computers

MDPI

Review

Data Lakes: A Survey of Concepts and Architectures

Sarah Azzabi, Zakiya Alfughi and Abdelkader Ouda *

Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada; sazzabi@uwo.ca (S.A.); zalfughi@uwo.ca (Z.A.)
* Correspondence: aouda@uwo.ca

Abstract: This paper presents a comprehensive literature review on the evolution of data-lake technology, with a particular focus on data-lake architectures. By systematically examining the existing body of research, we identify and classify the major types of data-lake architectures that have been proposed and implemented over time. The review highlights key trends in the development of data-lake architectures, identifies the primary challenges faced in their implementation, and discusses future directions for research and practice in this rapidly evolving field. We have developed diagrammatic representations to highlight the evolution of various architectures. These diagrams use consistent notations across all architectures to further enhance the comparative analysis of the different architectural components. We also explore the differences between data warehouses and data lakes. Our findings provide valuable insights for researchers and practitioners seeking to understand the current state of data-lake technology and its potential future trajectory.

Keywords: big data; hadoop; data lake; data warehouse; data lakehouse; data management; schema; internet of things

Citation: Azzabi, S.; Alfughi, Z.; Ouda, A. Data Lakes: A Survey of Concepts and Architectures. *Computers* **2024**, *13*, 183. https://doi.org/10.3390/computers13070183

Academic Editor: Paolo Bellavista

Received: 27 June 2024
Revised: 17 July 2024
Accepted: 18 July 2024
Published: 22 July 2024

1. Introduction

Timely access to high-quality data is the life-giving force of modern decision-making systems and the foundation for accountable knowledge. In this essence, consistent focus on data and adapting data-driven approaches from academics and professionals exists because the knowledge extracted from data analysis leads to innovations that redefine enterprises and national economies [1].

In the era of the 4th Industrial Revolution, approximately 328.77 million terabytes of data are generated, captured, copied, or consumed globally every day [2]. Organizations of all sizes across various sectors are developing their technological capabilities to extract knowledge from large and complex datasets, commonly known as "big data". This term refers to large datasets that include structured, semi-structured, and unstructured data, which traditional data management tools struggle with due to their vast size, heterogeneity, and complexity. In addition, big data comes in multiple formats, including text, sound, video, and images, with unstructured data growing faster than structured data, accounting for 90% of all data [3]. Driven by the changing data landscape, new processing capabilities are required to gain insights from big data, leading to better decision-making.

The challenges associated with the data life cycle are primarily related to data processing and management issues that arise from the volume, velocity, variety, and veracity of big data [3]. Data processing challenges involve techniques that are used for acquiring, integrating, transforming, and analyzing big data. On the other hand, data management challenges deal with ensuring data security, privacy, governance, and operational cost issues. All of these challenges can lead to information overload and low productivity due to difficulties in obtaining actionable knowledge from data.

Data management is the systematic retrieval and administration of information assets within an organization to ensure that the data are accessible and shareable across different formats and systems. Data management systems utilize a variety of technical frameworks,

methodologies, and tools that are applicable to different scenarios, use cases, and lines of business demands. However, the very nature of modern big data poses significant challenges to traditional data management systems [4].

The main limitations of traditional data management systems are represented by their fixed schemas and rigid data models, which in many situations constrain the organizations' capacity to handle diverse data types. It is challenging to include new data sources or to evolve with the business needs [5]. Also, data silos, created when departments or applications store data independently, obstruct data sharing and integration, resulting in incomplete and inconsistent datasets that undermine the trustworthiness of analysis results. Finally, scaling these systems to manage extremely large volumes of data is often excessively expensive in terms of the initial investment in hardware and software platforms, as well as the ongoing maintenance and upgrade costs, which can represent a significant cost factor for organizations dealing with big data [5].

To overcome these limitations, the concept of the "data lake" has been introduced. Data lakes consolidate data from various disparate sources into a single, unified management system. This approach eliminates the fragmentation and inconsistency of data silos and allows organizations to apply uniform data governance. Data lakes provide flexible data access, automated and efficient data preparation workflows, and reliable data pipelines [6]. These capabilities support a wide range of analytical applications, enabling organizations to conduct comprehensive analytics on all available data and facilitate end-to-end machine-learning workflows.

This survey aims to shed light on the existing works that explore data-lake architectures. The main research contributions made in this study are as follows:

- We provide a thorough explanation of the data-lake definition and concept.
- We provide a detailed overview of the differences between both data warehouses and data lakes.
- We review and categorize existing data-lake solutions based on their architectures.
- We construct a chronological timeline graph to illustrate the evolution of data-lake architectures.
- We explore the significance of data lakes in modern data architecture, the challenges they pose, and future data-lake development.

The remainder of this paper is organized as follows: Section 2 presents the applied review methodology. Section 3 introduces the data-lake definition and its main characteristics. Recent existing studies related to data lakes are reviewed in Section 4. The findings of this study are presented in Section 5. Section 6 provides a chronological timeline graph to illustrate the evolution of data-lake architectures. Finally, Section 7 concludes the paper by addressing the importance of data lakes in modern data architecture, along with their adaptation challenges and future development trends.

2. Review Methodology

This study employed a systematic literature review approach to examine and synthesize the existing research on data-lake architectures. The review process followed these key steps:

1. Research question formulation: We defined the primary research question as "What are the major types of data-lake architectures that have been proposed and implemented, and how have they evolved over time?"
2. Literature search: A comprehensive search was conducted using academic databases including IEEE Xplore, ACM Digital Library, ScienceDirect, and Google Scholar. These databases were chosen for their extensive coverage of relevant literature:
    - IEEE Xplore: Provides a vast collection of technical literature in electrical engineering, computer science, and electronics, essential for research on data-lake architectures.

- ACM Digital Library: Contains a comprehensive collection of full-text articles and bibliographic records in computing and information technology.
- ScienceDirect: Offers access to a wide range of scientific and technical research, including key journals and conference proceedings.
- Google Scholar: Ensures broad coverage across disciplines and indexes a variety of academic publications.

Search terms included "data lake architecture", "data lake design", "data lake implementation", and related keywords. The search covered publications from 2008 to 2024 to capture the full evolution of data-lake concepts. The year 2008 was chosen as the starting point as it marks the early development of data-lake technology, and the period up to 2024 includes the latest advancements and implementations in the field.

3. Study selection: Inclusion criteria were applied to select relevant papers that specifically discussed data-lake architectural models, implementations, or evaluations. Exclusion criteria filtered out papers that only mentioned data lakes tangentially or did not provide substantive architectural details.
4. Data extraction: Key information was extracted from each selected paper, including the proposed architecture type, key components, advantages, limitations, and use cases. A standardized data extraction form was used to ensure consistency.
5. Architectural diagram standardization: To facilitate easy evaluation and comparison, the extracted architecture information was sketched into new diagrams using consistent notations across all architectures. This standardization process ensured that all architectural representations followed a uniform format, making it easier to identify similarities, differences, and trends across various data-lake designs.
6. Quality assessment: The selected papers were evaluated for quality based on criteria such as clarity of architectural description, empirical evidence provided, and relevance to practical implementation.
7. Data synthesis: The extracted information was synthesized to identify major categories of data-lake architectures, their defining characteristics, and trends in their development over time. A chronological analysis was conducted to map the evolution of architectural approaches.
8. Critical analysis: The strengths, weaknesses, and applicability of different architectural models were critically analyzed. Comparisons were made between different approaches to highlight their relative merits and limitations.
9. Findings compilation: The key findings from the analysis were compiled, including a classification of major data-lake architecture types, a timeline of architectural evolution, and insights into the drivers of architectural changes over time.

We followed this approach in this survey study to present a comprehensive and structured review of the literature on data-lake architectures, enabling the identification of key trends, challenges, and future directions in this rapidly evolving field.

3. Data-Lake Definition and Characteristics

The term data lake was first introduced by Pentaho CTO James Dixon [7], to address the limitations of data marts, which are business-specific subsets of data warehouses, allowing only a subset of questions to be answered. In the research literature, data lakes are also referred to as data reservoirs [8] and data hubs. However, data lake is the most commonly used term in the literature. In the early stages of development, data lakes were perceived as the Hadoop technology equivalent. From this perspective, the data-lake concept refers to the practice of using open or low-cost technologies, typically Hadoop, to store, process and explore raw data in the enterprise [9]. In recent years, data lakes are generally seen as a central data repository where data of all types are stored in a loosely defined schema for future use. This definition is derived from two distinct characteristics of data lakes: data variety and schema-on-read approach "late binding", which states that schema and data requirements are not defined until the time of data querying [10].

The most elaborate definition of data lake, as quoted in reference [11], is "a scalable storage and analysis system for data of any type, retained in their native format, and employed primarily by data specialists, (statisticians, data scientists, or analysts) for knowledge discovery. The data-lake properties include: (1) a metadata catalogue that imposes data quality; (2) data governance policies and tools; (3) availability to different kinds of users; (4) integration of any kind of data; (5) a logical and physical organization; (6) scalability in terms of storage and processing".

The characteristics of a data lake have been widely explored in academic literature. First, data lakes are characterized by storing massive amounts of data, both structured and unstructured, from diverse and varying sources. Second, data lakes aim to be agile and scalable, i.e., data volumes and types stored in the data lake may vary as business demands change. Third, data lakes are envisioned to support a wide range of data processing and analytics technologies, such as batch processing, interactive analytic queries, and machine learning [4].

Another important characteristic of data lakes is that they utilize metadata to store, manage, and analyze the data stored in the lake. Metadata describes the data, such as its source, format, and schema [11]. In addition, data lakes provide data exploration and discovery, where data analysts can navigate and delve into data without requiring explicit schemas or structures to be defined prior to analysis. Finally, data lakes are characterized by the use of low-cost storage, which permits organizations to store petabytes of data without incurring significant expenses [12].

4. Literature Review

According to Dixon, "whilst a data warehouse seems to be a bottle of water cleaned and ready for consumption, then 'Data Lake' is considered as a whole lake of data in a more natural state" [7]. Understanding the distinctions and advantages of both approaches can help businesses make informed decisions on which option best suits their needs. In [13], the authors delve into whether data lakes will supplant data warehouses in the foreseeable future. They begin by introducing the data-lake concept as a new architecture that stores data in its raw format, allowing for flexible processing and analysis. The paper then performs a critical analysis of the advantages and disadvantages currently offered by data warehouses to compare the two concepts—data lake (DL) and data warehouse (DW). The authors explore the scalability, flexibility, and cost-effectiveness of data lakes, highlighting how they accommodate the increasing variety and volume of data in modern business environments. The paper contrasts these features with the structured, schema-on-write framework of traditional data warehouses that are less adaptable to the dynamic nature of today's data needs. Furthermore, the discussion includes the transition from ETL to ELT processes, emphasizing how data lakes support real-time data processing and analytics.

The work presented in [14] discusses the fundamental differences between data warehouses (DW) and data lakes (DL), comparing them across various dimensions to clearly distinguish their functionalities and use cases. The authors focus on key aspects such as data handling, schema design, data volume and growth, data processing, agility, user access and tools, and integration and maintenance costs. This comparison highlights the distinct advantages and disadvantages of each system, providing insights into their suitability for different analytical needs and environments.

Nambiar and Mundra [15] conduct a comparative analysis between data warehouses and data lakes, emphasizing their key distinctions. In particular, the review detailed the definitions, characteristics, and principal differences of data warehouses and data lakes. Additionally, the architecture and design aspects of both storage systems were thoroughly discussed. Their paper also provided an extensive overview of popular tools and services associated with data warehouses and data lakes. Furthermore, the review critically examined the overarching challenges associated with big-data analytics, as well as specific hurdles related to the implementation of data warehouses and data lakes.

Harby and Zulkernine [16] present a comparative review of existing data-warehouse and data-lake technologies to highlight their strengths, weaknesses, and shortcomings. They analyze design choices, architecture, and metadata storage and processing features. Consequently, the authors propose the desired and necessary features of the data-lakehouse architecture, which has recently gained a lot of attention in the big-data management research community.

ElAissi et al. [17] discuss in depth the differences between data lake and data-warehouse architectures. The authors emphasize that a dedicated architecture, known as a data lake, has been developed to extract valuable insights from large volumes of diverse data types, including structured, semi-structured, and unstructured data.

Hagstroem et al. [18] discuss the advantages and strategic implementation of data lakes in business settings, emphasizing their role in handling vast amounts of structured and unstructured data cost-effectively. Furthermore, there is a focus on the stages of data-lake development, particularly how organizations can use data lakes for scalable and low-cost data storage solutions. This includes the creation of raw-data zones and integration with existing data warehouses.

