Data Architectures and their Technical Requirements (DATER)

Sayed Hoseini[a,c,*], Christoph Quix[a,b], Stefan Decker[b,c]

[a]*Niederrhein University of Applied Sciences, Krefeld, Germany*
[b]*Fraunhofer FIT, St. Augustin, Germany*
[c]*RWTH Aachen University, Aachen, Germany*

arXiv:2606.08811v1 [cs.DB] 7 Jun 2026

Abstract

Modern organizations generate and consume massive volumes of heterogeneous data at high speed. This requires a continuous development of new techniques for more efficient and reliable data management. Designing appropriate data architectures has therefore become a strategic necessity, as they shape how data is integrated, governed, and made available for analytics and decision-making. This paper introduces a conceptual framework—**D**ata **A**rchitectures and their **Te**chnical **R**equirements (DATER)—to systematically describe and evaluate data architectures based on technical requirements. Six modern architectures are examined: data warehouse, (semantic) data lake, data lakehouse, data fabric, and data mesh. Each is analyzed by historical context, defining features, and conformance to DATER dimensions. The study supports researchers and practitioners in navigating architectural paradigms, clarifying overlaps, and highlighting strengths, limitations, and use-case suitability.

Keywords: Data Architectures, Data Integration, Semantic Technologies, Data Mesh, Data Fabric, Data Lakehouse

1. Introduction

The exponential increase in the volume, velocity, and variety of data, fueled by digitalization, the Internet of Things, and Artificial Intelligence (AI), has made traditional, monolithic data systems insufficient. Traditional architectures such as data lakes and warehouses are increasingly complemented or replaced by emerging paradigms, because modern data ecosystems face growing demands for scalability, advanced analytics, interoperability, and decentralized governance. In this context, understanding and selecting the appropriate data architecture becomes a strategic imperative, as it directly affects the ability to derive value from data assets [1].

In recent years, the landscape of data management has undergone a profound transformation and data architectures play a fundamental role. In information technology, a data architecture describes various aspects of system design in order to provide guidance for the development of data management system [2]. The common goal is the effective management of data and the various systems or components in which it resides. Data architectures are most effective when they support the entire organization, enabling consistent data standardization and integration across all data-related activities [3]. Data architecture designs may thus vary for different levels within an organization, for particular domains, or based on its focuses such as infrastructure, applications, or data.

This paper gives an overview of the current predominant data architectures. Choosing the appropriate data architecture is critical, as substantial changes in enterprise data management typically require significant organizational effort. Gaining insights into the experiences, strategies, implementations, and deployment processes involved in transitioning to a modern architecture helps mitigate costly pitfalls and prevents data mess. We focus on the technical characteristics for particular data architectures including their historical development and then discuss main differences and similarities. Our descriptions of the different architectures are intended to support the development of reference architectures [4, 5, 6]. These reference architectures serve as blueprints for designing concrete solutions based on specific architecture types, offering standardized design guidelines [7] and fostering a shared vocabulary.

Furthermore, we propose a comprehensive framework for evaluating and classifying various data architectures referred to as **D**ata **A**rchitectures and their **T**echnical **R**equirements (DATER). DATER provides valuable insights into the evolving landscape of data systems, helping both researchers and practitioners make informed decisions about which architectural models best suit their specific needs and goals. While many real-world systems do not fully align with theoretical models, DATER enables a precise identification of which architectural components or capabilities differ. This helps bridge the gap between abstract design principles and practical implementations, guiding more effective system development and evaluation.

1.1. Review methodology and Contributions

This article follows a structured review methodology commonly applied in survey and review studies [8]. The review process is organized into three main stages: (i) planning the review by defining the scope and research question, (ii) conducting the literature search and screening process, and (iii) reporting and synthesizing the findings.

*Corresponding author.
Email address: sayed.hoseini@hsnr.de (Sayed Hoseini)

1.1.1. Scope and research question

This review focuses on modern data architectures and architectural frameworks proposed in both academic research and industry practice. The central research question motivating this review is:

> *What constitutes a structured framework for characterizing and differentiating modern data architectures, and how do existing architectures align with the technical requirements articulated by this framework?*

The scope is limited to data management within a single organization or enterprise context. Architectures primarily concerned with inter-organizational data sharing, such as data spaces [9], are excluded, as their primary focus lies on cross-organizational governance and collaboration rather than internal data architecture design. In particular, Gessler et al. [10] situate data spaces at the ecosystem level, not at the architecture level (see section 2).

Related architectural concepts that address specific aspects of data processing, such as Lambda and Kappa architectures [11], or extensions of classical data warehouses (e.g., Data Vaults [12, 13]), are considered only insofar as they contribute to the broader architectural landscape.

1.1.2. Search criteria

To ensure relevance and quality, the literature selection process was guided by explicit inclusion and exclusion criteria.

