered information, planned considerations, and description will be transformed into a logical design, followed by a physical design. The initial design will be utilized in the initial version for the initial implementation. Logical design, a crucial part of the design process, is platform-independent, meaning it can adapt to any hardware or software platform. This adaptability allows the logical design to focus on the architecture's functional and operational elements. Its goal is to provide a detailed understanding of how the lakehouse functions and how it can improve data management compared to traditional methods. Once the logical design of the lakehouse architecture is determined, the transformation of the logical design into a tangible implementation is referred to as physical design [13]. The programming language, platforms, and other tools will be chosen to align with the logical design. The final output of the design phase will be the lakehouse architecture specification in a physical form, ready for implementation.

2.3 Implementation

The designed physical lakehouse architecture specification will turn into a working architecture system in the implementation phase. According to [13], implementation may include coding, testing, installation, and initial user support such as documentation, training programs, and ongoing user assistance. The initial implementation is utilized in the initial version to meet dynamic business environment deadlines and the environment because it may take longer between the original idea and the actual implementation. Iterative development is preferred for this project to develop a portion of the system quickly and put it in the hands of users for evaluation and feedback [12].

2.4 Testing

Throughout the testing phase, the team collaboratively examines the architecture system to identify and correct any errors that may have been introduced. These errors could have occurred at any stage during the implementation process. The goal is to ensure that the lakehouse architecture adheres to the requirements and functions as intended (proof of concept). Testing will take place after the installation of the lakehouse (as part of the implementation phase), which means this phase will occur subsequent to configuring the data lakehouse architecture.

2.5 Development and Maintenance

Following the completion of the testing, the system is now in a state where it is prepared for deployment as well as production. The deployment process is critical to ensure that the software runs effectively in the production environment, with minimal downtime, and is available to the end-users. The final phase in iterative development is maintenance, which is an ongoing process of improvement and adaptation to study the situation and make necessary adjustments [13]. Once a company has implemented a system and provided users with the required resources, such as training, documentation, and support, users will inevitably find flaws in the system and develop ways to improve its usage. Additionally, the organization's system requirements evolve over time. Programmers implement user-requested modifications during system maintenance to improve the system over time.

3 RESULTS AND DISCUSSION

This research was completed independently, and the environment setup took place on a personal laptop, specifically the Dell G15 5515 Ryzen Edition. The laptop specifications are as follows:

a) The processor is an AMD Ryzen 7 5800H with Radeon Graphics, operating at 3201 MHz, with eight cores and 16 logical processors.

b) The installed physical memory, or RAM, is 16.0 GB.

c) The operating system is Windows 11 Home Single Language 22H2.

There is no cost encountered in this research as the environment was set up on a personal laptop. Free trial versions of Azure, Amazon Web Services, and Google Cloud Platform were utilized for cloud data lake storage.

3.1 Design

The process of designing architecture was done in phases. The first phase is logical design, where the process does not involve any hardware or software platform in this part. The proposed architecture's logical design consists of six layers: data ingestion, storage, catalog, semantics, processing, and consumption. The proposed architecture is expected to be able to connect to multiple data sources, with lakehouse serving as a central repository where all data is accessible for the subsequent process phase. Relevant data sources include line-of-business (LOB) software, enterprise resource planning (ERP), customer relationship management (CRM), online applications, mobile devices, sensors, video feeds, and social media. Then, the data will be ingested into the ingestion layer before it goes to the storage layer. Ideally, the ingestion layer can receive raw data in native format from the data sources, where the ingested data can be ingested and delivered in batches, and real-time streaming data or hybrid data into the storage layer. The ingestion layer can ingest and deliver structured, semi-structured, and unstructured data into the storage layer.

The storage layer of the lakehouse architecture, a crucial element, provides reliable, flexible, and cost-effective components for storing, processing, and securing vast quantities of structured, semi-structured, and unstructured data. As suggested by [14], it should be designed using cheap, readily available storage while incorporating the standard administration and performance features of an analytical DBMS. The catalog layer, a shared and integral part of the storage layer, plays a significant role. It stores business and technical metadata about datasets kept in the storage layer, making it searchable and enabling users to discover data in a lakehouse. The ideal scenario is a unified lakehouse interface, where the storage layer and the catalog layer are shared, as proposed by [15]. This provides a centralized, searchable catalog that indexes the metadata of all datasets stored in a data lake, simplifying the management and administration of datasets.

