From Data Warehouse to Lakehouse: A Comparative Review

Conference Paper · December 2022

DOI: 10.1109/BigData55660.2022.10020719

2 authors:

Ahmed Harby
Queen's University
5 PUBLICATIONS 12 CITATIONS
SEE PROFILE

Farhana H. Zulkernine
Queen's University
101 PUBLICATIONS 1,438 CITATIONS
SEE PROFILE

From Data Warehouse to Lakehouse: A Comparative Review

Ahmed A.Harby
School of Computing
Queen's University
Kingston, ON, Canada
Ahmed.harby@queensu.ca

Farhana Zulkernine
School of Computing
Queen's University
Kingston, ON, Canada
Farhana.zulkernine@queensu.ca

***Abstract*— Digital information systems currently generate a vast amount of data every minute which emphasizes the continuing need to advance big data management systems with efficient data ingestion and knowledge extraction capabilities. To address the 'big data' problems due to high volume, velocity, variety, and veracity, data management systems evolved from structured databases to big data storage systems, graph databases, data warehouses, and data lakes but each solution has its strengths and shortcomings. The need to produce actionable knowledge fast from unstructured data ingested from distributed sources requires a marriage of data warehouses and data lakes to create a data Lakehouse (LH). The objective is to use the strengths of the data warehouse in producing insights fast from processed merged data, and of the data lake in ingesting and storing high-speed unstructured data with post-storage transformation and analytics capabilities. In this paper, we present a comparative review of the existing data warehouse and data lake technology to highlight their strengths and weaknesses and propose the desired and necessary features of the LH architecture, which has recently gained a lot of attention in the big data management research community.**

Keywords*— Big data, Data Warehouse, Data Lake, Data Lakehouse

I. Introduction

The pace of technological progress has revolutionized the way we generate, collect and process big data [1]. Big Data originates from multiple sources at different times and have different formats [2]. Traditional databases are not able to handle such complex, mixed modality unstructured data arriving at varying speeds, which require different transform methods [3]. In addition, the established Extract, Transform, and Load (ETL) cannot support high-speed data ingestion and handle variations in the structure and modality of the incoming data to store and manage data efficiently[5].

In the 1990s, Data Warehouses (DW) [4] gained popularity by executing overnight ETL processes, extracting, transforming, and loading information from multiple data sources, and integrating the same into a data cube with additional analytical tools to enable Business Intelligence (BI) as shown in Fig.1. DWs enabled fast execution of complex frequent analytical queries to generate insights for business decisions. The big data era offered new challenges for the DWs. Complex mixed modality unstructured data could no longer be extracted, processed, integrated, and stored using the ETL pipeline in a structured DW in a timely manner. Now high-speed and high-volume data from social media containing images, text, and audio, or data from multiple connected devices or Internet of Things (IoT) must be ingested, processed, and linked in near real-time for just-in-time decision support to generate value. The advancements in storage capacity and high-speed processors are not able to transform the traditional DWs, which house structured filtered data, to efficiently handle big data.

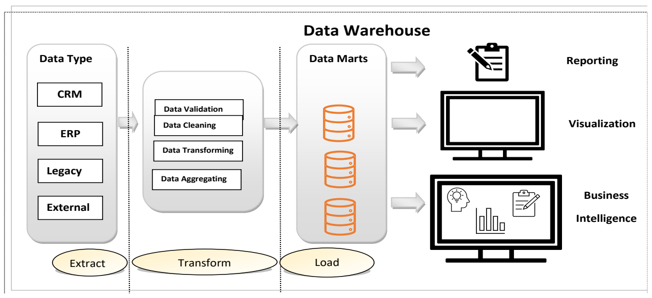

Fig. 1 Data Warehouse (DW) architecture: Customer data from two data silos, Customer Relationship Management systems (CRMs) and Enterprise Resource Planning systems (ERPs), are extracted, transformed, linked and loaded in diverse formats to be stored in Data Mart DW in an integrated manner for visualization, reporting, and business intelligence.

During the last two decades, Data Lakes (DL) evolved to support the storage requirements of high-speed hybrid unstructured data [6]. DLs implement efficient data ingestion methods to extract data from multiple sources and enable fast storage in big data storage systems as shown in Fig. 2. Subsequent analytics performed on the data allow knowledge extraction and management using DWs and other structured storage to produce BI. The DL technology quickly led to a new

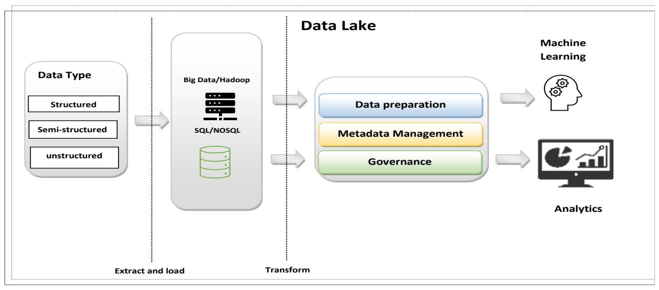

Fig. 2. Data Lake architecture: structured, unstructured, and semi-structured data are ingested and loaded using a metadata management system that enables data discovery, governance, data preparation, and cleansing.