Hassan et al. [19] discuss different big-data storage options, including data lakes, big-data warehouses, and lakehouses, and compare their major characteristics. The paper then explores the transformation among these approaches to introduce the data lakehouse as a new concept that combines the flexibility of data lakes with the structured query capabilities of data warehouses. Additionally, the authors conduct a detailed comparison of data warehouses, big-data warehouses, data lakes, and lakehouses, highlighting their individual strengths and limitations in terms of storage capacity, data processing capabilities, security, and cost-effectiveness.

Over time, various architectures within the data-lake ecosystem have emerged, each designed to address specific needs related to data storage, analysis, and accessibility for end-users. According to [20], the idea of data lakes first emerged with the introduction of Hadoop, which Doug Cutting and Mike Cafarella developed while working to fix the Nutch search engine at Yahoo. Google's MapReduce [21] and Google File System [22] papers served as inspiration for them. Hadoop provided a framework for distributed storage and processing, paving the way for the first versions of data lakes

In his blog post "How to beat the CAP theorem", Nathan Marz [23] first introduced the concept of Lambda architecture. He developed his framework to efficiently manage large volumes of data by utilizing both batch and real-time processing techniques. Lambda architecture describes two processing layers: A batch layer and a speed layer. In some cases, a serving layer is also included, as in [24]. The designed system aims to optimize for fault tolerance, latency, and throughput by processing data through those three layers. On the other hand, maintaining code that needs to produce the same result in two complex distributed systems was the problem with Lambda architecture. In summer 2014, Kreps [25] posted an article addressing the pitfalls associated with Lambda architecture in [23] and introduced the idea of Kappa architecture to handle those drawbacks. To manage various data types, including structured and textual data, and to organize them into an analyzable structure, Bill Inmon [26] introduced the data-pond architecture model. This model comprises a series of components known as ponds, each distinct and logically segregated from the others.

Five major zone models (i.e., variants of the zone architecture) have been discovered: Gorelik [27], IBM [28], Madsen [29], Ravat [30], and Zaloni, whose model exists in multiple versions by different authors [31,32]. Ravat and Zhao [30] review various architectural approaches and then propose a generic, extensible architecture for data lakes. The authors propose a new, more structured architecture for data lakes that consists of four key zones, including raw-data ingestion, data processing, data access, and governance. Each zone is designed to handle specific functions, which enhances the management and utility of data within the lake. In [33], the authors present a three-step solution to ensure that ingested data are findable, accessible, interoperable, and reusable at all times. Firstly, they introduce

a metadata management system that enables users to interact with and manage metadata easily. This system is crucial for ensuring that data ingested into the data lake can be efficiently used and managed over time. Then, they detail algorithms for managing the ingestion process, including storing data and corresponding metadata in the data lake. These processes ensure that metadata is captured accurately and maintained throughout the data lifecycle. Finally, the metadata management system is illustrated through use cases, demonstrating its application in managing real datasets and facilitating easy data discovery and usage within a data-lake environment.

The data lakehouse was first introduced by Armbrust et al. [34] as a recent approach to combine a data lake with a warehouse. Specifically, they proposed a unified data platform architecture, which integrates data warehousing capabilities with open data lake file formats, can achieve performance levels competitive with current data-warehouse systems, and can effectively tackle numerous challenges encountered by data-warehouse users. Thus, the lakehouse seeks to provide features unique to data warehouses (ACID characteristics, SQL queries) and data lakes (data versioning, lineage, indexing, polymorphism, inexpensive storage, semantic enrichment, etc.). Their lakehouse platform design was based on the design at Databricks through the Delta Lake, Delta Engine, and Databricks ML Runtime projects.

In recent years, data lakes have emerged as a platform for big-data management in various domains, such as healthcare and air traffic, to name a few. This new approach enables organizations to explore the value of their data using advanced analytics techniques such as machine learning [35]. The study in [36] examines various established enterprise data-lake solutions, highlighting that numerous organizations have opted to develop enhancements over Hadoop to mitigate its inherent limitations and enhance data security. This includes adaptations seen in platforms such as Amazon Web Services (AWS) data lake and Azure data lake. Additionally, the authors indicate how these solutions are increasingly being adopted across diverse sectors, including banking, business intelligence, manufacturing, and healthcare, underscoring their growing popularity and utility in handling extensive data needs.

A recent study in [37] proposes a complete fish farming data-lake architecture based on a multi-zone structure with three main zones: the Raw Zone (RZ), the Trusted Zone (TZ), and the Access Zone (AZ). This architecture is designed to collect data from different farms in various formats and types, which is then saved directly into the RZ without any transformation, aiming to form a solid historical repository. Following this, the TZ applies lean transformations to prepare the data for further analysis. Finally, the AZ consists of a dedicated layer with the data required for each team. Additionally, the paper highlights various big-data technologies and their uses. The authors also investigate the recently proposed data-lake functional architecture from a technical perspective by explaining each component of the Hadoop ecosystem used.

Ref. [38] discusses the limitations of data-warehouse approach and how the concept of a data lake is introduced as a solution to overcome these limitations. Further, the paper explores the architecture of a data lake and how the healthcare sector can benefit from data lakes. The study also highlights the use of data lakes in developing a prediction system for cardiovascular diseases. An architectural overview of the Azure data lake is presented to offer a concrete example of how a data lake can be effectively implemented within the healthcare sector. This overview emphasizes the enhancement of data management and analytics capabilities, highlighting key components such as the raw, stage, and curated zones.

The authors in [39] implement a data-lake prototype on a hybrid infrastructure combining Dell servers and Amazon AWS cloud services. The prototype illustrates how air traffic data from various FAA sources like SFDPS, TFMData, TBFM, STDDS, and ITWS can be ingested, processed, and refined for analysis. Another work in [40] presents a comprehensive framework for managing diverse data types across the Internet of Things (IoT) and

big-data environments. The proposed architecture is a direct extension and implementation of the conceptual zone-based data-lake architecture outlined in [30].

Some survey articles summarized recent approaches and the architecture of data lakes, such as the one in [11], which reviews several data-lake architectures, introducing a new classification system that categorizes data lakes based on their functionality and maturity. The authors investigate the architectures and technologies used for the implementation of data lakes and propose a new typology of data-lake architectures. This includes zone architectures, which organize data by stages of processing and refinement, and pond architectures, which treat subsets of data-lake contents with specific processing and storage strategies. Another survey in [41] identified two major architectures: the data pond and the data zone. In [42], several architecture models are taken into consideration, including the basic two-layered architecture, multi-layered model, and data-lakehouse model. As more architecture models prove valuable to evaluate, our survey will expand the classification to include more data-lake architectures.

5. Findings

Based on the review that has been conducted in Section 4, we can categorize the findings observed in this study as follows:

5.1. Distinguish Data Lake and Data Warehouse

Many previous research works have highlighted the position of data lakes in the current data architecture. However, it is equally important to distinguish between data lakes and traditional data warehousing since both are storage solutions for large-scale data but with different approaches and functionalities.

Data warehouses are centralized repositories which maintain structured data obtained from diverse sources in a form optimized for querying and analysis. They are based on a schema-on-write strategy, meaning the data are initially structured according to a predefined schema and then stored [19]. This schema-first strategy aims to optimize the queries and ensure data consistency, which makes the data-warehouse architecture more suitable for complex analysis and reporting [17]. On the other hand, data lakes are generally considered as storage repositories that hold both structured and unstructured data in their native format, without pre-processing or modeling, known as schema-on-read [43]. This approach provides flexibility as the schema is applied only during read time, allowing the data to be stored without any predefined structure and enabling the easy incorporation of new data types [44].

Regarding the data processing strategies, data warehouses employ a process named Extract, Transform, Load (ETL), wherein data are extracted from the source systems, transformed to a format suitable for analysis, and then loaded into the warehouse. On the contrary, data lakes employ a process named Extract, Load, and Transform (ELT), wherein data are ingested into the lake in their raw format and then transformed when required for analysis. This strategy provides organizations the freedom to apply different data processing techniques and analytics tools to the data stored in the lake as needed, without the necessity of transforming the data altogether [13,45,46].

Ease of use is another factor that differentiates data warehouses from data lakes. Owing to their unstructured architecture, data lakes offer more flexibility. In contrast, data warehouses are more structured, making them difficult and expensive to manipulate. Furthermore, data lakes compared to data warehouses are more agile and flexible since they are less structured, and developers and data scientists can modify or reconfigure them with relative ease. Table 1 provides a summarized description of the major differences between data warehouses and data lakes.

Table 1. Comparison of data-warehouse and data-lake characteristics.

Attribute	Data Warehouse	Data Lake
Schema	Schema-on-read	Schema-on-write
Data Type	Structured, processed data from operational databases, applications and transactional systems	Structured, Semi-Structured and Unstructured data from sensors, apps, websites, etc.
Workload	Support batch processing as well as thousands of concurrent users performing interactive analytics	Support batch and stream processing, plus an improved capability over data warehouse to support big-data inquiries from users
Data storage	Relational database	HDFS, NoSQL, relational database
Data preparation	Aggregated	On the fly
Data integration	Quality control, filtering	No treatment
Agility	Less agile and has fixed configuration compared with data lakes	Highly agile and can configure and reconfigure as needed
Data Granularity	Data at the summary or aggregated level of detail	Data at a low level of detail or granularity
Cost/Efficiency	Efficiently uses CPU/IO but high storage and processing costs	Efficiently uses storage and processing capabilities at very low cost
Tools	Mostly commercial tools	Can use open-source tools such as Hadoop or Map Reduce

5.2. Data Lake Architecture Classification

In our review paper, we apply the term data-lake architecture only in the sense of its conceptual organization on the highest abstraction level, while excluding architectural technologies. A reference architecture as described by [31] is a conceptual framework for understanding industry best practices, tracking processes, using solution templates, and understanding data structures and elements. It is inherently abstract in that it expresses basic design considerations without any implied constraints on the realization. A reference architecture is a generic use-case free design and not a technology blueprint that dictates choices but rather a way of mapping requirements and establishing the overall pattern for implementation. In contrast, a technology architecture identifies the services or capabilities needed to support these activities, and which services to include or exclude in an implementation given particular requirements [29]. Data-lake architecture has evolved over the years with major upgrades since its first proposal in 2010, mainly driven by the demands for agility, flexibility, and ease of accessibility for data analysis. Different data-lake architectures have been proposed, each offering some merit to data storage, data analysis, and consumer (end-user) in various combinations. This section provides further discussion on eight of the most well-known proposed architectural models, each illustrating different styles and merits in the data-lake architecture space.

5.2.1. Mono-Zone Architecture

The initial concept of a data lake involves creating a centralized pool of different data sources in one location, which allows businesses to use this data for future use cases [47]. According to this early description, the initial architecture to implement the data lake featured a simple, flat design consisting of a single zone. In this mono-zone architecture, as depicted in Figure 1, all raw data are stored in their native format. This setup is often closely associated with the Hadoop ecosystem, which helps to load diverse and large-scale data at a reduced cost. However, this basic architecture has significant limitations: it lacks the capability for users to process data within the lake, and it does not track or log any user activities [30].