The semantic layer is constructed based on the catalog layer, providing an abstraction layer for business users to interact with objects produced in the semantic layer without concerning themselves with the physical location or organization of the data [16]. This abstraction simplifies the representation of the underlying data, making analytics more accessible while abstracting its complexities. Additionally, it enables multi-zone analytics by incorporating data into staging,

unprocessed, business, and application zones. The processing layer's responsibility is to ensure that the data is prepared for use by the consumption layer. It encompasses checking, cleaning, normalizing, transforming, and enhancing the data [15]. Within the processing layer, users should be able to process the data within the lakehouse platform or integrate it with other third-party platforms or tools, such as Python and Apache Spark. As its name suggests, the consumption layer consumes the output or data provided by the processing layer. The processed and curated data can be integrated with third-party business intelligence platforms like Tableau and Power BI for user consumption. The lakehouse can also integrate with data science notebooks like Jupyter Notebook and Google Collab to perform various analytics and machine learning tasks. Furthermore, customers can interface their custom software applications with the lakehouse to conduct business operations.

The design, inspired by [17] and Microsoft Corporation, is adaptable and references physical design. It was created for use on the Azure Cloud Platform, with the adaptability to integrate with Dremio and Databricks. While Databricks is primarily available via cloud services, the architecture is designed to meet the needs of organizations that require on-premises solutions due to regulatory compliance, data sovereignty concerns, or sensitive data retention. In these situations, Dremio becomes an attractive and reassuring option. Figure 2 illustrates the proposed architecture, which is designed with five components: data sources, data processing and virtualization layer, processing layer, integrated metadata layer, and consumption layer.