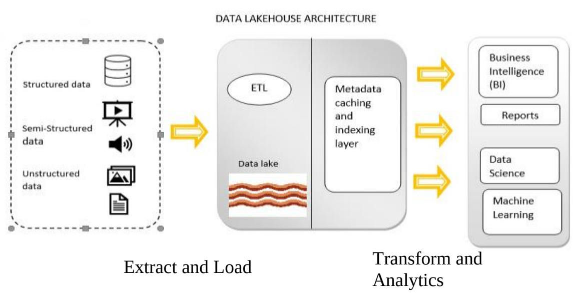

Fig. 3. Data Lakehouse Architecture.

problem of turning into data swamps due to the delay in processing raw data [7]. The delay resulted from multiple factors including changing data content and layout, inconsistency and error in the data, and the need to repeatedly extract metadata to update the data transformation processes to work properly. The resulting stale data in data swamps cause resource wastage and loss of value. DLs also lack an objective method that can be used to assess or compare the performance, especially of their metadata management systems.

To combine the desired features of the DL and DW and address their weaknesses with a view to expediting efficient knowledge extraction, researchers are exploring a new solution called Data Lakehouse (LH) [8]. The LH architecture should support big data storage, analytics, and decision query processes while salvaging the existing technology where possible as shown in Fig. 3. In this paper, we explore the state-of-the-art DWs, DLs, and LH technology, compare their strengths and shortcomings, analyze design choices and architecture, metadata storage and processing features to support evolving data structures that have analytic and decision support capabilities for anticipated needs for the future. The study will allow the best use and enhancements of the existing technology to design and validate a robust data LH.

The rest of the paper is organized as follows. Section 2 presents related work on DW, DL, and LH from the literature. Section 3 presents a list of popular existing technology and a comparison of the features and performance of DW and DL to identify the necessary features of a data LH system. Section 4 illustrates a high-level architecture of our proposed data LH. Finally, Section 5 summarizes the information and lists future research directions.

II. Overview and Related Work

A. *Data Warehouse (DW)*

A DW as shown in Fig. 1 [9], generally serves as an organization's central repository which extracts, transforms, and integrates data from a variety of relational data sources into a linked data cube. Inmon [10] presented a study on DWs including their use in data mining. Inmon stated that a DW is a time-variant, subject-oriented, integrated, and nonvolatile collection of data that aids management's decision-making process. A Data Mart (DM) for Online Analytical Processing (OLAP) was discussed and compared with DWs. DMs have variations of DW architecture and contain smaller subsets of data for faster and more focused queries. Extracting and managing up-to-date DMs from an enterprise-wide information system pose a difficult challenge.

Ghezzi et al. [11] presented a study focusing on efficient DM extraction and deployment using cutting-edge technologies such as ROLAP or MOLAP data servers. However, there were flaws due to the lack of specific strategies for representing and exploiting domain-specific information in DMs to improve knowledge representation and schema evolution. Although these features have improved and DMs have succeeded in facilitating the analysis to meet business requirements [13], without the evolution of DW, the maintenance of DMs proved to be time-consuming, expensive, and error-prone. As a result, later studies by Taktak focused on view maintenance, schema evolution, and versioning to improve DMs and DWs [12].

Yang et al. [14] also presented a study on the modeling and categorization of DWs. Enterprises expect DW tool vendors to satisfy all the following features, which makes it difficult to identify the right model that can satisfy the organizational needs.

- ETL and data integration
- Improving data quality
- Master data management
- Big data management
- Metadata management
- Reference data management

Sureddy et al. [15] conducted research on how to select the right DW based on organizational needs. Proprietary DW systems are expensive and often not compatible with open-source and cloud-based data science and data engineering tools. With the need to support big data processing tools and address the above difficulties, organizations often resort to selecting a different solution paradigm such as a DL.

B. *Data Lake (DL)*

DL was proposed by Dixon [16] as a solution to address the shortcomings of the DW. Dixon also provides information on DL issues and challenges including how data is processed inside DL and how DLs can add value to a business. Later, Inmon [17] recommended raising the level of data organization in a DL into several Data Ponds based on source, purpose, and the type of data as given below. Analytical processing enhances once the data in the DL has been divided into ponds.