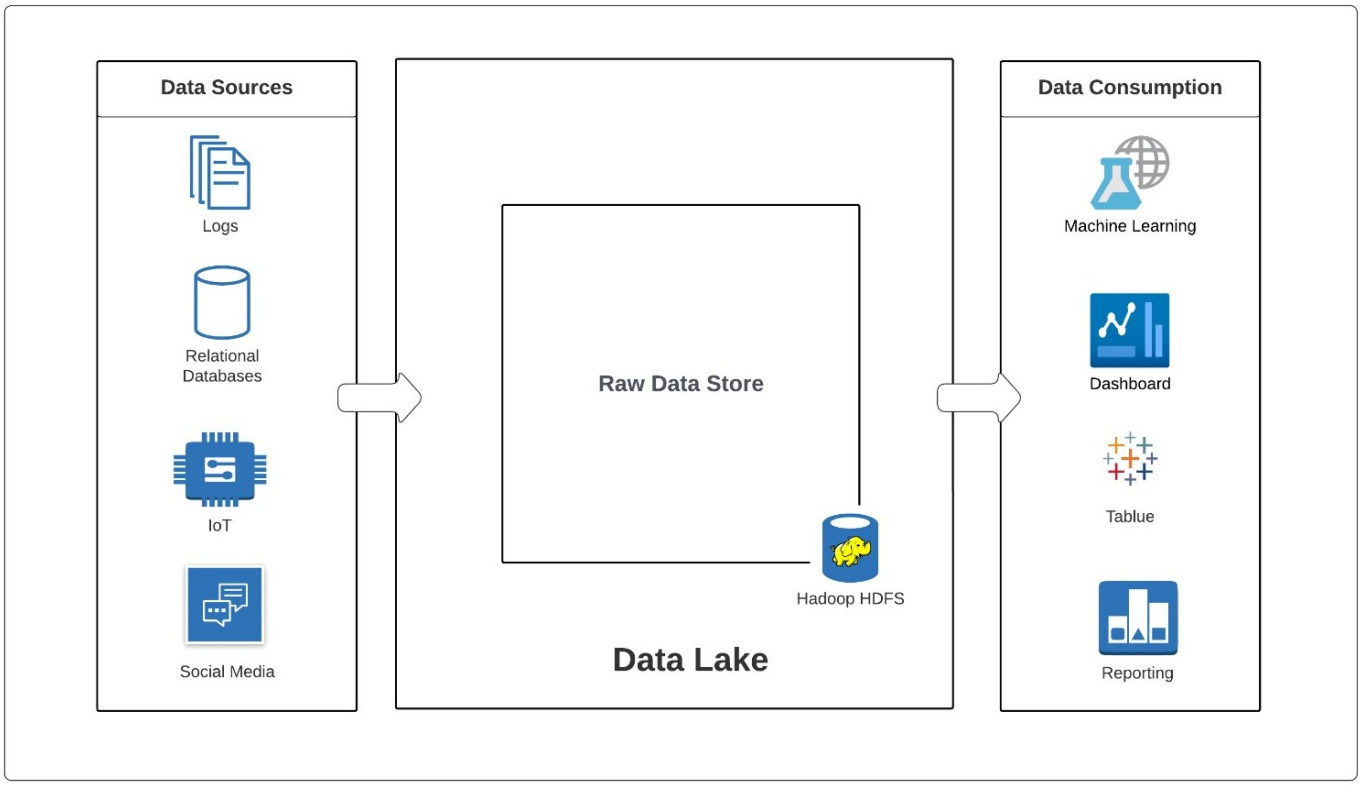

Figure 1. Mono-zone architecture.

5.2.2. Lambda Architecture

The Lambda architecture was first introduced to handle batch and real-time data processing within a single system Therefore, this architecture puts more focus on data processing and consumption, rather than on data storage [6]. The Lambda architecture, as shown in Figure 2, consists of three layers: a batch layer, a speed layer, and a serving layer. In the batch layer, the data stored in the persistent memory are available for consumption, providing a historical overview of the data. In a contrast to the batch layer, the speed layer processes only the incremental data that are still not stored in the persistent memory. Once the data are stored in the persistent memory, they stop being available in the speed layer. Together, these two layers provide the data to the end-users through the views in the serving layer [48].

5.2.3. Kappa Architecture

Kappa architecture [25], as illustrated in Figure 3, is a minimalist version of Lambda architecture which, for the sake of simplicity, removes the batch layer and retains only the speed layer. The primary goal is to eliminate the need to recompute a batch layer from scratch all the time and to try doing almost all of these processes in real-time or through the speed layer. One of the disadvantages of the Lambda architecture, which has been avoided in the Kappa Architecture, is having to code and execute the same logic twice [4].

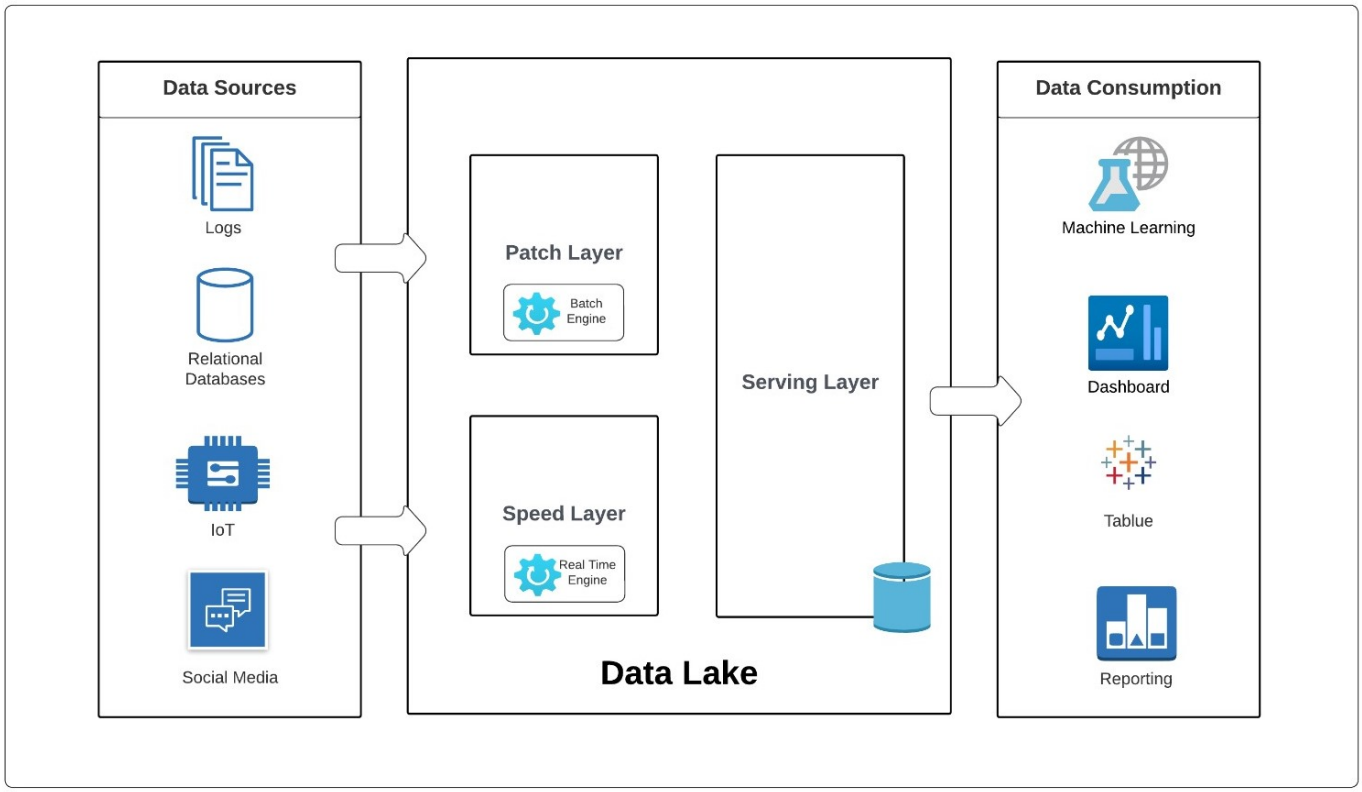

Figure 2. Lambda architecture.

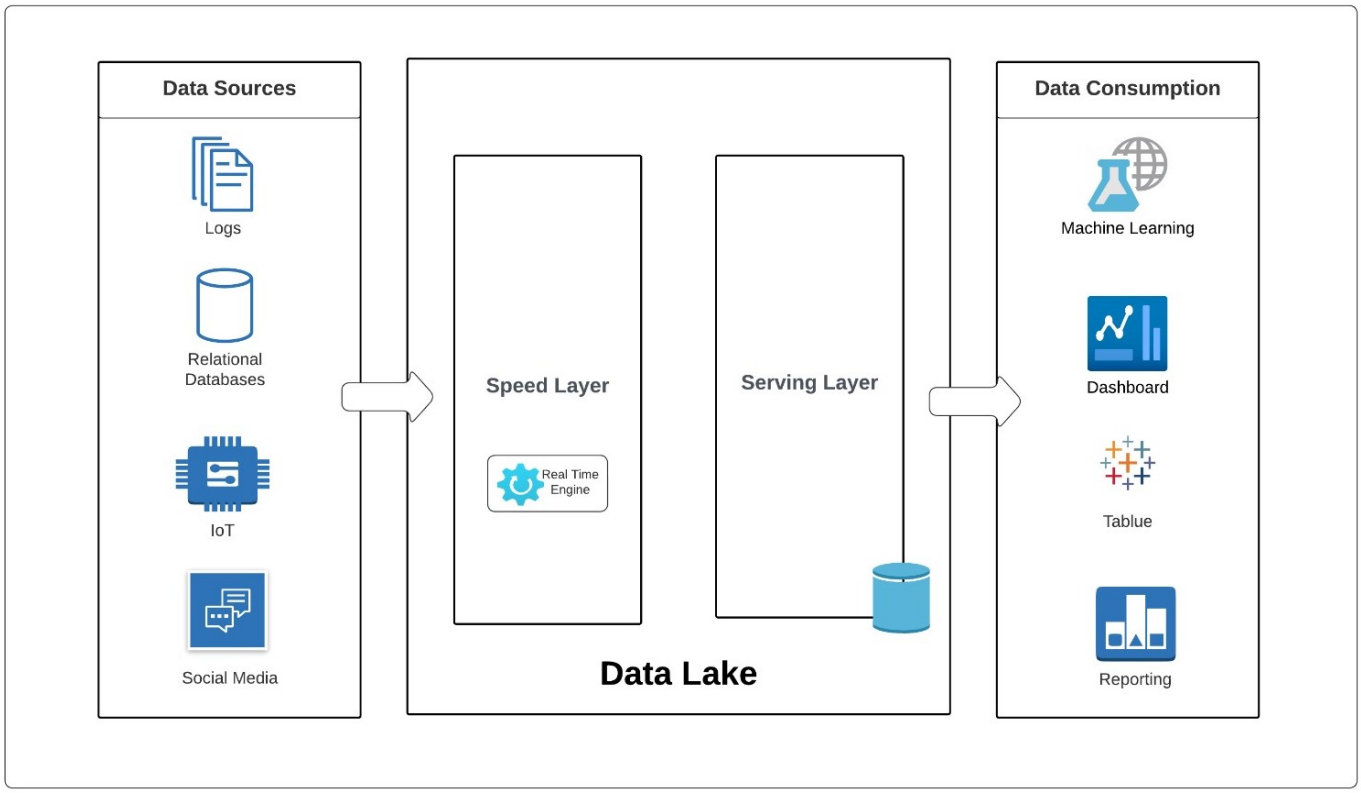

Figure 3. Kappa architecture.

5.2.4. Data-Pond Architecture

The data pond was introduced by Bill Inmon [26] as a proposed data-lake architecture model. As illustrated in Figure 4, data-pond architecture comprises five logically separated ponds, each serving a distinct purpose. The first pond is the raw-data pond, which houses the ingested raw data from sources and serves as a staging area for other ponds. Raw data are held in this pond until it is transferred to other ponds, after which the raw data are purged and inaccessible for further processing. The analog-data pond stores semi-structured, high-velocity data like those generated by IoT devices and APIs. The application-data pond functions like a data warehouse that is populated through extract–transform–load processes to provide support to businesses and existing applications. The textual-data pond contains unstructured textual data that undergoes a textual ETL process to achieve contextualization for further text analysis. The archival-data pond, on the other hand, serves to offload inactive data from the analog-, application-, and textual-data ponds and is only queried when such data are required for analysis. During the transfer from the raw-data pond, some metadata may be applied to data in the analog-data pond.