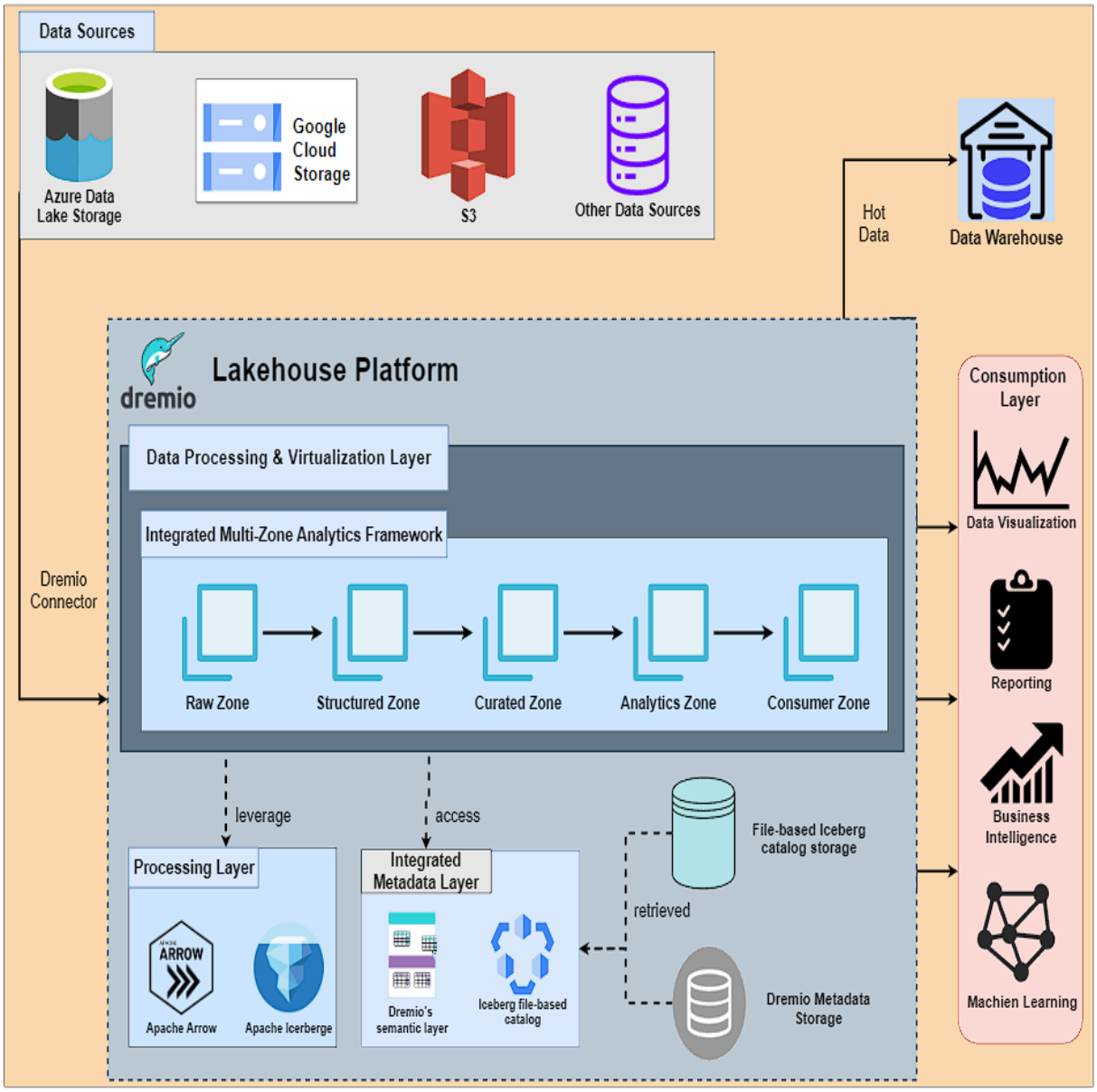

Figure 2 : Proposed data lakehouse architecture

First, the proposed architecture encourages open platforms and vendor neutrality, allowing for integrating various data sources. In this instance, the Dremio connector connects Azure Data Lake Generation 2, Amazon S3, and Google Cloud storage. Connecting to additional data sources, including Hadoop Distributed File System (HDFS), Microsoft SQL Server, PostgreSQL, Snowflake, and others, is possible. Dremio acts as a data processing and virtualization layer that connects to these storage systems or data sources and provides a unified interface for data access and querying after establishing the connection. Dremio treats data sources as a storage layer by abstracting the underlying complexities of various storage systems, data formats, and query languages. This provides a consistent and streamlined user interface to retrieve and process the data. The cleaned and

transformed data can be moved and ingested into the data warehouse. Inside the data processing and the virtualization layer, the Integrated Multi-Zone Analytics Framework was used for better data processing and administration. The five zones that comprise the Integrated Multi-Zone Analytics Framework are the Raw Zone, Structured Zone, Curated Zone, Analytics Zone, and Consumer Zone. The zones were created by leveraging Dremio's Space functionality. Each zone serves a specific purpose in the data analytics workflow. To distribute responsibilities and streamline the overall data processing and analysis, the tasks are assigned to team members for each zone. Table 1 tabulates a brief description of each zone and its corresponding responsibilities:

Table 1 : Brief description of each zone

Zones	Purpose	Tasks involve
Raw Zone	Store the raw and unprocessed data (virtual datasets)	Identifies the data needed, creates virtual datasets from physical datasets, and configures the table format.
Structured Zone	Raw data is transformed into a structured format suitable for analysis	Data cleansing, data normalization, schema creation, ensuring data consistency, ensuring data integrity, data validation, data format, and data partitioning.
Curated Zone	Refined, prepared, and enriched the data for further analysis and usable across multiple use cases	Data modelling, data governance practices, data integration, data transformation (join, merge)
Analytics Zone	Dedicated to conducting exploratory data analysis and implementing advanced analytics techniques.	Statistical analysis, data mining, machine learning (Forecasting, Clustering, and Classification), and recommendation.
Consumer Zone	Final zone where the analysed data and insights are made available to the end users or consumers.	Dashboards, visualizations, reports, and data-driven decision-making.

Users can create and customize numerous zones/spaces to meet their needs. These spaces serve as organizational units, connecting datasets and resources. Within each space, users can create virtual datasets, which not only provide logical views but also offer transformations to the underlying physical data. Folders offer a hierarchical structure for both physical and digital datasets. The physical datasets, which are the real data stored in the sources, form the basis for virtual datasets. Importantly, changes to a virtual dataset do not affect the data sources or physical datasets. Figure 3 provides a visual representation of the dataset and space views.