- Raw data pond: A DL region containing data without any significant analytical processing.
- Analog data pond: A DL region storing reduced rebuilt analog data to increase workability and manageability.
- Application data pond: A DL region storing data from one or more applications.
- Textual data pond: A DL region for storing text data. Archival data pond: A DL region for storing raw data for archival and retention even after being fully processed and not being generally accessible for processing.

Data conditioning techniques are applied to make the data in the ponds accessible and useful. Assessing data quality and cleaning large volumes of heterogeneous data sources pose a difficult challenge to upholding the sustainability of the DL systems. Farid et al. [18] presented a data discovery, integrity checking, and constraint definition and enforcement tool called

CLAMS. CLAMS combines signals from several sources encompassing many datasets with user input to provide accurate data corrections. CLAMS uses a data ingestion module to load datasets into a Hadoop Distributed File System (HDFS), identifies data sources for cleaning, and executes information extraction to produce semi-structured datasets. The constraint-building module has an interactive user interface, enables the automatic discovery of simple data quality rules, and guides the user in creating more complex rules.

Mathis et al. [19] proposed a centralized and scalable DL storage for organizations, which enables knowledge discovery using additional techniques and 3rd party tools. However, the process to apply these tools must be established as needed to enable data processing and integration. There is an agreed-upon notion of central data storage, but no agreed-upon definition of the DL components or architecture.

A functional architecture of DLs was proposed by Ravat et al. [20] that addressed the categorization of generic and extensible metadata. To extract information for BI, metadata were used to describe each dataset (intra-metadata) and to link datasets (inter-metadata). Using the proposed DL, which combines internal relational databases and medical e-documents with external data from the University hospital center, data analysts and business intelligence specialists can analyze accessible data to improve medical treatments.

Nargesian et al. [7] highlight the challenges and issues of DL management specifically in the following aspects.

- Data ingestion
- Data extraction
- Cleaning & versioning
- Dataset & schema discovery
- Metadata management
- Data integration

With limited bandwidth, ingesting data from external sources requires high levels of parallelism, and no in-depth analysis is performed. DLs contain so much data and there is no comprehensive schema or catalog, making it difficult to discover data, as the "wisdom of the lake" has not yet been fully utilized. Data catalogs are critical for on-demand discovery, integration with DLs, and raw data cleansing. If specific metadata is not provided, DLs can quickly become data swamps. Although metadata offers the required abstraction for understanding and discovering data, there is still potential for improvement in how DLs and their related information are accessed and linked to existing knowledge bases (general and domain-specific). To integrate data on-demand, DLs must manage heterogeneity and enable data extraction and cleansing on-demand.

C. *Data Lakehouse (LH)*

Shiyal [23] presented a comprehensive study to identify the key factors influencing the evolution of the DW and DL into the data LH. An overview of data LH is provided, which is a relatively new concept that combines the following features.

- The flexible big data storage component of DLs.
- The structured clean integrated storage of DW.
- Components of DL to facilitate data integration, discovery, metadata management, and analytics.
- A series of big data analytics, artificial intelligence, and machine learning tools can be applied to the data at various stages of processing.
- Components of DW to enable Online Analytical Processing (OLAP) queries for BI.

Over a long period of time, data landscapes have evolved from DWs to DLs with the recent progression to data LHs. Armbrust et al. [8] state that LHs address several major challenges that DWs are facing today with big data including data lock-in, staleness, reliability, and total cost of ownership. The use of separate DLs and DWs adds complexity and makes it harder to build seamless data connections and processing pipelines. DLs and DWs may support different data types, schemas, and SQL dialects, and cause an increased number of ETL/ELT jobs spanning multiple systems which in turn can increase the probability of errors and bugs. Furthermore, images, sensor data, and documents make up a great deal of unstructured data in many industries. SQL DWs and their APIs cannot manage this data due to their complexity [8]. DWs and DLs are not well suited to handle machine learning and data science applications. In addition, this study discussed some recent systems which use Apache Iceberg and Delta Lake [25] to manage data and execute ACID transactions to ensure data integrity and consistency. Metadata layers improve management capabilities, but they do not guarantee good SQL performance. The performance of DWs is optimized by storing frequently accessed data on faster devices, maintaining statistics, building indexes, and optimizing data formats.

Begoli et al. [24] proposed a data LH architecture for applications in biomedical research and health data analytics.