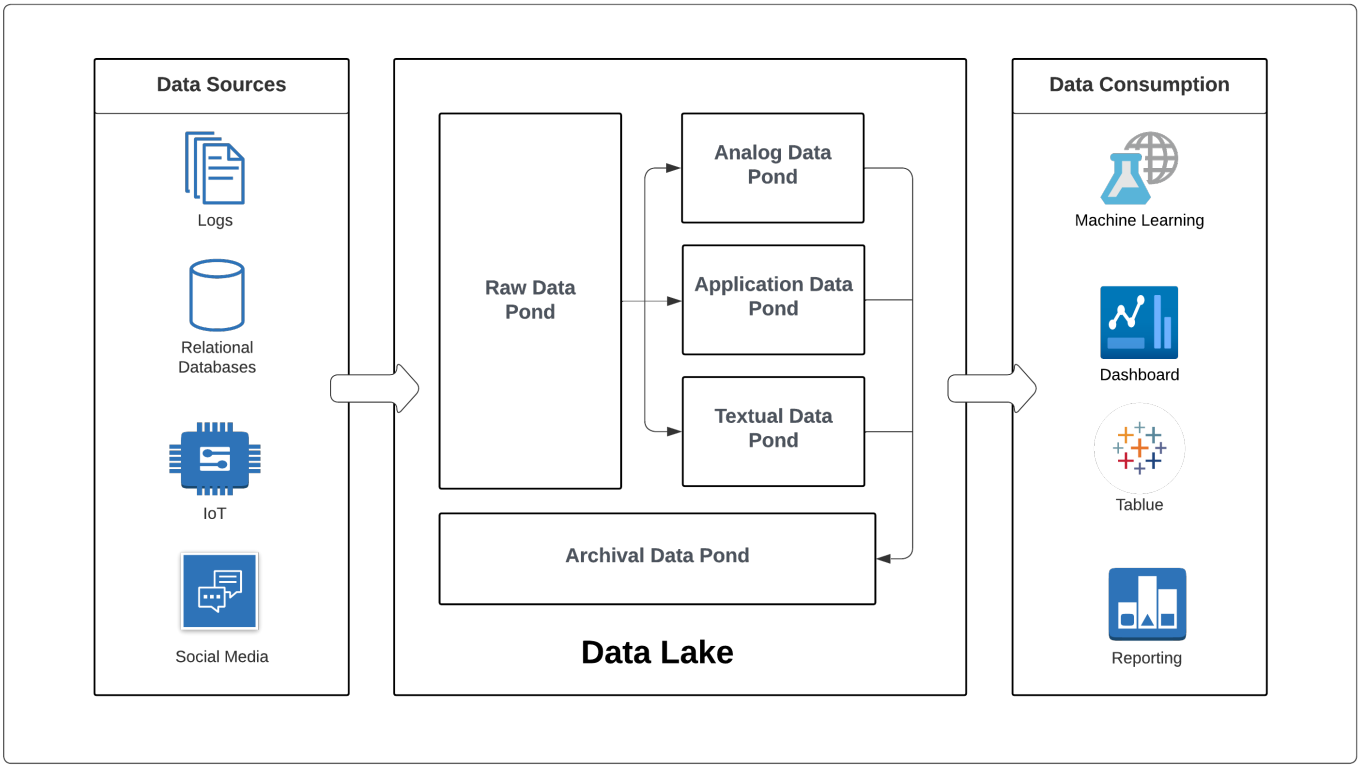

Figure 4. Data-Pond architecture.

5.2.5. Zone-Based Architecture

So-called zone architectures assign data to a zone according to their degree of refinement. Numerous variants of zone architecture have been proposed in the literature, such as the ones in [27–29,31,32]. These variants differ significantly in the zones they include, the number of zones, the user groups they support (either data scientists only or both data scientists and business users), and their focus (processing [29] versus governance [27]). However, the fundamental idea remains the same. To cite an instance, the zone architecture proposed by Zaloni [31,32] is one of the most widely used models for data lakes in recent years. This model in Figure 5 consists of four general zones and a sandbox zone, each with different data structures and uses. The transient landing zone is where the data first arrive and are temporarily stored in their raw format. The raw zone is where the raw data are permanently stored in their original form, and initial processing is done here, resulting in data indexing and record enrichment with appropriate metadata. The trusted zone holds

data that have undergone additional compliance and quality checks depending on their final purpose, and the refined zone is the source of data for users with restricted data access. Finally, the sandbox serves as a test area for ad-hoc analysis and data exploration.

The transient landing zone is similar to storage in data warehousing and includes preliminary data analysis and checks for business and technical compliance. The raw zone is the unique source of trusted data for analysis and further processing and does not check data quality or compliance. The trusted zone contains only technically and regulatory compliant data that have undergone additional compliance and quality checks, and the refined zone stores data in a form adjusted to end-users' business needs. The sandbox provides unrestricted data access for exploration and analysis purposes, but access should be restricted to users who need access to the entire data lake.

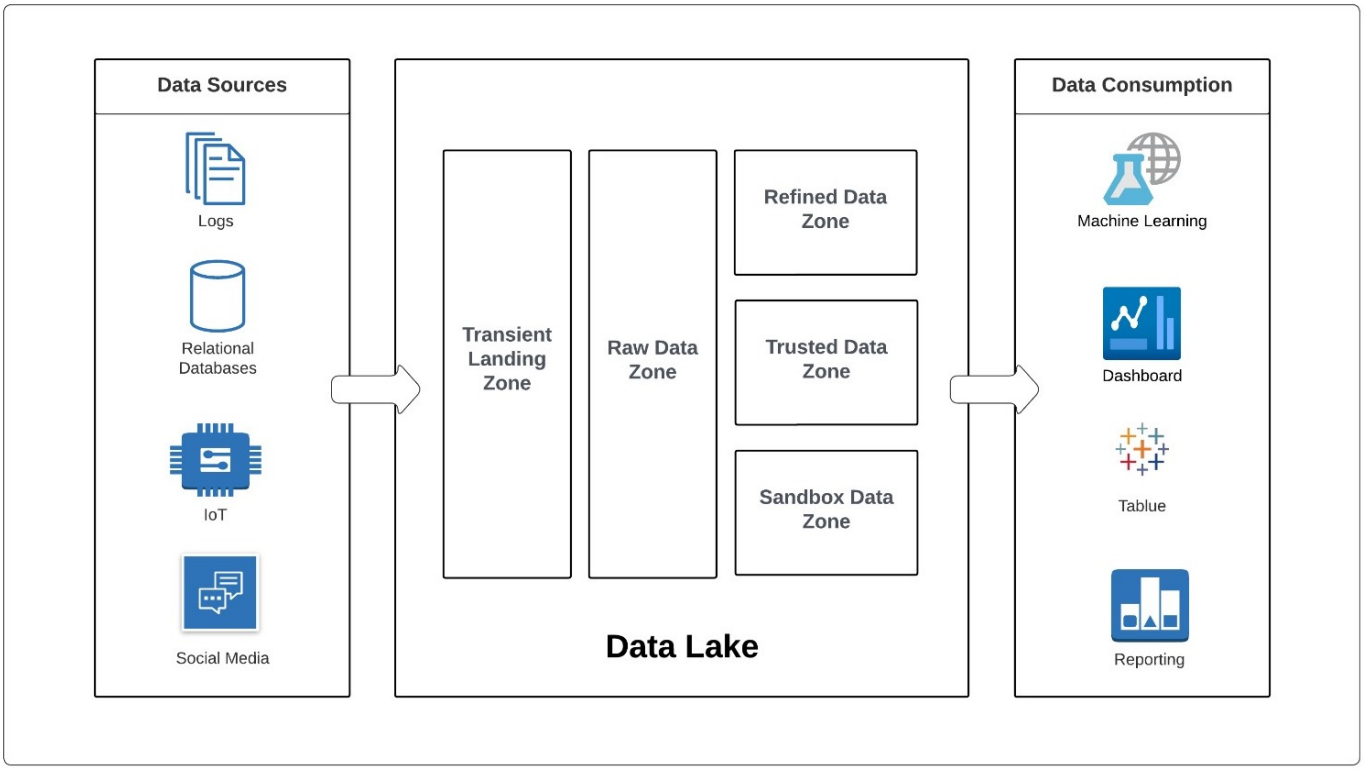

Figure 5. Zaloni data-lake architecture.

5.2.6. Multi-Zone Functional Architecture

Multi-zone functional architecture is based on the model proposed by [30]. The architecture provides an efficient means of data analytics by enabling users to easily locate, access, interoperate, and reuse existing data, data preparation processes, and analyses. As illustrated in Figure 6, it comprises several zones, including the raw ingestion zone, the process zone, the access zone, and the govern zone. The raw ingestion zone is where data are ingested and stored in their native format, whether as batch data or near real-time data. The process zone is where users can prepare data according to their needs and store intermediate data and processes, including all treatments and business knowledge applied to the raw data. The access zone is where processed data are accessible and consumable, enabling visualization, real-time analytics, advanced machine learning, BI analytics, and other decision support systems.

Finally, the govern zone is responsible for ensuring data security, quality, life-cycle, access, and metadata management, which are critical aspects of data sustainability and reuse. The govern zone comprises two sub-zones, namely metadata storage and security mechanisms, which allow authentication, authorization, encryption, and multilevel security policies to maintain quality of service through monitoring multilevel resource consumption.

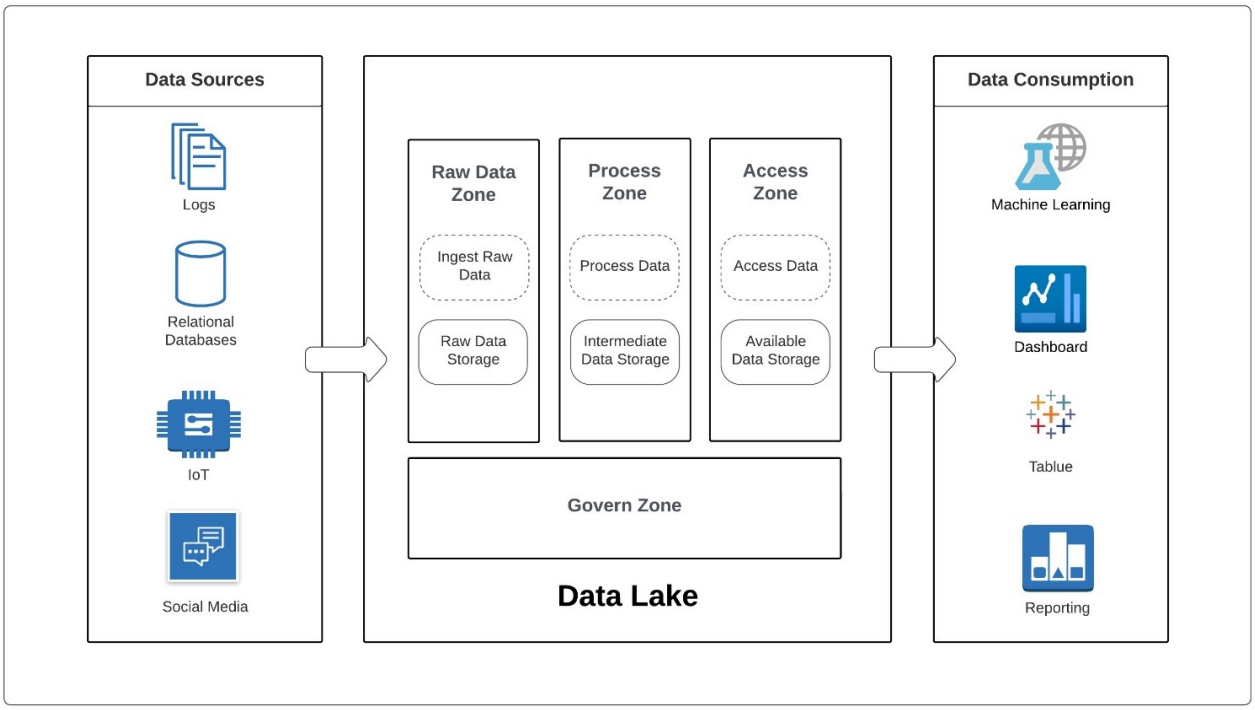

Figure 6. Multi-zone functional architecture.

5.2.7. Functional Data Lake Architecture

Multi-layered data-lake architecture, proposed by [49], uses a structured approach with distinct layers for data ingestion, storage, transformation, and interaction. This architecture ensures effective data management by separating concerns and enabling communication between layers. As reflected in Figure 7, the ingestion layer collects heterogeneous data from various sources and performs initial metadata extraction on structured and semi-structured data, which is stored in the metadata repository afterwards. The storage layer includes the metadata repository and raw-data repositories to support various data forms and structures. It also simplifies data storage complexity through a user interface that facilitates querying capabilities. The transformation layer executes data operations such as cleansing, transformation, and integration, creating models similar to data marts for user-specific data access. Finally, the interaction layer provides end-users access to the metadata repository and transformed data for data exploration, analytical queries, and visualization tasks.

5.2.8. Data Lakehouse Architecture

The data-lakehouse architecture represents a novel approach and an architectural pattern, emerging as a potential replacement for traditional data warehouses in the coming years [34]. It combines the flexibility of data lakes, allowing storage of various data formats, with the transactional integrity of data warehouses through its layered components. While the data lakehouse does not explicitly define a particular data model, it can be seen as bridging the two-layered architecture and the zone architecture by incorporating the well-defined model of the data warehouse. The data lakehouse employs a relational (structured) form as the final structure, which is chosen because most analytical and visualization tools support relational data, allowing for faster data analysis. Additionally, metadata is used to facilitate the analysis process when working with data from multiple sources.

As demonstrated in Figure 8, at its foundation, a lakehouse leverages cloud object storage to house data in open file formats like Parquet, ORC, and Avro. This open format allows any compatible processing engine to interact with the data [50]. Above the storage layer sits the transactional layer, also known as the metadata layer. This layer ensures data integrity through data-warehouse-like ACID (Atomicity, Consistency, Isolation, Durability) properties. It achieves this by managing metadata and organizing data. Additionally, it

supports schema enforcement, data versioning, and data lineage, all of which contribute to enhanced data quality. Open table format technologies like Delta Lake, Apache Hudi, and Apache Iceberg play a crucial role in this layer by bringing transactional capabilities to the data lake [51].