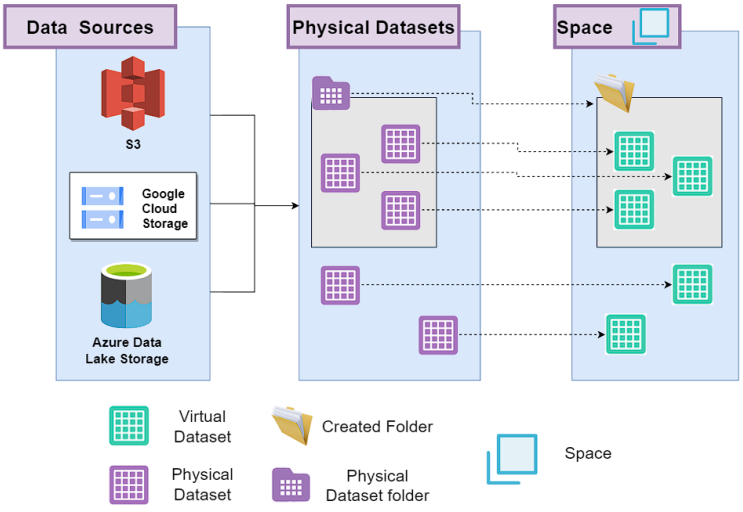

Figure 3 : Dataset and Dremio's Space

Next, Dremio is an originally columnar engine that Apache Arrow powers alongside Gandiva LLVM-based execution kernel, which supports the open-source standard columnar in-memory computing and query optimization. Apache Iceberg was utilized in the processing layer so that the Atomic, Consistent, Isolated, and Durable (ACID) capabilities can be achieved. The capabilities of Apache Arrow and Apache Iceberg were combined to implement the lakehouse architecture. The catalog and semantic layers were combined and incorporated into the proposed physical design to create an integrated metadata layer. The native Dremio semantic layer and file-based Iceberg catalog supports the integrated metadata layer. With the integrated metadata layer, the objective of implementing a self-service data discovery and metadata platform, serving as the single source of truth within the data lakehouse architecture powered by Dremio and Apache Iceberg, can be achieved. File-based Iceberg catalog storage was used to store the file-based Iceberg catalog metadata, whereas Dremio Metadata storage was used to store the semantic layer metadata. The final layer, consumption, consumes the output or data supplied by the processing layer or integrated multizone analytics framework. The data that has been processed and curated before being used in the consumption layer for analytics workloads such as reporting, data visualization, business intelligence, and machine learning.

3.2 Implementation

Based on Figure 2, Confirmed Physical Design, data sources such as Google Cloud Storage, Amazon S3, and Azure Data Lake Storage Gen 2 are seamlessly connected to Dremio using the Dremio connector. To connect all these data sources to Dremio, it is necessary to configure and register Google Cloud Storage, Amazon S3, and Azure Data Lake Storage Gen 2. The FIFA 23 complete player dataset was used to demonstrate data integration in this case study. In the subsequent phase, testing demonstrates that the proposed architecture system embraces and supports open platforms and vendor neutrality, allowing multiple data sources to be integrated by connecting to storage from three distinct vendors. It allows consumers to access data from multiple systems without being limited to a single vendor or service, providing a reassuringly smooth integration process.

3.3 Testing

The implemented architecture, Dremio, served as the central repository, equipping team members with self-service capabilities by allowing them to explore and analyze the data independently. It enables collaboration between team members by letting team members share queries, virtual datasets, and insights. With its easy-to-use interface, drag-and-drop features, and automated tool for building SQL queries, team members can easily access and analyze data without having to depend on data engineering support too much. The proposed integrated multi-zone analytics framework enables seamless collaboration and increased productivity among data teams by assigning distinct zones to different data team members according to their expertise. For example, data engineers are responsible for the Raw Zone and Structured Zone, ensuring the selection of relevant data and transforming it into a structured format suitable for analysis.

During Curated Zone, data analysts and data scientists enrich the data, making it usable for various use cases and scenarios. Within the Analytics Zone, machine learning engineers and data scientists work closely together, conducting exploratory data analysis and implementing advanced analytics techniques such as machine learning and data mining. Lastly, in the Consumer Zone, the analyzed data and insights are accessible to end users and stakeholders. The team presents the findings through interactive dashboards, visualizations, and reports, ensuring the delivery of meaningful insights to the relevant stakeholders. Dremio works with many of the most popular data tools and platforms, which makes it easy for people to work together within current data ecosystems. It works with BI tools, like Power BI and Tableau, as well as other data science platforms. Data team members can work together using their favorite tools while taking advantage of Dremio's features.

3.3.1 Streamlining the cumbersome process of managing multiple ETL

In modern architectures, the data lake is a central repository for raw and unprocessed data, enabling data ingestion and capture from multiple sources without needing upfront schema design or data transformation. Since the data lake is inefficient for analytics and querying, ETL into a data warehouse was required. This involves applying structured schemas and other transformations to the data to make them suitable for analysis. This extra ETL step is needed to ensure that the data warehouse data is well-organized, clean, and optimized for fast queries. However, this two-tiered architecture adds complexity, delays, and new failure possibilities [13]. Managing two separate ETLs or ELTs adds time to the data processing pipeline, which may delay data availability for analysis. The possibility of failure is higher since multiple ETL processes are involved at different stages. It indicates that the efforts to ensure data consistency across the two-tiered architecture is highly challenging, as it creates multiple data replications, causing data redundancy.

The proposed lakehouse architecture reduces the cumbersome process of managing multiple ETL processes. Firstly, the Dremio connector connects to the data sources (data lake), the data available in the Dremio UI. By leveraging the proposed Integrated Multi-Zone Analytics Framework and Dremio's Space, data teams can perform data transformations and define virtual datasets directly within the platform from the data sources' physical datasets, eliminating the need for separate ETL processes. It provides a unified interface for effectively managing and querying data from multiple sources, fostering a unified approach to data management. The Integrated Multi-Zone Analytics Framework provides a systematic and organized approach to data management and analytics. Data

teams can define data transformations within each zone, apply business rules, and create new virtual datasets from zone to zone without multiple ETL processes. It reduces data replication and simplifies the data management process by providing a unified environment for data teams to work together, share transformations, and move data seamlessly across the different zones, ultimately reducing complexity, promoting collaboration, and improving the overall efficiency of the architecture.