TABLE 2 EXISTING CATEGORIES OF DW ARCHITECTURE

Architecture	Category	Description
Layer	Single-Layer	A multidimensional model of operational data is created by the intermediate layer of the DW to reduce data redundancy by storing as few variables as possible.
	Two-Layer	In this model, the data should be retrieved from the source, cleaned to remove inconsistencies, and integrated with Extract, transform, and load (ETL) tools that combine disparate schemata, extract, convert, clean, verify, filter, and load source data into DWs (data marts), and analysis (reporting tools, OLAP tools, Data mining tools).
	Three-Layer	The three-tier design is made up of the source layer, the reconciliation layer, and the DW layer (containing both data warehouses and data marts). This layer of reconciliation between the DW and the source provides a standardized reference model for a company and helps differentiate between the challenges of extracting and integrating source data from the warehouse and the benefits of cleaning and integrating data in the warehouse.
Access	Bus	One key difference between an independent data mart and bus architecture is that data marts are logically connected, allowing for a broader view of knowledge across the enterprise
	Hub-and-Spoke	In a hub-and-spoke architecture, data sources are reconciled with data marts, and data marts hold normalized atomic data that feeds a chain of multidimensional data marts that enable scalability, extensibility, and retrieval of massive amounts of data.
Data mart	Dependent	A dependent data mart is a location whose data is derived from a DW. In a dependent data mart, data is accumulated, reorganized, and summarized after it has been transferred from a DW.
	Independent	Independent data mart architecture is composed of distinct data marts that are planned and developed separately and are not integrated with one another. Inconsistencies in data descriptions, as well as divergent dimensions and sizes, are common in data marts, making data analysis challenging.
Storage Layout	Centralized	The hub-and-spoke architecture is similar to this, except for the lack of data marts, as it consists of a single unified DW with integrated data and data marts.
	Distributed	This architecture includes each DW/data mart conceptually or physically through joint keys, global metadata, distributed queries, and other means.

The authors include various features to support the Findability, Accessibility, Interoperability, and Reuse (FAIR) standard, HIPAA regulations, and Institutional Review Board (IRB) protocols which are necessary for the biomedical data domain. Data access can be streamlined by automating manual processes and incorporating pre-processing mechanisms into data ingestion to enhance data access.

D. Comparative Overview of DW, DL, and LH

Due to the flexibility, scalability, analytic capabilities, and BI potential of both DLs and DWs, LH data management is progressing fast to become an industry standard [25]. A Data LH offers an attractive storage option for growing organizations with a cost-conscious mindset. Table 1 summarizes the major differences between the three approaches previously discussed in this section. For this comparison, we selected a collection of the most relevant criteria. The primary industrial metrics are cost and quality, but we also compare atomicity, consistency, isolation, and durability (ACID) to ensure that a database transaction is completed on time. Additionally, we include criteria for how the structure is customized, so developers can make a more informed decision. To provide a summary for users of currently compatible tools and storage systems, we also gather both qualitative and quantitative data on usability.

III. Taxonomy of DW, DL, & LH

A. Categorization based on architectures

A key set of functionalities are demonstrated by the DW, DL, and LH to support data extraction, transformation, loading, and management (storage, indexing, querying, archival). The need for analytic functionality is increasing to generate insights for business decisions in near real-time. Fig. 1, 2, and 3 show how and at what stage, each of the systems implements extract, transform, and load, and in some cases, execute analytics. Analytics capabilities are not a part of the storage system for DW and DL but are suggested to be incorporated in the LH. Each of these functionalities can have regular and advanced capabilities and that is where the systems mainly differ. Fig. 4 shows the different categories of existing DW architectures. A description of each category is provided in Table 2 based on the study done by Jameel[31]. DL architectures, on the other hand, are categorized either as a pond [17], or based on other functional features to offer a sufficient flow for structured, semi-structured, and unstructured data as illustrated in Fig. 5. Using both models, businesses and industries can handle big data management challenges, while data warehouses enable efficient data loading and access.

TABLE 1 COMPARISON BETWEEN DW, DL, AND DATA LH

Criteria	Data Warehouse	Data Lake	Data Lakehouse
Importance	Data analytics and business intelligence	Machine learning (ML) and artificial intelligence	Both data analytics and machine learning
Data type	Structured	Semi-Structured and Unstructured	Structured, Semi-structured, and Unstructured
Cost	Expensive and time-consuming	Inexpensive, quick, and adaptable.	Inexpensive, quick, and adaptable.
Structure	Unconfigurable	Customizable	Customizable
Schema	Defined before the data is stored	Developed after the data is saved.	Developed after the data is saved.
Usability	Users can easily access and report data	Analyzing vast amounts of raw data without tools that classify and catalog the data can be arduous.	Combining the structure and simplicity of a DW with the wider use cases of a DL.
ACID Conformity	Guarantees the greatest levels of integrity, data is recorded in an ACID-compliant way.	Updates and deletes are difficult procedures that need non-ACID compliance.	ACID-compliant to ensure consistency when several parties read or write data at the same time
Quality	High	Low (data swamps)	High