Inclusion criteria. Publications were included if they met the following conditions:

- Publication in English and full-text availability through digital libraries or publisher platforms;
- Relevance to data architecture, data management architectures, or architectural frameworks;
- Coverage of concepts such as data warehouses, data lakes, lakehouses, data fabrics, or comparable architectural paradigms;
- Recognized relevance in academic or industrial discourse, indicated by citation counts or adoption in practice.

Exclusion criteria. Publications were excluded if they:

- Focused primarily on applications rather than reference architectures or data management;
- Addressed specialized processing architectures (e.g., stream-processing-only systems) without architectural considerations for data organization and governance;
- Concentrated on cross-organizational data ecosystems, such as data spaces or data marketplaces, without a focus on internal enterprise architectures.

Although a wide range of additional architectural terms exists, such as data hubs, grids, federated or hybrid architectures, and event-driven or knowledge-graph-based architectures [12, 14, 11, 15, 16, 17], many of these can be traced back to a small set of core architectural patterns [18]. Accordingly, this review concentrates on those architectures that have gained substantial traction across both academia and industry.

1.1.3. Search method

The literature search was conducted primarily using Google Scholar, which forwarded to articles on IEEE Xplore, ACM Digital Library, ScienceDirect, Springer, and related publisher platforms. For each architectural paradigm, an initial set of approximately 20 well-cited journal articles or books was identified.

To extend and refine the literature set, citation snowballing was applied [8], including both backward searches through reference lists and forward searches using, for example, the "Cited by" functionality in Google Scholar.

In addition to academic literature, real-world systems referenced in the evaluation were identified using targeted web searches, technical documentation, and publicly available repositories on platforms such as GitHub.

1.1.4. Reporting and synthesis

The results of the review are reported by synthesizing architectural concepts into a unified framework termed DATER. The framework enables a structured comparison of architectures based on technical requirements and evaluation dimensions, and serves as the basis for mapping real-world systems to architectural models in subsequent sections. The screened literature and synthesized technical requirements form the basis for deriving the evaluation criteria presented in section 4.

Selection of evaluated systems. The systems included in the evaluation were selected as representative implementations of the architectural paradigms discussed in this review rather than as an exhaustive survey of available platforms. In addition to well-established reference systems, we deliberately included systems whose architectural classification is ambiguous or hybrid in nature.

This choice allows us to demonstrate the analytical value of DATER in cases where existing architectural labels are insufficient or contested. By applying DATER to such systems, it becomes possible to pinpoint which architectural layers and technical requirements are fulfilled, partially addressed, or blurred across paradigms.

The number of evaluated systems was therefore intentionally limited, as additional systems with similar architectural characteristics would not provide further insight beyond implementation-specific variations.

Contributions. In summary, this article makes the following contributions:

1. We first define our terminology to clarify key terms such as data ecosystem, data architecture, and data platform (section 2).

2. The core of the DATER framework is the characterization, definition, and identification of technical requirements for six data architectures (section 3).

3. From this, we extract nine evaluation dimensions to assess data architectures. Moreover, a comparative evaluation of real-world systems is conducted by mapping them against the DATER framework, providing insights into their alignment with various architectural models (section 4).

2. Terminology

Terms such as architecture, framework, ecosystem, platform and infrastructure are strongly related and often used interchangeably, causing confusion and misinterpretation [19]. For example, Geisler et al. [20] introduce and define the term data ecosystem, but point to *Ontario* [21] and *Constance* [22] as concrete implementations, which are titled as (semantic) data lake by the authors. While DAMA-DMBOK [23] distinguishes between data architecture, framework, and strategy, their descriptions occasionally overlap, reflecting the inherent ambiguity of terminology in this domain. Figure 1 illustrates the introduced terminology in a hierarchy of data strategy, ecosystems, governance frameworks, architectures, and technical components. Each concentric ring moves from abstract, organizational-level concerns (such as strategy and governance) toward concrete, technical implementations, e.g., data architectures and individual components for data processing. The inner layers in the core of the figure highlight architectural aspects, grouped first by the selected architectures in this article, then by central technical functionality, and finally with a selection of concrete technologies for individual components.

The concepts data strategy, data ecosystems, data governance framework and data architecture describe different perspectives of data management. In the following, we aim to give precise definitions, while being aware that others may use the terms differently depending on the context:

Data strategy refers to a high-level plan for how an organization will collect, manage, and use data to achieve goals, which must be directly aligned with the overall business strategy [23].

Data ecosystem, as defined by [24], refers to "*set of networks composed of autonomous actors, which consume, produce, or provide data or other related resources*". It describes the information management landscape of an organization on a conceptual level, i.e., how information is used or produced in business processes as well as how it is managed in information systems. It might include relationships to external organizations, e.g., for information exchange along the supply chain.