By leveraging the Apache Iceberg format, direct data analysis on data lakes can be achieved using the proposed architecture that utilized Dremio to reduce the need for multiple ETL processes that involve separate systems (data lake and data warehouse). With Apache Iceberg, storing data in a structured format (an Iceberg Table) directly within the data lake is possible. Data teams can directly analyze the data lake without additional data movement or transformation. Since Iceberg Table is an open format, it facilitates seamless integration and compatibility with different data processing frameworks, including the Dremio used in the proposed architecture, providing a unified interface for querying and analyzing data. This combination empowers data teams to perform direct analysis on the data lake and other data sources, eliminating the need for separate data warehouses and simplifying the overall architecture.

In terms of cost, the proposed architecture utilizes the Dremio community version, which offers a cost-effective solution as users are only required to pay for the virtual machine (VM) prices. The community version of Dremio is free to use and provides access to the platform's essential features and capabilities. Users can leverage Dremio's community edition on their virtual machines, eliminating the need for additional licensing or subscription fees. This cost structure makes Dremio an affordable option for organizations seeking a non-proprietary architecture that aligns with their cost-effective data management requirements. Dremio's cost-effectiveness also comes from its ability to eliminate the need for expensive data replication and storage. Providing a virtualization layer allows data teams to work directly with the underlying physical datasets, avoiding duplicating and storing data in separate systems. This significantly reduces storage costs and overall infrastructure requirements. Furthermore, Dremio's non-proprietary architecture ensures vendor independence and flexibility. It supports various data sources and formats, enabling seamless integration with existing data ecosystems. Data teams can leverage Dremio's open architecture to connect and query data from various sources without being tied to a specific vendor or technology stack. This promotes interoperability, avoids vendor lock-in, and allows organizations to choose cost-effective solutions that best fit their needs. The Apache iceberg table is utilized in this architecture, which is another example of a non-proprietary architecture for data management. It is an open-source table format that provides a consistent and reliable structure for managing large-scale data sets.

The key advantage of Apache Iceberg is its ability to work with multiple data processing engines, such as Apache Spark, Apache Hive, Presto, and Dremio [18]. This flexibility allows organizations to choose the most suitable processing engine for their specific needs without being locked into a proprietary solution. Being an open-source project, it encourages community collaboration and contributions. This ensures the technology remains non-proprietary, free from any specific vendor or proprietary limitations. Using the Apache Iceberg with Dremio, data teams perform direct data analysis on the data lake storage without additional data movement or transformation. Apache Iceberg's features, such as schema evolution, hidden partitioning, time travel, and rollback within the

Dremio platform, organizations can achieve a cost-effective solution that replaces the need for a separate data warehouse, as Iceberg provides ACID functionality and enables efficient data management and analysis, reducing the costs associated with extensive ETL processes, separate storage systems, and complex versioning requirements [19].

3.4 Deployment

After testing, the system is now ready for deployment and production. In this phase, a containerized version of Dremio was deployed on the Azure cloud using the Azure Kubernetes Service (AKS). A Kubernetes cluster named "DSP_deploymentDremio" was created under the resource group "Dremio_dsp." The Kubernetes version utilized is 1.25.6. The primary node pool consists of 2 vCPUs and 8 GiB memory, employing the standard B4ms node size. Autoscaling is enabled in the node pool, allowing it to scale from 1 node to 2 nodes dynamically. The operating system type for the cluster is Linux. For cluster networking, the Kubernetes networking plugin is configured. This results in creating a new VNet for the cluster, utilizing default values and standard load balancer traffic routing. To ensure the security of the Kubernetes cluster, the Defender for Cloud is configured under the basic plan. This security solution continuously monitors the configuration of the created Kubernetes services, identifying potential security vulnerabilities and providing recommendations for mitigation.

However, to deploy Dremio on the AKS cluster, the minimum worker node instance type is E16s_v3 (16 core, 128 GiB memory, 400 GiB SSD, and 32 GB disk size), which is not supported under the Azure for student subscription. Since limited resources and budgets exist to deploy the environment, the following configuration and discussion are referenced in the Dremio documentation. Configuring Dremio for distributed storage is essential to leverage the full potential of the platform's data processing capabilities (Dremio, n.d. -d). Dremio can efficiently handle large volumes of data and parallelize data processing tasks by distributing the storage across multiple nodes or servers. This enables faster query execution, improved scalability, and fault tolerance. Additionally, the Dremio version started from 21.xx, so configuring for distributed storage is necessary.