B. Functionality

In this part, we go through the functional categorization of each system in further detail. All of the functional features of DW are connected to data loading and data access as shown in Fig. 6. Data loading is separated into four major categories: data storage, data capture, data normalization, and data sourcing, whilst data access is used for many sorts of workloads such as analytics, reporting, and BI. Hierarchical data distribution is essential for DL, which means the functional features, as shown in Fig. 7, differ quite a bit, but both systems should enable improved accessibility and scalability, along with high availability and comprehensive data management. DW storage systems include DynamoDB, Hadoop, Amazon Redshift, IBM DB2, and Microsoft SQL Server databases, while DL storage systems include Hadoop (HDFS, MapReduce), and Amazon S3. Metadata management (Fig. 7) provides some important features of DL as listed below [22].

- Semantic Annotation
- Data heterogeneity

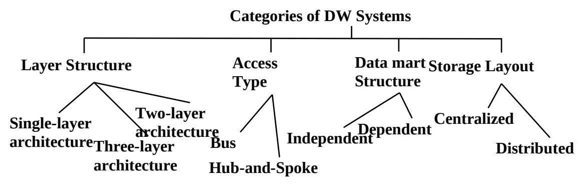

Fig. 4. Categorization of DW system architectures.

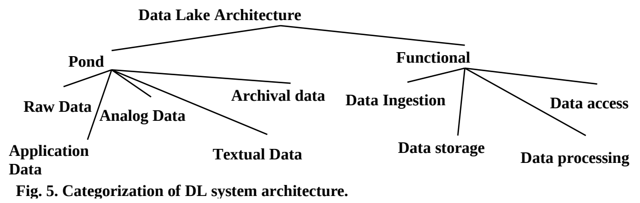

Fig. 5. Categorization of DL system architecture.

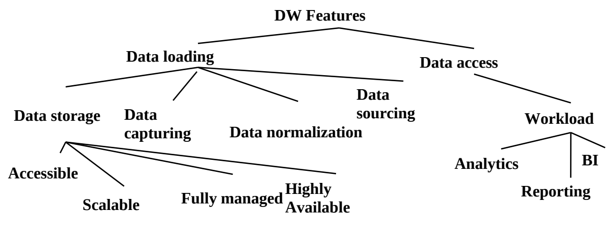

Fig. 6. Categorization of DW functional features.

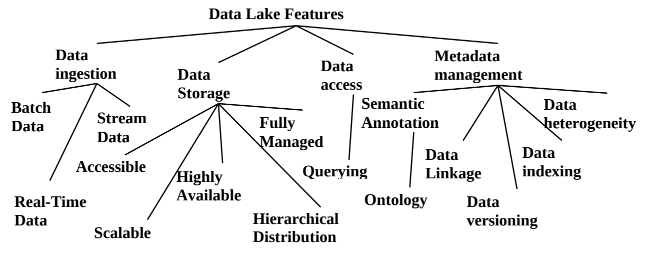

Fig. 7. Categorization of DL functional features.

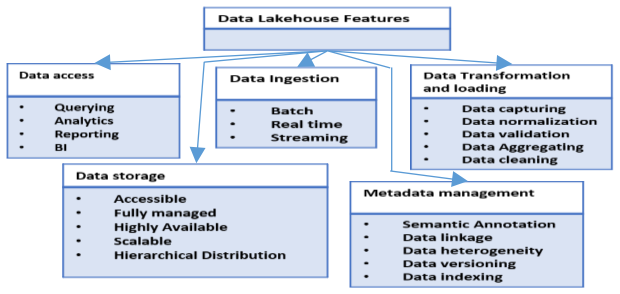

Fig. 8. Desired data LH Features for the proposed LH architecture.

- Data linkage
- Data versioning
- Data indexing

Next, we will discuss the design of our proposed data LH.

IV. Proposed Data Lakehouse Architecture

In this part, we present details about the current data LH architectures, and major elements included in an LH, and explain the necessary features and recommended direction for our conceptual model.

A. *Architecture and Functionality*

Only a few articles focus on the LH design. Armbrust [8] proposed an LH architecture which was adopted by Begoli et al. [24]. In Fig. 3, we combine the data LH architectures from both research studies [8][24], which demonstrates data storage, data ingestion, metadata layer, and querying. The architecture was evaluated using the Parquet file system. Delta Lake is used for the metadata layer, and it caches files from the cloud object stored on faster storage devices such as SSDs and RAM in the processing nodes. Spark is used for query optimization. Using the Parquet file system, the proposed data LH achieved a 50% performance improvement and a significant storage reduction.

B. *Required Features and Proposed Model*

This section describes the features required to develop an efficient data LH. Fig. 8 shows the necessary properties from both DW and DL, which are described below.