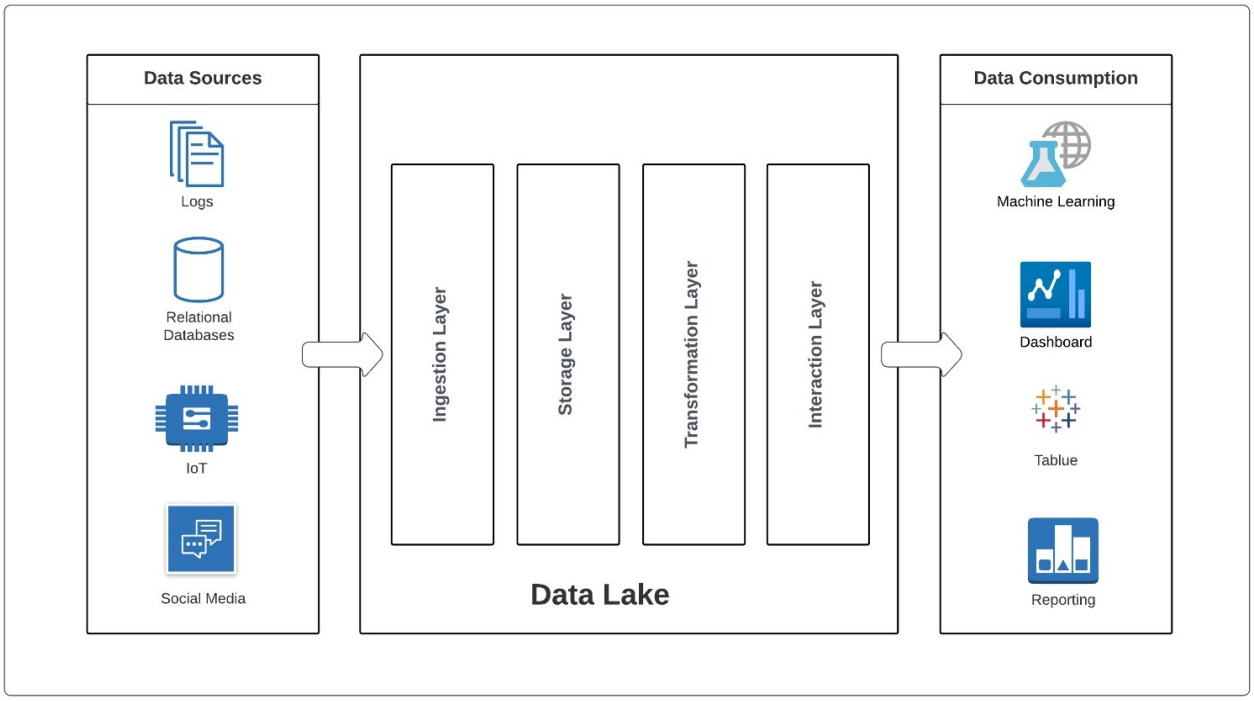

Figure 7. Functional data-lake architecture.

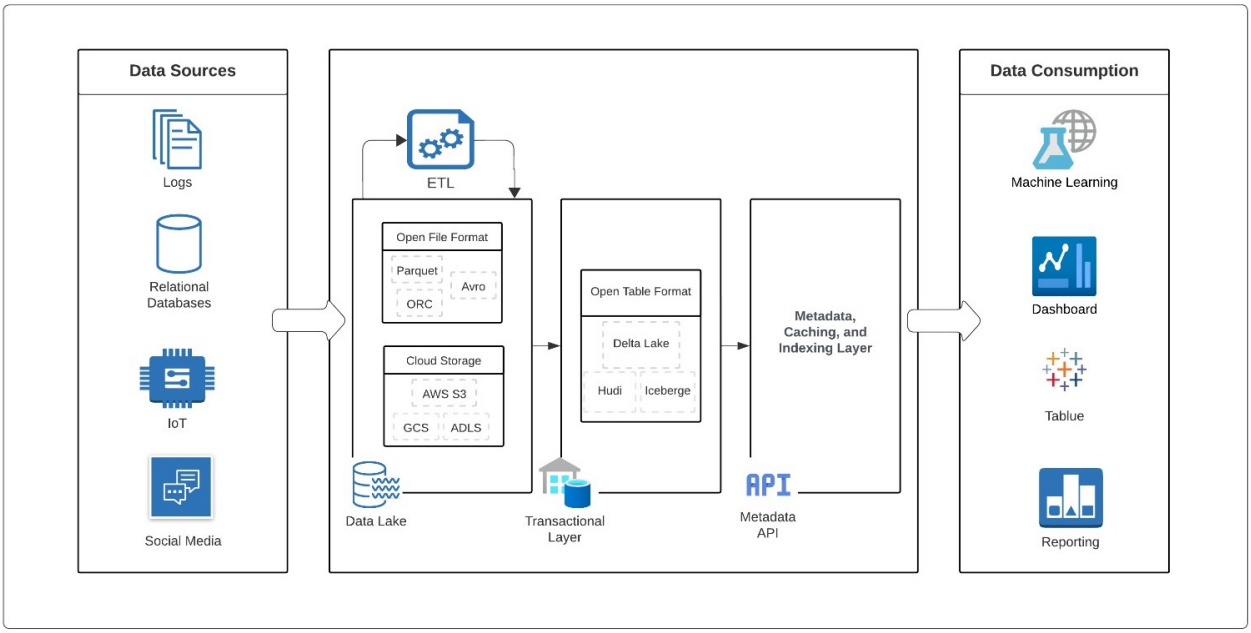

Figure 8. Data-lakehouse architecture.

To summarize, Table 2 provides a comparative analysis of different data-lake architectures, highlighting the main advantages and disadvantages of those architectures. Furthermore, we have included some use cases that offer practical insights into how each data-lake architecture can effectively apply in real-world scenarios, supporting in educated decision-making.

Table 2. Data-lake architectures model comparison.

Architecture	Advantages	Disadvantages	Use Cases
Mono-zone Architecture [30]	• Straightforward design for easy implementation and management • Cost-effective as it requires less management overhead and fewer resources • Initially simple, with the flexibility to expand by adding more zones as needed for enhanced performance	• Lacks data structure that limits the governance • Potential of data becoming unmanageable, difficult to find, and useless • Performance bottleneck as querying and processing raw data directly from a single-zone can be inefficient and slow	• Initial data exploration and discovery for hypothesis generation • Developing a proof of concept for a data lake by starting with a single-zone design [6,12]
Lambda Architecture [6]	• High-throughput processing is well suited for handling large volumes of data efficiently • Ensures fault tolerance by recomputing lost or corrupted data using historical data • Allows for time series analysis due to immutability (append-only data model)	• High processing latency due to batch re-computation • Complexity in developing and maintaining two separate processing layers (batch and speed) • Resource intensive as using different processing frameworks for the batch and speed layers increases the required hardware and synchronization efforts	• Electrical consumption forecasting in educational institutions buildings [52] • Disease Surveillance and Prediction [53] • Aviation manufacturing [24,54]
Kappa Architecture [25]	• Simple architecture with just two layers • Flexibility in handling evolving processing and analytics requirements which can cope with diverse analytics scenarios • Lower complexity and overhead in maintaining different processing pipelines • Resource efficient as it requires only one processing engine	• Recomputing the entire history increases linearly with the growing volume of data • Unsuitable for some ML algorithms where streaming and batch outputs differ • Does not fit data analytics scenarios where full historical data processing is required	• Electricity load forecasting in smart homes context [55] • User-centric IoT analytics platform to detect outliers in the electricity consumption consumption [56]
Data-Pond Architecture [26]	• Enhance data management and retrieval by organizing data based on its type and purpose • Facilitates better governance practices as data are managed according to their type and usage requirements • Eliminates data redundancy because data are stored only in the pond that suits their type • Each pond has a fixed infrastructure, including metadata and metaprocess definitions to support scalability	• Loss of original raw data as this design mandates transforming data after it leaves the raw pond, contradicting the core data-lake concept that data should be stored permanently in its native form for future reference and integrity • Increase overall management and operational costs as each pond may require distinct infrastructure and resources	• Analyzing cultural heritage data [57] • Processing of multimodal biological data [58] • Power Grid Monitoring and Diagnostic System [59]
Zone-Based Architecture [27–29,31,32]	• Enhanced data quality and security as data undergoes through several layers of processing and validation • Better data governance as metadata can be applied throughout the zones to ensure that data lineage, quality, privacy, and role-based security measures are maintained consistently • Supports both operational use cases and ad hoc exploratory analysis through the use of sandbox and explorative zones. This allows data scientists to perform advanced analytics without interfering with the production environment • Optimized storage costs and improved performance by categorizing data into hot and cold zones	• Adds complexity and maintenance overhead due to varying management needs for each zone • Data duplication across multiple zones significantly challenges data lineage management, increases storage costs, and risks data inconsistencies • Delayed data availability as data must pass through several zones before it becomes available for analysis, which can be a drawback for use cases requiring real-time or near-real-time data access • Requires domain expertise in integration, metadata management, and enforcing governance across zones	• Fish farming management [37] • Banking Data Management [60] • Cardiovascular disease prediction [38]
Functional Data Lake Architecture [49]	• Layering provides a logical framework for grouping services that provide similar functionality, which can enhance clarity and navigability for users and administrators • Due to clear separation of concerns, each layer can be independently maintained, upgraded, troubleshooted, and scaled, ensuring efficient resource utilization and cost optimization while minimizing disruptions to other layers • The inherent modularity enables the selection of technologies tailored to each layer's specific requirements, thus allowing for greater specialization and optimization	• Absence of a clear data organization within the storage layer can hinder the implementation and the enforcement of strict data governance policies across layers • Oversimplifies data flow as a linear progression (ingestion—storage—processing—consumption), neglecting the iterative nature of data cycling between storage and processing, which can lead to inefficient resource utilization	• Environmental Monitoring [61] • NetFlow-based Cyberattack Detection [62] • Energy Optimization Analytics [63]
Multi-Zone Functional Architecture [30]	• Offers the flexibility and scalability of the functional approach, along with the enhanced governance and data lifecycle management benefits of the zone-based approach • Supports various use cases, from simple data lakes for small teams to complex enterprise-level data platforms with strict governance requirements • Unlike purely functional architectures that often depict a linear flow, the hybrid approach explicitly showcases the continuous data processing loop, offering a more realistic depiction of how data are refined and enriched over time	• Complex to design and implement compared to purely zone-based or functional-based models • Lacks built-in mechanisms for ensuring data consistency and isolation, which can lead to challenges with managing concurrent workloads and maintaining data integrity	• Web server access log file management [64] • Air Traffic Management [39] • Real-Time IoT Sensor Data Management for University Buildings [40] • Healthcare data management and analytics [65]
Data Lakehouse Architecture [34]	• Simplifies enterprise analytics architectures, which are often costly and slow due to separate data platforms (warehouses and lakes) required for analytical workloads • Reduces complexity by combining features of data warehouses and lakes into a single platform • Avoids redundant storage of multiple data copies across several platforms, which helps maintain a single source of truth • Provides a unified data format across the platform to reduce errors in data handling • Facilitates near-real-time analytics and advanced data processing features like stream processing • Offers a single point of access to data for users, improving ease of use and transparency across various data analysis tasks	• Transformation of data to a uniform format during ingestion can introduce errors due to differences in source and target data types • Potential loss of read and write performance depending on data size and technology stack optimizations • Requires significant adjustments and careful design to efficiently implement functionalities that cater to both batch and stream processing without compromising performance	• Biomedical research and health data analytics [66] • Vessel Monitoring System [67]

6. Timeline of Data-Lake Architecture Development

The understanding of the data-lake vision has shifted away from considering it solely as a storage solution to recognizing it as a data platform. The purpose of a storage solution is to fulfill specific requirements for holding data, whereas a data platform is designed to accommodate the needs of various applications with more generalization and support for diverse data processes, analysis, insights, and use cases [29]. The objective of this section is to map the progression of proposed architectures for data lakes. This has been accomplished, as shown in Figure 9, by creating a chronological timeline graph and categorizing the evolution into four distinctive phases. Additionally, we have explored potential technological landscapes and business requirements that drive such development.