In order to maintain the continuous availability of Dremio, it is crucial to configure two or more coordinator nodes with the master coordinator role. Among these nodes, one is designated the active coordinator, responsible for system operations. The remaining nodes are initialized but remain in a standby state, ready to take over the active coordinator role in the event of failure or disappearance of the active coordinator. This setup guarantees uninterrupted system availability. Additionally, assigning one or more executor nodes to each coordinator node is possible, ensuring optimal system performance and faster processing of tasks. By implementing this configuration, Dremio is designed to consistently deliver optimized performance while maintaining high availability for critical data operations. Other configurations, such as ODBC/JDBC port configuration, Power BI connector, Tableau connector, Apache Zookeeper configuration, Metadata Storage configuration and are configured by defaults; changing can configured after the installation of Dremio on AKS.

After all the necessary configurations are completed before deployment, the dremio-cloud-tools can be downloaded using git. Once the clone is complete, navigate to /dremio-cloud-tools/charts/dremio_v2 and look at the values.yaml file. The memory and CPU count for the coordinator and executor need to be changed to match the resources available in the AKS cluster.

Then, the command “helm install dremio-cloud-tools/charts/dremio_v2” is run to deploy Dremio. Once the installation is done, the IP address obtained will be used to access Dremio’s UI.

3.5 Maintenance

Following the deployment of a Dremio cluster, the final phase of the iterative development process is dedicated to maintenance. This phase is essential for ensuring smooth operation, data protection, and optimum performance of the Dremio system. Alongside the previously mentioned tasks, several crucial aspects require consideration. One critical aspect of maintenance is backup management, necessitating the implementation of a robust backup strategy. It is vital to perform regular backups of Dremio’s metadata, configuration files, and critical data sources, securely storing these backups. Conducting periodic restoration tests is essential to ensure data integrity and effective disaster recovery. Another crucial responsibility is safeguarding Dremio’s security. Administrators should regularly check and update security settings, including user access controls, authentication methods, and encryption settings. Immediately installing Dremio’s security updates and patches is crucial for protecting the system from potential vulnerabilities.

User roles and access control are paramount for maintaining data security and privacy. Administrators must define and enforce appropriate user roles, permissions, and access policies, regularly reviewing user access rights to ensure compliance with security policies and to mitigate potential risks. Proper cleaning job results and temporary data is essential for optimizing storage usage and maintaining data cleanliness. Scheduling cleaning tasks to remove temporary or unnecessary data generated by job executions, including clearing intermediate results, temporary files, and outdated metadata, is necessary to free up storage space and ensure efficient resource utilization. Crucially, monitoring the health and performance of individual nodes in the Dremio cluster is indispensable for identifying potential issues and optimizing system performance. Regularly monitoring node-level metrics such as CPU usage, memory utilization, disk I/O, and network traffic allows administrators to identify bottlenecks, resource constraints, and other performance-related issues, enabling proactive troubleshooting and capacity planning. It is also crucial to regularly update and monitor the nodes to ensure no visible user interruption or query failure. Furthermore, system telemetry monitoring, which involves tracking system-wide metrics to gain insights into overall system performance, stability, and resource utilization, is essential. Monitoring metrics such as query execution time, job success rates, system uptime, and system-level resource consumption is crucial for identifying trends, areas for improvement, and opportunities for optimizing system performance.

In cloud environments like Azure Kubernetes Service (AKS), monitoring cluster usage and resource utilization becomes necessary. Administrators should make use of cloud-native monitoring tools to track cluster metrics such as CPU and memory usage, pod health, and network traffic. This facilitates effective resource scaling based on usage patterns, ensuring optimal performance and cost efficiency. Monitoring cost usage is crucial for managing the operational costs associated with Dremio deployments. Administrators should track and analyze the cost implications of various components, including storage, computer resources, and data transfer, to identify cost optimization opportunities, such as optimizing data storage formats or right-sizing compute resources. Fine-tuning the Dremio configuration is another significant maintenance task. Administrators should adjust various settings based on the workload and data characteristics, particularly parameters related to memory

allocation, query execution, caching, and parallelism, to optimize performance. It is also crucial to conduct periodic reviews and optimization of data reflections, which are pre-aggregated or pre-joined subsets of data, to ensure their efficiency and relevance.

Proactive capacity planning is vital to ensure the Dremio environment can accommodate growing data volumes and user demands. Regular monitoring of system usage trends, anticipation of future growth, and adjusting hardware resources, such as CPU, memory, and storage, are necessary to meet the system's expanding requirements. Finally, maintenance also involves providing support and assistance to users. Administrators should be responsive to user queries and issues, offering timely resolutions and assistance as needed. This may include troubleshooting performance problems, assisting with query optimization, and providing guidance or documentation on the best practices.

4 CONCLUSION

This paper proposes a lakehouse architecture that utilizes Dremio and Apache Iceberg to create a collaborative analytics environment for data teams. It allows team members to independently explore and analyze data from various sources and formats, and share their queries, virtual datasets, and insights with others. It also offers an easy-to-use interface and automated tools for building SQL queries, reducing the need for data engineering support. The lakehouse architecture implements a Multi-Zone analytics framework that assigns different zones to different data team members based on their expertise. This promotes seamless collaboration and increased productivity among the team members. It also enables data teams to perform data transformations and define virtual datasets directly within the platform, eliminating the need for separate ETL processes. The framework simplifies data management, reduces data replication, and improves overall efficiency.