- Data access enables analytics, reporting, querying, and business insights.
- Data ingestion should support all types of data including batch, real-time, and streaming data.
- Transformation and loading of data are essential after extracting metadata. Data transformation is required to enable data normalization, cleaning, aggregation, loading with validation, and data capturing.
- Data storage systems should have high accessibility, scalability, availability, hierarchical distribution, and comprehensive data administration.

We discuss relevant tools to support these functionalities with a view to preserving coherence.

a) Data storage: A robust storage management strategy guarantees that the data is always available to users and applications. An effective storage management system should support the following properties in order to meet the specific performance and capacity demands of each business [21].

- Elasticity
- Cost-effective
- Availability
- Durability
- Data integrity
- Fault detection & recovery
- Access control

The HDFS open-source framework is the most widely used framework for DLs, but there are other systems that are more flexible, like Amazon S3. Table 3 shows the most commonly used data storage systems that are compatible with DWs, DLs, and LH. Amazon S3 has the lowest cost among all, and its performance is almost double that of HDFS [29]. It also provides the support of Databricks for data integrity. However, with S3, all reads must go across the network prohibiting performance optimization, which is a serious drawback.

TABLE 3 DATA STORAGE SYSTEMS COMPARISON

Criteria	**Amazon S3**	**Apache Hive**	**Google Cloud Store**	**HDFS**
Elasticity	Yes	No	Yes	No
Cost-effective	Low	high	high	High
Availability	High	High	High	High
Durability	High	High	High	High
Data integrity	Yes with Databricks	Yes	Yes	Yes
Fault detection and & recovery	Yes	Yes	Yes	Yes
Access Control	Full	Full	Full	Not fully managed
Compatibility	DW, DL, LH	DW	DW, DL, LH	DL, DW, LH
Performance	High for low cost	High for high cost	High for high cost	High for high cost

b) Data ingestion: Due to the continuing growth in the volume of data, fast data ingestion and knowledge extraction in a cost-effective manner is a challenging problem due to a) 4V properties of big data i.e., volume, velocity, variety, and veracity, b) variation in data structure and content from numerous data sources, which requires continuous adaptation of data ingestion algorithms and automatic detection of data content and structure, c)need for real-time BI, which requires low latency data ingestion and fast learning important information including semantic information processing for real-time BI, and d) evolving *syntax and semantics of data.*

Table 4 compares the commonly used data import technologies based on the type of data ingested. In essence, Apache NIFI and Apache Flink [28] handle both batch and real-time operations.

TABLE 4 DATA INGESTION TOOLS

Tool	Batch	Real-time	Near real-time
Apache Kafka		✓	
Apache NIFI	✓	✓	✓
Amazon Kinesis		✓	✓
Apache Flink	✓	✓	
Apache Spark	✓		✓
Apache Sqoop	✓		
Apache Storm		✓	
Apache Gobblin	✓	✓	

c) Metadata Management: Metadata in a DL is written in a machine-readable format that follows the Common Data Model (CDM) standard. Metadata generally contains data catalogs containing the type, size, and location of the data but with variations in data, incorporating sufficient details about the data for automated processing and optimizing the use of metadata poses difficult challenges. We propose to extend the

TABLE 5 COMPARES SEVERAL METADATA MANAGEMENT APPROACHES USED WITH EXISTING DL IN THE LITERATURE (SA- SEMANTIC ANNOTATION, DL- RESEARCH TO ASSIST US IN DETERMINING WHICH IS THE MOST BENEFICIAL TECHNIQUE ALIGNED WITH OUR SUGGESTED MODEL.

	Semantic Annotation	Data Linking	Data Heterogeneity	Data Versioning	Data Indexing
CLAMS [18]	✓				
CODAL [22]	✓				
Medal [22]	✓		✓	✓	✓
Constance[22]	✓				✓
DLDS[22]	✓		✓		
CEBA [22]	✓			✓	
KAYAK[22]	✓				✓
DM [27]	✓			✓	
CoreDB[22]				✓	✓
CoreKG [22]	✓	✓	✓	✓	✓
Egeria [26]	✓	✓	✓	✓	✓

metadata to include a knowledge graph to include information about linked data and enhance analytical capabilities in the LH.