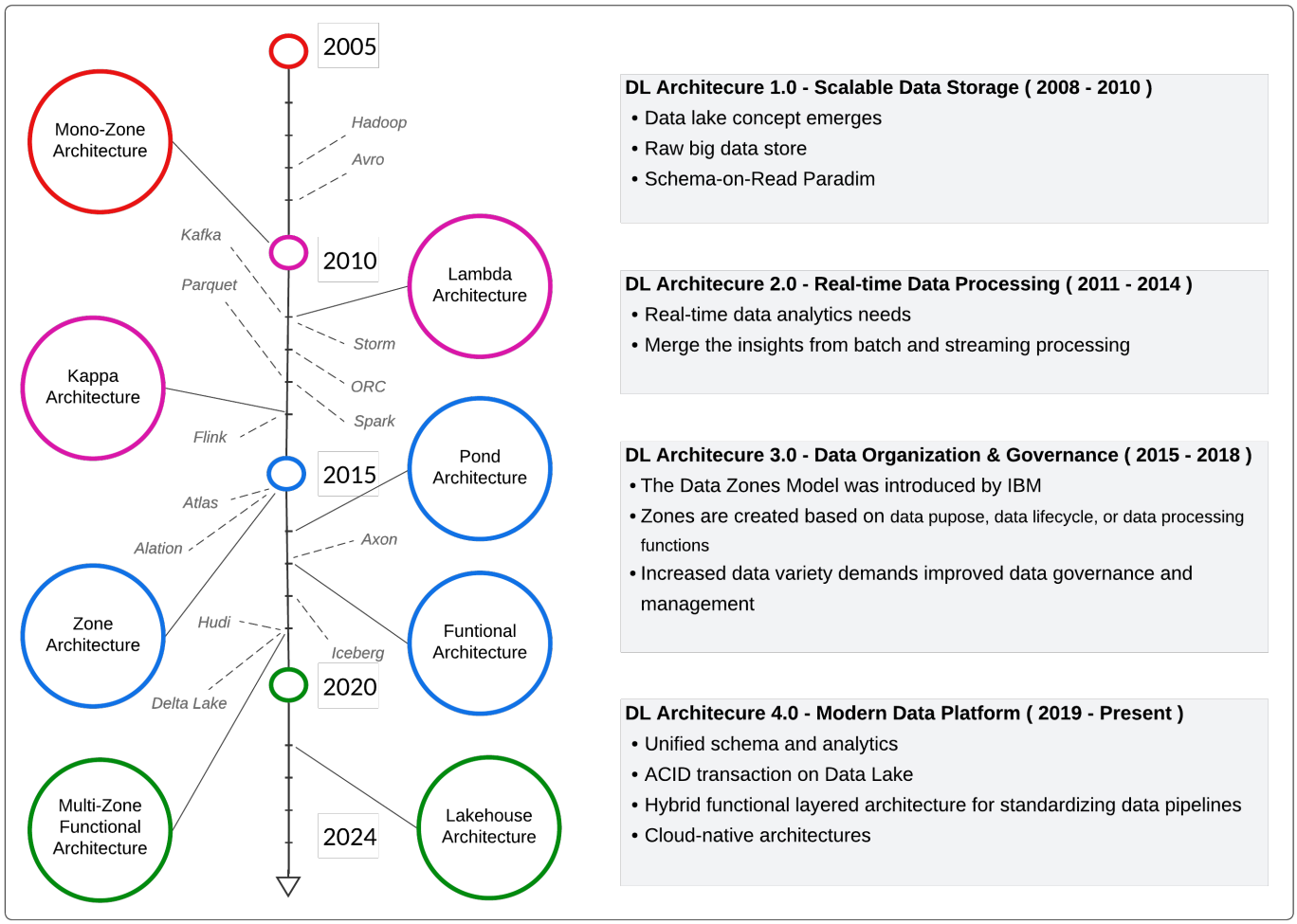

Figure 9. Chronological timeline graph for data-lake architectures.

6.1. Dl Architecture 1.0—Scalable Data Storage (2008–2010)

The first phase of data-lake architecture, introduced between 2008 and 2010, laid the foundation for scalable data storage through the utilization of big-data technologies that came in the form of the Apache Hadoop stack. This stack is represented by its distributed file system and MapReduce, which have sat the grounds for the concept of data lakes. This new vision was first coined in the industry in 2010 by Pentaho CTO James Dixon, as a solution that handles raw data from a single source and supports diverse user requirements [7]. It is a sharp contrast to data warehouses for which the structure and usage of the data must be predefined and fixed, and rigorous data extraction, transformation, and cleaning are necessary. Data lakes, on the other hand, could avoid the expensive standard processes of data warehouses, such as data transformation, by storing raw data in the original format. This enables the adaptation of the schema-on-read paradigm, which

defers schema application until data retrieval, thus offering flexibility in handling diverse data types.

The initial implementation of the data lake, driven by this vision, featured a flat architecture with a mono-zone approach, as described in [30]. The main focus of such a design was to fulfill business and data demands by capturing and storing massive amounts of raw, unstructured data in their original formats. This facilitated the future analysis of large datasets for insights without the constraints imposed by traditional database systems, thereby breaking down silos and providing reusable data assets for various business functions.

The original Apache Hadoop stack heavily relied on a tight coupling of storage and computing within the nodes of a cluster, meaning that each Hadoop cluster node had storage and computational power. Although this model functioned well for batch processing and large-scale data analytics, it presented challenges in independent resource scaling. For instance, if a workload required additional computer power but not additional storage, the entire node (with both storage and compute) had to be scaled up, leading to inefficient resource utilization [68]. In addition, with ongoing changes in data processing capabilities and processing demand, achieving high real-time performance in specific scenarios was a challenge, regardless of improvements in batch processing. Stream computing engines addressed this limitation, pushing the data-lake processing paradigm to its next evolutionary stage.

6.2. Dl Architecture 2.0—Real-Time Data Processing (2011–2014)

In response to the growing need for both batch and streaming data processing requirements, the second generation of data-lake architectures prioritized improvements to address the challenge of combining these requirements for timely insights and faster response times. Nathan Marz proposed the Lambda architecture [23] as a data processing architecture to address these new business needs by allowing a distributed data store to be simultaneously consistent, highly available, and partition tolerant. The Lambda architecture achieves this by creating two streams for the same input data: one is processed using a real-time framework and the other one is processed with a batch framework, which are then combined into a unified view. This architecture ensures data accuracy and low-latency processing [24]. Although the Lambda architecture successfully provided a solution, its complexity and the need for separate batch and streaming pipelines constrained its maintainability. Consequently, Jay Kreps, a co-founder of Apache Kafka, introduced the Kappa architecture [25] in a 2014 blog post. This architecture utilizes a single and unified processing pipeline for both real-time and batch processing and eliminates the necessity for a separate batch layer that can cause latency and data processing delays.

In general, reliability, scalability, and maintainability are the three most important considerations in the software engineering of these approaches. Technologies such as Apache Kafka for distributed streaming, Apache Parquet and ORC for open data storage formats, Apache Storm for real-time computation, and Apache Spark for unified batch and stream processing played crucial roles in enabling these architectures' capabilities.

6.3. Dl Architecture 3.0—Data Organization & Governance (2015–2018)

At the beginning of 2015, there were more debates on the absence of proper data governance mechanisms that initial data-lake implementations suffer from, turning them into unusable data swamps. As data continues to grow in variety and complexity, data-lake architectural proposals focus on data organization that enhances governance practices. Most architectural models proposed during this phase were originally inspired by industrial implementations of data-lake systems in [28,29], with a set of remarkable additions to data-lake research. Among these was the suggestion of having a shared platform equipped with subsystems that support the distinct needs a data lake should deliver, including acquiring data, processing data, and delivering data for use. Each sub-system could be represented by a set of technology components, rather than prescriptive blueprints or

products. These proposals emphasized that the core of the data lake is its data architecture, which is the structure of the data arranged to permit its three primary functions: acquiring data, managing data, and utilizing data. This arrangement minimizes reprocessing and user effort by separating components that change at different rates due to varying demands. Separating the storage and models used when the data are collected from those used to normalize or deliver the data facilitates this adaptability. This pattern, known as zone-data architecture, establishes discrete zones for storing data that correspond to these component functions.

There are several other high-level proposals for data-lake architecture that establish delineations for datasets. One proposal segments ingested data by type and use [26], while another organizes data by the extent to which they are refined [31]. For instance, there may be zones for loading and quality checking, raw-data storage, refined and validated-data storage, discovery and exploration, and business or scientific analysis of data. This zone architecture, when applied to the latter proposal, enables incremental and progressive refinement of data quality and structure as data pass through each zone. By standardizing and cleansing data in a progressive manner, the architecture facilitates easier management of the data lifecycle and governance and security policies by applying distinct rules and permissions to each zone. However, this results in multiple copies of data, which complicates data lineage and analysis and becomes increasingly costly as the number of zones is increased. Additionally, this proposal lacks technical detail concerning the functions, which impedes modular and reusable implementations. Addressing both of these deficiencies, a function-oriented architecture proposal [49] establishes a four-layer mapping between data-lake users and a storage layer consisting of potentially multiple technologies, including data ingestion, storage, transformation/processing, and access. This architecture is more transparent in matching the required technologies since each layer has clearly defined functions. During this phase of the evolutional journey of data-lake architecture, businesses encountered increasing regulatory demands and needed data governance frameworks to ensure that data quality, security, and compliance are maintained. This historical development resulted in the recognition of the need for controlled and structured data environments to enable advanced analytics while maintaining regulatory compliance. Governance tools like Apache Atlas for data governance and metadata management, Informatica Axon Data Governance, and Aletion Data Catalog have enabled this structured paradigm for data management in data lakes and addressed these concerns. While the this phase established the foundational technologies and processing frameworks for handling large volumes of streaming and batch data, the rising demand for effective data governance and quality standards set the stage for the subsequent evolution, which emphasized creating more manageable and compliant data environments.

6.4. Dl Architecture 4.0—Modern Data Platform (2019–Present)

The growing demand for machine-learning (ML) and large language-model (LLM) workloads was the primary driver pushing businesses to enhance their data-lake architectures and adopt robust data infrastructures. Such infrastructure must prioritize provisioning high-quality, voluminous, and diverse data, recognized as the raw material for training effective models. Furthermore, it is crucial to enforce data governance policies, including security and privacy measures, across the data lifecycle to ensure data integrity and compliance.

Early on-premises, Hadoop-based data lakes suffered from poor performance due to the need to replicate data from an external repository to another storage-compute node for analytics. Additional common problems with traditional data lakes included management complexity, scalability limitations, performance constraints, and data inconsistency. Building and maintaining data pipelines within these on-premises environments was complex, as it involved managing both hardware infrastructure (such as provisioning and configuring servers, scheduling batch Extract, Load, Transform (ELT) jobs, and handling outages) and software components (such as integrating divers tools for data ingestion, organization,

preprocessing, and querying) [68]. Furthermore, scaling on-premises data lakes was a manual process that required adding and configuring new servers, demanding constant monitoring of resource utilization, and resulted in increased maintenance and operational costs, primarily related to IT and engineering resources. Performance limitations often arose under high concurrency or when dealing with large datasets. The complex architecture could lead to broken data pipelines, slow data transformations, and error-prone data movement outside the data lake. These issues, coupled with the rigid architecture, also increased the risk of data governance and security breaches [68]. The absence of ACID transaction support in traditional data lakes resulted in the potential for partial updates or data corruption during concurrent writes. As a result, the reliability of data queries and analyses was compromised, impacting the effectiveness of machine-learning models trained on this data [34].

Consequently, the migration of on-premises data lakes to the cloud as infrastructure-as-a-service offerings gained significant momentum and accelerated considerably. The increasing volume of data, the need for scalable storage solutions, and the maturation of cloud service offerings for handling both structured and unstructured data have motivated this shift. Cloud platforms now deliver comprehensive services for the entire data journey, from ingestion and storage to processing and analysis [68]. Moreover, cloud service providers continually enhance security features to guarantee high levels of data protection [69]. End-to-end cloud-native data lakes offer several benefits, including no data restrictions, a single storage layer without silos, the flexibility to run diverse compute tasks on the same data store, independent scaling of compute and storage, reduced costs through elastic scaling, and the ability to leverage advanced analytics and machine-learning tools [70].