The Lakehouse architecture is cost-effective as it utilizes Dremio's community version, which does not require additional licensing or subscription fees. It also reduces storage costs by avoiding expensive data replication and storage. The architecture is vendor-independent and flexible as it uses the non-proprietary architecture of Dremio and Apache Iceberg. The architecture also enables data warehousing and ACID transaction support by leveraging Dremio's integration with Apache Iceberg. The architecture allows for data integration, ACID transactions, partitioning, and indexing, providing efficient data management and analysis. Dremio provides a self-service data discovery and metadata platform that simplifies data exploration and collaboration. This project concluded by recommending the hardware requirements and available resources for future deployments, and exploring alternative deployment options or adjusting the infrastructure to meet the necessary specifications. The project also suggested further evaluating the proposed architecture's feasibility, scalability, security, and performance.

Dremio Community Edition is a powerful self-service analytics platform that enables users to explore and analyze data from various sources. While it offers several advanced features and benefits, there are limitations. One of the limitations of Dremio Community Edition is the absence of Fine-Grained Access Control. This feature enables data teams to define granular permissions and roles for different datasets and resources. Without this level of control, there is a risk of unauthorized access and modification of sensitive data. Fine-grained access Control is crucial for ensuring data security and privacy while facilitating collaboration among team members. By allowing organizations to define

specific access privileges based on roles and responsibilities, Fine-Grained Access Control helps mitigate the potential risks associated with data misuse or unauthorized access.

Another important limitation of Dremio Community Edition is the lack of built-in Data Lineage functionality. Data lineage is a critical feature for tracking the flow and history of data across different systems and stages. It provides valuable insights into the origin, transformations, and destinations of data, facilitating data quality assessment, compliance, and auditing. Visual representations of data lineage, such as diagrams or graphs, help users understand the context and reliability of data. Without data lineage capabilities, users may face challenges in comprehensively tracking and verifying the data they are working with, potentially impacting data trustworthiness and compliance efforts. While the proposed architecture does not specifically emphasize machine learning, it is worth noting that the Dremio Community Edition has limited built-in support for machine learning workflows. Although Dremio can integrate with external tools like Jupyter Notebook, it does not provide native machine learning capabilities within its core functionality. Additional integration and customization may be required for organizations seeking a comprehensive analytics solution that includes advanced machine learning capabilities. This limitation should be considered when evaluating Dremio's suitability for projects where machine learning plays a central role.

Organizations can consider upgrading to the Dremio Enterprise Edition to overcome the limitations mentioned. The Enterprise Edition offers advanced features, including Fine-Grained Access Control and Data Lineage, which are essential for enhanced data security, compliance, and data flow tracking, as shown in Figure 4. By upgrading, organizations gain access to more comprehensive access control mechanisms and the ability to track data lineage, ensuring better governance and data quality.

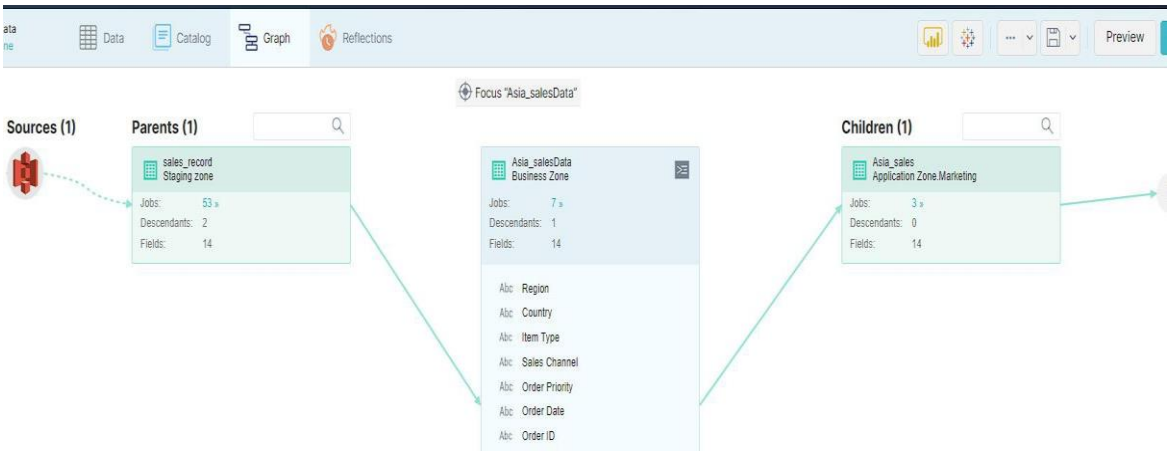

Figure 4 : Dremio Enterprise Version's data lineage

Organizations heavily focused on machine learning should consider evaluating dedicated machine learning platforms that integrate well with Dremio. They should seek platforms with seamless integration, robust machine learning capabilities, and compatibility with Dremio's data sources. Databricks with MLflow is a recommended option [20]. For future study recommendations, a few case studies should be implemented using the proposed architecture since this study relies on the proof of concept to validate the architecture. From the case studies, a quantitative comparison against a direct performance baseline or a primary competitor can be done to provide a rigorous and