Table 5 compares several metadata management systems that are generally used in DL research [22] and can be used in data LH as well. Furthermore, we incorporated comparison criteria such as semantic annotation, data linkage, data heterogeneity, versioning, and indexing that are used by most of the existing metadata management tools.

d) Data Transformation and Loading: Ingested raw data often needs transformation i.e., cleansing, preprocessing to remove noise, conversion to desirable format, and summarization before storage and loading into memory for further processing or analytics. Unprocessed data dumps lead to data swamps, whereas applying lengthy preprocessing can create a bottleneck in the data ingestion pipeline. Therefore, an optimal and efficient data ingestion pipeline is critical to sort the data at ingestion and distribute it to proper storage. We will develop a knowledge extraction method to incorporate into the data ingestion pipeline to learn the key features in the data and store the significant patterns as data profiles to reduce storage footprint and avoid data swamps. The stored data profiles will also facilitate the exploration of linked data for augmenting the metadata representation. We also plan to develop a tool to detect changes in the incoming data and apply adaptations to the data ingestion, metadata management, and knowledge extraction methods.

e) Data Access: To facilitate efficient data access in a data LH, OLAP queries should be able to execute advanced knowledge extraction methods involving the exploration of linked data. For faster BI, LHs should incorporate offline knowledge exploration and linking. We will explore methods to automatically discover linked data based on user queries on the stored data profiles, generate usable knowledge to enrich the LH and discover anomalies to report. Advanced visualization and communication tools will be incorporated to facilitate the presentation of and access to knowledge.

C. *Proposed Model Architecture*

Fig. 9 presents our proposed data LH architecture. Based on an extensive review of the existing tools and architecture of DW, DL, and data LH, we include a concise set of tools and subsystems that can lead to a preliminary LH design; however, the design will evolve as these tools are implemented based on the use case.

Amazon S3 is a cost-effective data storage option. We will incorporate an adaptive metadata management portfolio that has many metadata management features to handle various forms of data such as text, video, and image. We like to use an open-source ontology-based tool such as Egeria [26] with a wide range of capabilities. We intend to use Apache Flink and Apache

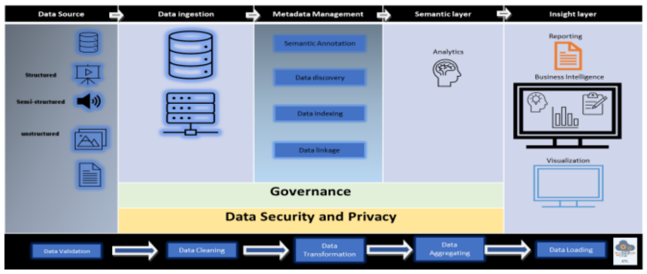

Fig. 9. Proposed data LH Architecture.

NIFI as our ingestion tools based on the study of Sharma [28]. Data ingestion tools must be scalable, have security features to filter malicious data, and be able to handle 4V challenges. Apache Flink can process 4 million records per second with batch and real-time data using a single node cluster. It has been successfully integrated with a variety of tools such as Apache Kafka, Elastic Search, HDFS, RabbitMQ, Amazon Kinesis Streams, Twitter, Apache NIFI, Apache Cassandra, and Apache Flume.

Some studies introduced graph knowledge extraction for DL which demonstrated great success [22]. We propose to employ a graph knowledge extraction and representation model for efficient data analytics and querying [30]. Finally, we will use Snowflake's processing and storage capacity for cost-effective storage. It has a large user base which will allow the LH to be used with other 3rd party tools and vendors for a wide number of use cases. Snowflake's data-sharing capabilities can be leveraged to build a reference to a patient database, thereby, reducing the cost of building custom HIPAA-compatible infrastructure for medical data domains.

V. CONCLUSION AND FUTURE WORK

The development of DLs grew out of the need to harness large amounts of structured and unstructured data for machine learning, while DWs were invented to allow users to run powerful analytical queries over multiple data sources. However, both these technologies are facing challenges with high-speed data. New data LH technology has emerged to address today's information needs. Data LH architecture is still in its early stages, and early adopters will need time to discuss the best practices. In this research, we present a comparative study of the state-of-the-art (SOTA) DW, DL, and data LH systems to explore their strengths and weaknesses, and thereby, propose a new LH architecture. We present a literature review of the existing DW, DL, and LH systems including a comparative study to identify the desired features of a future data LH and list the existing tools that are commonly used in the SOTA platforms to help us select the tools to incorporate in our proposed LH platform.

As future work, we propose to explore and advance the existing LH technology by developing a) an advanced and intelligent data ingestion pipeline with knowledge extraction and linking capabilities, b) a data profiling method to capture the significant and anomalous data to reduce storage footprint while enhancing data linking capabilities and c) a knowledge graph to augment the metadata management in LH. These advancements will address the problems of data swamps and enable faster execution of advanced OLAP queries.

Acknowledgment

The research is supported by IBM Center for Advanced Studies, Toronto, Canada.

References

[1] Mayer-Schönberger, V., & Cukier, K. (2013). Big data: A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt.Conference Name:ACM Woodstock conference

[2] Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. In 2013 international conference on collaboration technologies and systems (CTS) (pp. 42-47). IEEE.