Modern cloud data-lake architectures leverage two key hybrid approaches. The first approach, known as multi-zone functional architecture, integrates zone-based data organization (raw, curated, processed) with dedicated functional layers for tasks like ingestion, processing, and analysis for a more comprehensive data-lake model [30,64,71]. The second approach, the data lakehouse, focuses on unifying data access by integrating the cost-effective and flexible storage of a data lake with the transactional consistency (ACID guarantees) offered by a data warehouse [34]. The data-lakehouse model is gaining significant traction in both academic research and industry applications. This is attributed to its capability to unify schemas and seamlessly combine both streaming and batch data processing, ensuring reliable and consistent outcomes. The data lakehouse also utilizes the Unity Catalogue for data governance to handle permissions and access controls across workspaces. Other data-lakehouse benefits comprise machine-learning support, data openness promotion, and performance optimization. Hudi, Delta Lake, and Iceberg are table formats that enable ACID transactions, which serve as the foundation for building data lakehouses. These transactions rely on detailed transaction logs that record all data modifications within the lake. In the event of failures or errors, these logs enable the rollback or replay of operations. This feature simplifies data pipelines and accelerates SQL workloads by eliminating the need for complex error handling and data reconciliation processes typically required to ensure data consistency in non-ACID environments [51]. Businesses now require scalable, flexible, and cost-effective solutions that provide robust performance and governance to enable them to leverage data as a strategic asset across all operational aspects. This evolution underscores the ongoing refinement of data-lake architectures to meet modern data management challenges and business needs.

7. Conclusions

This section addresses the importance of data lakes in modern data architecture, along with the challenges organizations may encounter while implementing and managing them. Furthermore, we have discussed the future development trends of data-lake implementation and demonstrated how integrating AI and ML directly within data lakes revolutionizes

their functionality. Finally, we have suggested some avenues for future research that build upon our current understanding and expand the scope of data-lake implementations.

7.1. Significance of Data Lakes in Modern Data Architecture

The diversity of data sources requires new integration solutions that can handle the data volume, velocity, and variety created by distributed systems. Schema-first integration, utilized by data warehouses and implemented using ETL (Extract-Transform-Load) frameworks, is not agile or dynamic enough to cope with the data management cycle, making it unsuited to the integration of modern data sources. This is where a data lake excels, offering the ability to integrate, manage, and analyze any data from different sources [4].

The advent of large data lakes represented a paradigm shift in the way that organizations would manage and analyze their data assets. First generation data lakes were built in corporate data centers using large nodes and, while ambitious in intent, these early initiatives often conflicted in their storage and serving requirements. For example, the deep analytic queries required by a data scientist working on predictive analytics for revenue modeling clashed with the fast API responses needed for customers facing applications using edge computing. This was largely due to the tightly coupled storage and computing architecture of the Apache Hadoop stack, upon which most of these implementations were based [51].

While first-generation data lakes struggled with these inherent conflicts, the underlying principle of the data lake represented a step change in data management. As data lakes matured, they soon became perceived as a major pillar in modern data architecture because they enabled enterprises to handle the vastness of the data they wished to capture, store, and process [42]. Data were landing in their native unstructured format, thus maintaining the full integrity of the data and avoiding unnecessary data loss due to pre-conceived pre-processing and transformation steps. Instead, the data could then be integrated and analyzed when required in a schema-on-read layer, thus enabling agility in adapting to new data sources and changing business needs.

Data lakes also simplify the integration of multiple data sources, thereby enabling the enterprise to insightfully discover patterns and relationships between differing data types, in particular unstructured data, which constitutes many big-data sources. Due to the heterogeneity of the data types and formats that a data lake supports, the analysis is quicker and large-scale data analysis is more precise, leading to improved and informed decisions [12].

Another key advantage of a data lake is that it reduces reliance on proprietary data-warehousing solutions, which typically involve heavy investment in hardware and software. Data lakes instead leverage open-source tools and commodity hardware, making them far more cost-effective for storing and managing extremely large volumes of data.

Lastly, data lakes facilitate real-time data ingestion, with no transformations, thus removing the latency between extraction and ingestion and enabling timely decision-making [27]. A data lake and the analysis performed around it are typically implemented using distributed systems, which provide high scalability and resilience to hardware and software failures.

7.2. Data Lake Challenges

Despite all the benefits of data lakes discussed in Section 7.1, there are also some serious issues that organizations may encounter when deploying and maintaining their data lakes. First of all, data quality is considered a major concern [70]. Since data lakes store raw, unstructured data, it is hard to control the consistency and accuracy of such data. Therefore, organizations need to define policies for data governance to maintain data quality, which might take time and incur high costs. Another issue is data security, as data lakes accommodate users who have easy access to data, making them prone to security attacks [72]. Hence, proper security features should be developed to safeguard sensitive data from unwanted users.

Moreover, data lakes do not possess any standardized schema or data models, making it hard to integrate data from various sources. Lack of visibility into the data stored in the data lake is another issue. Data warehouses, on the other hand, came with governance built-in, while early data lake use cases often had no attempt to manage the data. This led to a data swamp with poor visibility into data type, lineage, and quality [4].

Similar to data warehouses, governance practices are often hard to deploy and enforce as metadata is not automatically attached to the data when ingested into the data lake. As a result, it is not possible to structure the data lake and apply governance policies [48]. Metadata helps organizations implement data privacy and role-based security, which are essential for organizations operating in heavily regulated industries. To effectively utilize metadata, organizations need to integrate the data lake with existing metadata utilities in the overall ecosystem to track data usage and transformations outside the data lake.

Lastly, creating a big-data-lake environment is not simple and involves integrating many different technologies. As a consequence, the strategy and architecture are also complex, as organizations need to integrate existing databases, systems, and applications to break data silos, automate and operationalize certain processes, and implement and enforce enterprise-wide governance policies. However, most organizations lack the necessary in-house skills to successfully execute an enterprise-grade data-lake project, which might lead to expensive errors and delays [5].

7.3. Future Data Lake Developments Trends

The future of data lakes is dynamic and constantly changing, influenced by several important trends. One major shift is the increasing integration of AI and ML capabilities directly within data lakes, allowing for end-to-end workflows and real-time insights from vast amounts of structured and unstructured data. This integration requires APIs to improve data accessibility for downstream AI/ML applications and scalable storage solutions optimized for AI, such as vector and tensor storage [73]. Such storage solutions are necessary for deep learning because tensors can manage not only embedding vectors but also input datasets and model parameters like weights and biases. It is a unified approach to handling multi-dimensional data. The democratization of data is also gaining momentum for empowering business users with self-service analytics tools and intuitive interfaces to enable non-technical users to perform analyses and generate insights independently. Some direction in this area focuses on developing NLP-based interfaces and low-code/no-code platforms to make data accessible to a wider audience [74]. Furthermore, organizations are striving to create unified data ecosystems that seamlessly integrate data lakes with other data management and analytics tools. This involves integrating with data warehouses, metadata management systems, and AI/ML platforms, with research focusing on metadata exchange standards and data-lakehouse architectures for enhanced interoperability and a unified view of data [75]. Advances in AI can also provide opportunities to revolutionize data management within data lakes. AI-driven automation can simplify data ingestion, ensuring data are loaded accurately and efficiently into the lake. AI algorithms also help in identifying and resolving data quality issues such as error detection and correction [76] and data cleaning [77] so that analysis can maintain high standards. Automated metadata generation and management through AI tools simplifies data understanding and utilization. The augmented data catalog represents one of the innovations focusing on this trend. The augmented data catalogs leverage machine-learning capabilities to automatically manage the discovery, inventorying, profiling, tagging, and creation of semantic relationships among distributed and siloed data assets. The goal is to create active metadata that can guide and automate data management tasks.

7.4. Future Research Directions

Future research work that expands on our current work may include conducting a multidisciplinary literature review that integrates insights from diverse fields such as big-data analytics, data governance, distributed computing, cloud computing, data integration,

AI and machine learning, edge computing, semantic web and ontologies, cybersecurity, and human–computer interaction, which could significantly enhance data-lake research. This comprehensive approach could lead to the development of advanced data-lake solutions that address challenges related to data management, security, interoperability, and user experience. Another important research initiative could involve developing standardized reference architectures and implementation frameworks, which are crucial to guide organizations in setting up and managing data lakes effectively. These frameworks should encompass best practices and guidelines for data modeling, metadata management, and integration with existing systems. Additionally, establishing industry standards and compliance requirements will ensure data quality, security, and interoperability across different data-lake implementations. Future research should prioritize a thorough, in-depth evaluation of the technological tools used in data-lake implementations. This evaluation should include feature analysis, performance benchmarking, and suitability assessment for various use cases, such as real-time analytics and machine learning. Practical case studies demonstrating real-world implementations and a detailed comparative analysis of different tools can help practitioners make informed decisions when choosing the most appropriate solutions for their specific needs.

Author Contributions: Conceptualization, A.O., S.A. and Z.A.; Methodology, A.O., S.A. and Z.A.; Validation, A.O., S.A. and Z.A.; Formal analysis, A.O., S.A. and Z.A.; Investigation, A.O., S.A. and Z.A.; Resources, A.O., S.A. and Z.A.; Data curation, S.A. and Z.A.; Writing—original draft, S.A. and Z.A.; Writing—review & editing, A.O., S.A. and Z.A.; Visualization, A.O., S.A. and Z.A.; Supervision, A.O.; Project administration, A.O.; Funding acquisition, A.O. All authors have read and agreed to the published version of the manuscript.

Funding: The study is funded by the Libyan Ministry of Higher Education and Scientific Research.

Acknowledgments: The first and second authors would like to express their sincerest appreciation to the Libyan Ministry of Higher Education and Scientific Research for their scholarship and financial support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Naeem, M.; Jamal, T.; Diaz-Martinez, J.; Butt, S.A.; Montesano, N.; Tariq, M.I.; De-la Hoz-Franco, E.; De-La-Hoz-Valdiris, E. Trends and future perspective challenges in big data. In *Advances in Intelligent Data Analysis and Applications, Proceeding of the Sixth Euro-China Conference on Intelligent Data Analysis and Applications, Arad, Romania, 15–18 October 2019*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 309–325.
2. Data Growth Worldwide 2010–2025 | Statista. Available online: https://www.statista.com/statistics/871513/worldwide-data-created/ (accessed on 27 May 2024).
3. Sivarajah, U.; Kamal, M.M.; Irani, Z.; Weerakkody, V. Critical analysis of Big Data challenges and analytical methods. *J. Bus. Res.* **2017**, *70*, 263–286. [CrossRef]
4. John, T.; Misra, P. *Data Lake for Enterprises*; Packt Publishing Ltd.: Birmingham, UK, 2017.
5. LaPlante, A. *Architecting Data Lakes*; O'Reilly Media: Sebastopol, CA, USA, 2016.
6. Liu, R.; Isah, H.; Zulkernine, F. A big data lake for multilevel streaming analytics. In Proceedings of the 2020 1st International Conference on Big Data Analytics and Practices (IBDAP), Bangkok, Thailand, 25–26 September 2020; pp. 1–6.
7. Dixon, J. Pentaho, Hadoop, and Data Lakes. Available online: https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/ (accessed on 4 May 2024).
8. Chessell, M.; Jones, N.L.; Limburn, J.; Radley, D.; Shank, K. *Designing and Operating a Data Reservoir*; IBM Redbooks: Indianapolis, IN, USA, 2015.
9. Fang, H. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In Proceedings of the 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Shenyang, China, 8–12 June 2015; pp. 820–824.
10. Couto, J.; Borges, O.T.; Ruiz, D.D.; Marczak, S.; Prikladnicki, R. A Mapping Study about Data Lakes: An Improved Definition and Possible Architectures. In Proceedings of the SEKE, Lisbon, Portugal, 10–12 July 2019; pp. 453–578.
11. Sawadogo, P.; Darmont, J. On data lake architectures and metadata management. *J. Intell. Inf. Syst.* **2021**, *56*, 97–120. [CrossRef]
12. Mehmood, H.; Gilman, E.; Cortes, M.; Kostakos, P.; Byrne, A.; Valta, K.; Tekes, S.; Riekki, J. Implementing big data lake for heterogeneous data sources. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering Workshops (Icdew), Macao, China, 8–12 April 2019; pp. 37–44.