generalizable proof. Other validation such as user satisfaction scores or user adoption rates can be added as well to strength the advantages of the proposed architecture.

REFERENCES

[1] Microsoft, "Lakehouse overview," Microsoft Learn, 2025. [Online]. Available: https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-overview

[2] D. Mazumdar, "What is a Data Lakehouse & How does it Work?," Apache Hudi, 2024. [Online]. Available: https://hudi.apache.org/blog/2024/07/11/what-is-a-data-lakehouse/

[3] J. Holdsworth and M. Kosinki, "What is a data warehouse?," IBM, 2024. [Online]. Available: https://www.ibm.com/think/topics/data-warehouse

[4] B. A. Devlin and P. T. Murphy, "An architecture for a business and information system," IBM Syst. J., vol. 27, no. 1, 2010. doi: 10.1147/sj.271.0060.

[5] J. A. Hoffer, V. Ramesh, and H. Topi, Modern Database Management, 11th ed. Pearson Education, 2016.

[6] P. P. Khine and Z. S. Wang, "Data lake: A new ideology in big data era," ITM Web Conf., vol. 17, p. 03025, 2018. doi: 10.1051/itmconf/20181703025.

[7] U. Jagare, The Data Lakehouse Platform for Dummies: Databricks Special Edition. John Wiley & Sons, Inc., 2022.

[8] J. Schneider, C. Gröger, A. Lutsch, et al., "The Lakehouse: State of the Art on Concepts and Technologies," SN Comput. Sci., vol. 5, p. 449, 2024. doi: 10.1007/s42979-024-02737-0.

[9] Qlik, "What is a Data Lakehouse?," 2025. [Online]. Available: https://www.qlik.com/us/data-lake/data-lakehouse

[10] S. Nuthalapati, "The Power of the Data Lakehouse: Shaping The Future of Analytics and Machine Learning," Forbes Technology Council, 2024. [Online]. Available: https://www.forbes.com/councils/forbestechcouncil/2024/09/06/the-power-of-the-data-lakehouse-shaping-the-future-of-analytics-and-machine-learning/

[11] A. A. Ahmed and F. Zulkernine, "Data Lakehouse: A survey and experimental study," Inf. Syst., vol. 127, p. 102460, 2025. doi: 10.1016/j.is.2024.102460.

[12] A. Dennis, Systems Analysis and Design, 5th ed. Wiley Publishing, 2015.

[13] J. S. Valacich and J. F. George, Modern Systems Analysis and Design, 8th ed. Pearson Education, 2017.

[14] M. Armbrust, A. Ghodsi, R. Xin, M. Zaharia, and U. Berkeley, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," in Proc. CIDR, 2021.

[15] Amazon Web Services, “Best practices for building a data lake for games,” AWS White Paper, 2025. [Online]. Available: https://docs.aws.amazon.com/pdfs/whitepapers/latest/best-practices-building-data-lake-for-games/best-practices-building-data-lake-for-games.pdf

[16] Dremio, “Best Practices for Efficient and Productive Analytics,” White Paper, 2020. [Online]. Available: https://hello.dremio.com/rs/321-ODX-117/images/Dremio-Semantic-Layer.pdf

[17] A. Ali, “Building the Lakehouse - Implementing a Data Lake Strategy with Azure Synapse,” Microsoft Learn, 2022. [Online]. Available: https://techcommunity.microsoft.com/t5/azure-synapse-analytics-blog/building-the-lakehouse-implementing-a-data-lake-strategy-with/ba-p/3612291

[18] Team Atlan, “6 Apache Iceberg Benefits for Large-Scale Data Lakehouses,” Atlan, 2025. [Online]. Available: https://atlan.com/know/iceberg/apache-iceberg-benefits/

[19] A. V. Chaudhari and P. A. Charate, “Optimizing Data Lakehouse Architectures for Scalable Real-Time Analytics,” Int. J. Sci. Res. Sci. Eng. Technol., vol. 12, no. 2, 2025. doi: 10.32628/IJSRSET25122198.

[20] Databricks, “Managed MLflow,” 2025. [Online]. Available: https://www.databricks.com/product/managed-mlflow