[3] Hariri, R. H., Fredericks, E. M., & Bowers, K. M. (2019). Uncertainty in big data analytics: survey, opportunities, and challenges. Journal of Big Data, 6(1), 1-16.

[4] Inmon, W. H. (1996). The data warehouse and data mining. Communications of the ACM, 39(11), 49-51.

[5] Gardner, S. R. (1998). Building the data warehouse. Communications of the ACM, 41(9), 52-60.

[6] Miloslavskaya, N., & Tolstoy, A. (2016). Big data, fast data and data lake concepts. Procedia Computer Science, 88, 300-305.

[7] Nargesian, F., Zhu, E., Miller, R. J., Pu, K. Q., & Arocena, P. C. (2019). Data lake management: challenges and opportunities. Proceedings of the VLDB Endowment, 12(12), 1986-1989.

[8] Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021, January). LH: a new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of CIDR.

[9] Vaisman, A., & Zimányi, E. (2014). Data warehouse systems. Data-Centric Systems and Applications.Conference Short Name:WOODSTOCK'18

[10] Inmon, W. H. (1995). What is a data warehouse. Prism Tech Topic, 1(1), 1-5.

[11] Ghezzi, C. (Ed.). (2001). Designing data marts for data warehouses. ACM Transactions on Software Engineering and Methodology (TOSEM), 10(4), 452-483.

[12] Taktak, S., & Feki, J. (2012, October). Toward propagating the evolution of data warehouse on data marts. In International Conference on Model and Data Engineering (pp. 178-185). Springer, Berlin, Heidelberg.

[13] Sen, A., & Sinha, A. P. (2005). A comparison of data warehousing methodologies. Communications of the ACM, 48(3), 79-84.

[14] Yang, Q., Ge, M., & Helfert, M. (2019). Analysis of Data Warehouse Architectures: Modeling and Classification. In ICEIS (2) (pp. 604-611).

[15] Sureddy, M. R., & Yallamula, P. (2020). Approach to help choose right data warehousing tool for an enterprise. International Journal of Advance Research, Ideas and Innovations in Technology, 6(4).

[16] Dixon J (2010) Pentaho, hadoop, and data lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/

[17] Inmon, B. (2016). Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump. Technics publications.

[18] Farid, M., Roatis, A., Ilyas, I. F., Hoffmann, H. F., & Chu, X. (2016, June). CLAMS: bringing quality to data lakes. In Proceedings of the 2016 International Conference on Management of Data (pp. 2089-2092).

[19] Mathis, C. (2017). Data lakes. Datenbank-Spektrum, 17(3), 289-293.

[20] Ravat, F., & Zhao, Y. (2019, September). Metadata management for data lakes. In European Conference on Advances in Databases and Information Systems (pp. 37-44). Springer, Cham.

[21] Daher, Z., & Hajjdiab, H. (2018). Cloud storage comparative analysis amazon simple storage vs. Microsoft azure blob storage. International Journal of Machine Learning and Computing, 8(1), 85-9.

[22] Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. Journal of Intelligent Information Systems, 56(1), 97-120.

[23] Shiyal, B. (2021). Modern Data Warehouses and Data LHs. In Beginning Azure Synapse Analytics (pp. 21-48). Apress, Berkeley, CA.

[24] Begoli, E., Goethert, I., & Knight, K. (2021, December). A LH Architecture for the Management and Analysis of Heterogeneous Data for Biomedical Research and Mega-biobanks. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 4643-4651). IEEE.

[25] L'Esteve, R. C. (2021). Delta Lake. In The Definitive Guide to Azure Data Engineering (pp. 321-346). Apress, Berkeley, CA.

[26] "Egeria Metadata Management Tool." Egeria, egeria-project.org. Accessed 5 Aug. 2020.

[27] Machado, I. A., Costa, C., & Santos, M. Y. (2022). Data mesh: concepts and principles of a paradigm shift in data architectures. Procedia Computer Science, 196, 263-271.

[28] Sharma, G., Tripathi, V., & Srivastava, A. (2021). Recent Trends in Big Data Ingestion Tools: A Study. In Research in Intelligent and Computing in Engineering (pp. 873-881). Springer, Singapore.

[29] "Databricks Benifits." Databricks, databricks.com/blog/2017/05/31/top-5-reasons-for-choosing-s3-over-hdfs.html. Accessed 13 Aug. 2022.

[30] Abughofa, T., Harby, A. A., Isah, H., & Zulkernine, F. (2021, August). Incremental Community Detection in Distributed Dynamic Graph. In 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService) (pp. 50-59). IEEE.

[31] Jameel, K., Adil, A., & Bahjat, M. (2022). Analyses the Performance of Data Warehouse Architecture Types. Journal of Soft Computing and Data Mining, 3(1), 45-57.