13. Zagan, E.; Danubianu, M. From Data Warehouse to a New Trend in Data Architectures–Data Lake. *IJCSNS Int. J. Comput. Sci. Netw. Secur.* **2019**, *19*, 30–35.
14. Herden, O. Architectural patterns for integrating data lakes into data-warehouse architectures. In Proceedings of the Big Data Analytics: 8th International Conference, BDA 2020 (Proceedings 8), Sonepat, India, 15–18 December 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 12–27.
15. Nambiar, A.; Mundra, D. An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management. *Big Data Cogn. Comput.* **2022**, *6*, 132. [CrossRef]
16. Harby, A.A.; Zulkernine, F. From data warehouse to lakehouse: A comparative review. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 389–395.
17. El Aissi, M.E.M.; Benjelloun, S.; Loukili, Y.; Lakhrissi, Y.; Boushaki, A.E.; Chougrad, H.; Elhaj Ben Ali, S. Data lake versus data warehouse architecture: A comparative study. In *Proceedings of the WITS 2020: 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 201–210.
18. Hagstroem, M.; Roggendorf, M.; Saleh, T.; Sharma, J. *A Smarter Way to Jump into Data Lakes. McKinsey Reports*; Technical Report; McKinsey & Company: New York City, NY, USA, 2017.
19. Hassan, I. Storage structures in the era of Big Data: From Data Warehouse to Lakehouse. *J. Theor. Appl. Inf. Technol.* **2024**, *102*, 6.
20. White, T. *Hadoop: The Definitive Guide*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2012.
21. Lämmel, R. Google's MapReduce programming model—Revisited. *Sci. Comput. Program.* **2008**, *70*, 1–30. [CrossRef]
22. Ghemawat, S.; Gobioff, H.; Leung, S.T. The Google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, New York, NY, USA, 19–22 October 2003; pp. 29–43.
23. Marz, N. How to Beat the CAP Theorem—Thoughts from the Red Planet—Thoughts from the Red Planet. Available online: http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html (accessed on 10 May 2024).
24. Munshi, A.A.; Mohamed, Y.A.R.I. Data lake lambda architecture for smart grids big data analytics. *IEEE Access* **2018**, *6*, 40463–40471. [CrossRef]
25. Kreps, J. Questioning the Lambda Architecture—O'Reilly. 2014. Available online: https://www.oreilly.com/radar/questioning-the-lambda-architecture/ (accessed on 15 May 2024).
26. Inmon, B. *Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump*; Technics Publications, LLC.: Basking Ridge, NY, USA, 2016.
27. Gorelik, A. *The Enterprise Big Data Lake*; O'Reilly Media: Sebastopol, CA, USA, 2016.
28. Zikopoulos, P.; DeRoos, D.; Bienko, C.; Buglio, R.; Andrews, M. *Big Data Beyond the Hype*, 1st ed.; McGraw-Hill Education: New York, NY, USA, 2015.
29. Madsen, M. *How to Build an Enterprise Data Lake: Important Considerations before Jumping*; Third Nature Inc.: San Mateo, CA, USA, 2015.
30. Ravat, F.; Zhao, Y. Data lakes: Trends and perspectives. In Proceedings of the Database and Expert Systems Applications: 30th International Conference, DEXA 2019 (Part I 30), Linz, Austria, 26–29 August 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 304–313.
31. Patel, P.; Wood, G.; Diaz, A. Data Lake Governance Best Practices. *Dzone Guide Big Data—Data Sci. Adv. Anal.* **2017**, *4*, 6–7.
32. Sharma, B. *Architecting Data Lakes—Data Management Architectures for Advanced Business Use Cases*, 2nd ed.; O'Reilly Media: Sebastopol, CA, USA, 2018.
33. Zhao, Y.; Megdiche, I.; Ravat, F. Data lake ingestion management. *arXiv* **2021**, arXiv:2107.02885.
34. Armbrust, M.; Ghodsi, A.; Xin, R.; Zaharia, M. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of the CIDR, Online, 21 January 2021; Volume 8, p. 28.
35. Giebler, C.; Gröger, C.; Hoos, E.; Eichler, R.; Schwarz, H.; Mitschang, B. The data lake architecture framework: A Foundation for Building a Comprehensive Data Lake Architecture. *Ges. für Inform. Bonn* **2021**. [CrossRef]
36. Hukkeri, T.S.; Kanoria, V.; Shetty, J. A study of enterprise data lake solutions. *Int. Res. J. Eng. Technol. (IRJET)* **2020**, *7*, 1924–1929.
37. Benjelloun, S.; El Aissi, M.E.M.; Lakhrissi, Y.; El Haj Ben Ali, S. Data lake architecture for smart fish farming data-driven strategy. *Appl. Syst. Innov.* **2023**, *6*, 8. [CrossRef]
38. Maini, E.; Venkateswarlu, B.; Gupta, A. Data lake-an optimum solution for storage andanalytics of big data in cardiovascular disease prediction system. *Int. J. Comput. Eng. Manag. (IJCEM)* **2018**, *21*, 33–39.
39. Raju, R.; Mital, R.; Finkelsztein, D. Data lake architecture for air traffic management. In Proceedings of the 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), London, UK, 23–27 September 2018; pp. 1–6.
40. Zhao, Y.; Megdiche, I.; Ravat, F.; Dang, V.n. A zone-based data lake architecture for IoT, small and big data. In Proceedings of the 25th International Database Engineering & Applications Symposium, New York, NY, USA, 14–16 July 2021; pp. 94–102.
41. Giebler, C.; Gröger, C.; Hoos, E.; Schwarz, H.; Mitschang, B. Leveraging the data lake: Current state and challenges. In Proceedings of the Big Data Analytics and Knowledge Discovery: 21st International Conference, DaWaK 2019 (Proceedings 21), Linz, Austria, 26–29 August 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 179–188.
42. Hlupić, T.; Oreščanin, D.; Ružak, D.; Baranović, M. An overview of current data lake architecture models. In Proceedings of the 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 23–27 May 2022; pp. 1082–1087.

43. Data Lake vs. Data Warehouse | Snowflake. Available online: https://www.snowflake.com/trending/data-lake-vs-data-warehouse/ (accessed on 8 May 2024).
44. Miloslavskaya, N.; Tolstoy, A. Application of big data, fast data, and data lake concepts to information security issues. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Vienna, Austria, 22–24 August 2016; pp. 148–153.
45. Simon, A.R. Data Lakes For Dummies | Wiley. Available online: https://www.wiley.com/en-ca/Data+Lakes+For+Dummies-p-9781119786184 (accessed on 2 May 2024).
46. O'Leary, D.E. Embedding AI and crowdsourcing in the big data lake. *IEEE Intell. Syst.* **2014**, *29*, 70–73. [CrossRef]
47. Gartner Says Beware of the Data Lake Fallacy. 2014. Available online: https://www.gartner.com/en/newsroom/press-releases/2014-07-28-gartner-says-beware-of-the-data-lake-fallacy (accessed on 17 May 2024).
48. Warren, J.; Marz, N. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*; Simon and Schuster: Toronto, ON, Canada, 2015.
49. Jarke, M.; Quix, C. On warehouses, lakes, and spaces: The changing role of conceptual modeling for data integration. *Concept. Model. Perspect.* **2017**, *2017*, 231–245.
50. Schneider, J.; Gröger, C.; Lutsch, A.; Schwarz, H.; Mitschang, B. The Lakehouse: State of the Art on Concepts and Technologies. *SN Comput. Sci.* **2024**, *5*, 1–39.
51. Thalpati, G.A. *Practical Lakehouse Architecture: Designing and Implementing Modern Data Platforms at Scale*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2024.
52. Daki, H.; El Hannani, A.; Ouahmane, H. Big-data architecture for electrical consumption forecasting in educational institutions buildings. In Proceedings of the 2nd International Conference on Networking, Information Systems & Security, New York, NY, USA, 27–29 March 2019; pp. 1–6.
53. Manogaran, G.; Lopez, D. Disease surveillance system for big climate data processing and dengue transmission. In *Climate Change and Environmental Concerns: Breakthroughs in Research and Practice*; IGI Global: Hershey, PA, USA, 2018; pp. 427–446.
54. Wang, W.; Fan, L.; Huang, P.; Li, H. A new data processing architecture for multi-scenario applications in aviation manufacturing. *IEEE Access* **2019**, *7*, 83637–83650. [CrossRef]
55. Augenstein, C.; Zschörnig, T.; Spangenberg, N.; Wehlitz, R.; Franczyk, B. A generic architectural framework for machine learning on data streams. In Proceedings of the Enterprise Information Systems: 21st International Conference, ICEIS 2019 (Revised Selected Papers 21), Heraklion, Crete, Greece, 3–5 May 2019; Springer: Berlin/Heidelberg, Germany, 2020; pp. 97–114.
56. Zschörnig, T.; Windolph, J.; Wehlitz, R.; Franczyk, B. A cloud-based Analytics-Platform for user-centric Internet of Things domains–Prototype and Performance Evaluation. In Proceedings of the 53rd Hawaii International Conference on System Sciences Maui, HI, USA, 7–10 January 2020.
57. Deligiannis, K.; Raftopoulou, P.; Tryfonopoulos, C.; Platis, N.; Vassilakis, C. Hydria: An online data lake for multi-faceted analytics in the cultural heritage domain. *Big Data Cogn. Comput.* **2020**, *4*, 7. [CrossRef]
58. Che, H.; Duan, Y. On the logical design of a prototypical data lake system for biological resources. *Front. Bioeng. Biotechnol.* **2020**, *8*, 553904. [CrossRef]
59. Li, Y.; Zhang, A.; Zhang, X.; Wu, Z. A data lake architecture for monitoring and diagnosis system of power grid. In Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference, New York, NY, USA, 21–23 December 2018; pp. 192–198.
60. Golec, D. Data lake architecture for a banking data model. *Entren.-Enterp. Res. Innov.* **2019**, *5*, 112–116. [CrossRef]
61. Sarramia, D.; Claude, A.; Ogereau, F.; Mezhoud, J.; Mailhot, G. CEBA: A data lake for data sharing and environmental monitoring. *Sensors* **2022**, *22*, 2733. [CrossRef] [PubMed]
62. Shih, W.C.; Yang, C.T.; Jiang, C.T.; Kristiani, E. Implementation and visualization of a netflow log data lake system for cyberattack detection using distributed deep learning. *J. Supercomput.* **2023**, *79*, 4983–5012. [CrossRef]
63. Youssef, H.Y.; Ashfaque, M.; Karunamurthy, J.V. Dewa r&d data lake: Big data platform for advanced energy data analytics. In Proceedings of the 2023 International Conference on IT Innovation and Knowledge Discovery (ITIKD), Manama, Bahrain, 8–9 March 2023; pp. 1–6.
64. Zagan, E.; Danubianu, M. Data Lake Architecture for Storing and Transforming Web Server Access Log Files. *IEEE Access* **2023**, *11*, 40916–40929. [CrossRef]
65. Manco, C.; Dolci, T.; Azzalini, F.; Barbierato, E.; Gribaudo, M.; Tanca, L. HEALER: A Data Lake Architecture for Healthcare. In Proceedings of the EDBT/ICDT Workshops, Ioannina, Greece, 28–31 March 2023.
66. Begoli, E.; Goethert, I.; Knight, K. A lakehouse architecture for the management and analysis of heterogeneous data for biomedical research and mega-biobanks. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 4643–4651.
67. Park, S.; Yang, C.S.; Kim, J. Design of Vessel Data Lakehouse with Big Data and AI Analysis Technology for Vessel Monitoring System. *Electronics* **2023**, *12*, 1943. [CrossRef]
68. Gopalan, R. *The Cloud Data Lake*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.
69. Zhang, J.; Ouda, A.; Abu-Rukba, R. Authentication and Key Agreement Protocol in Hybrid Edge–Fog–Cloud Computing Enhanced by 5G Networks. *Future Internet* **2024**, *16*, 209. [CrossRef]
70. Laurent, A.; Laurent, D.; Madera, C. *Data Lakes*; John Wiley & Sons: Hoboken, NJ, USA, 2020.

71. Davoudian, A.; Liu, M. Big data systems: A software engineering perspective. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–39. [CrossRef]
72. Marjani, M.; Nasaruddin, F.; Gani, A.; Karim, A.; Hashem, I.A.T.; Siddiqa, A.; Yaqoob, I. Big IoT data analytics: Architecture, opportunities, and open research challenges. *IEEE Access* **2017**, *5*, 5247–5261.
73. Bao, Z.; Liao-Liao, L.; Wu, Z.; Zhou, Y.; Fan, D.; Aibin, M.; Coady, Y. Delta Tensor: Efficient Vector and Tensor Storage in Delta Lake. *arXiv* **2024**, arXiv:2405.03708.
74. Pulivarthy, P. Enhancing Database Query Efficiency: AI-Driven NLP Integration in Oracle. *Trans. Latest Trends Artif. Intell.* **2023**, *4*, 4.
75. Hai, R.; Koutras, C.; Quix, C.; Jarke, M. Data lakes: A survey of functions and systems. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 12571–12590. [CrossRef]
76. Elouataoui, W. AI-Driven Frameworks for Enhancing Data Quality in Big Data Ecosystems: Error_Detection, Correction, and Metadata Integration. *arXiv* **2024**, arXiv:2405.03870.
77. Tae, K.H.; Roh, Y.; Oh, Y.H.; Kim, H.; Whang, S.E. Data cleaning for accurate, fair, and robust models: A big data-AI integration approach. In Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning, New York, NY, USA, 30 June 2019; pp. 1–4.