A **data governance framework** defines best practices, standardized roles, processes, technologies, and guiding principles for managing data assets across their lifecycle, in alignment with the data strategy of the organization. It defines the governance principles of the data ecosystem of an organization.

Data architecture refers to the technical design of multiple data-related components and their interactions. It should be developed according to the guidelines and goals defined in the data strategy and governance framework.

Data platform, in this article, refers to a concrete implementation of a data architecture.

To clearly distinguish each term, we address potential points of confusion by discussing areas where their meanings may overlap.

Data ecosystems are usually composed of multiple data architectures, i.e., an organization might have data warehouses as well as data lakes. The abstraction at data ecosystem level should not only focus on technical aspects and tools, it should also include people and policies. In this sense, the term ecosystem emphasizes the socio-technical dimension, focusing on actors, governance structures, interoperability standards, and shared infrastructure.

In this sense, we illustrated data governance frameworks as complementary to data ecosystems. Data ecosystems rather focus on actors, components, and their relationships, whereas a data governance framework aims at policies, standards, responsibilities, and guidelines.

The core of Figure 1 illustrates common functions and components of data architectures. For example, we view database management systems, data management systems, and data catalogs as individual components of data architectures. We distinguish between *Database Management Systems* (DBMSs) and the broader category of *Data Management Systems* (DMSs). A DBMS is a software system that supports the definition, querying, and manipulation of structured data, typically offering declarative query languages (e.g., SQL), transaction management, and consistency guarantees [25]. DBMSs, but also DMSs, are central components of many data architectures and are usually tailored to a specific data model, which defines their query capabilities.

While traditional relational DBMSs remain dominant, recent years have seen the rise of specialized systems:

- NoSQL DBMSs, categorized into document-oriented (e.g., MongoDB), column-oriented (e.g., Cassandra, HBase), graph-oriented (e.g., Neo4j), and key–value stores [26],
- Vector DBMSs (e.g., QDrant) for similarity search and AI workloads,
- Time-series DBMSs (e.g., InfluxDB) for stream and sensor data, and
- Search-oriented DBMSs (e.g., Elasticsearch) for full-text indexing and distributed search.

In contrast, a *Data Management System* (DMS) provides basic mechanisms for storing, accessing, and organizing data, often in the form of files or objects, but lacks advanced DBMS features such as declarative query languages, fine-grained transaction support, or schema enforcement. DMSs prioritize scalability and availability over strict structure, and typically provide only basic access methods such as file download. Examples include Apache Hadoop and Ceph. Another important component of a data architecture is a *Metadata Management System* or *Data Catalog*, which is a system used to manage and organize metadata. Data catalogs provide capabilities for tracking data lineage, enabling data search and discovery, and ensuring

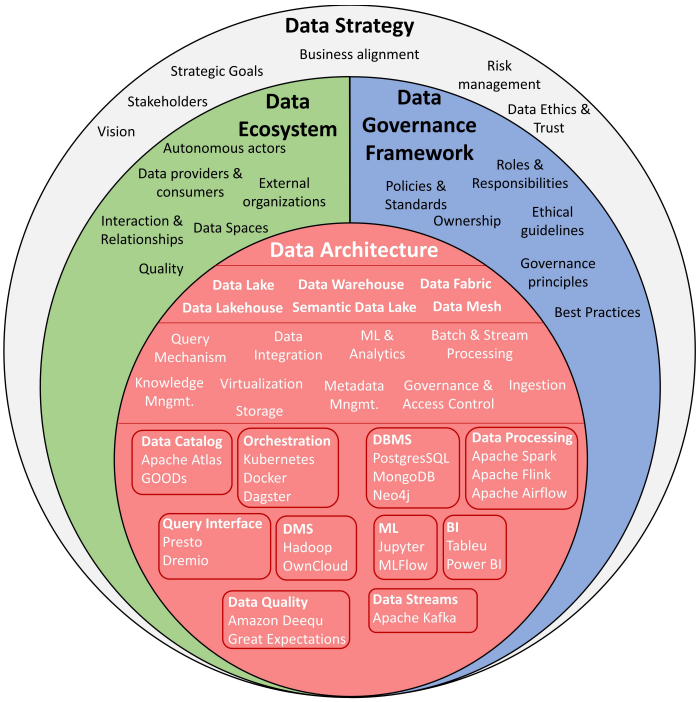

Figure 1: Overview of terminology, abstractions and components for data architectures

that data is well-documented and accessible for various users. They play a crucial role in data governance and help organizations maintain compliance, ensure data quality, and enhance the overall usability of data.

3. Data Architectures

With the terminology, we now introduce and analyze six prominent data architectures: data warehouse (DW), data lake (DL), semantic data lake (SDL), data lakehouse (DLH), data fabric (DF), and data mesh (DM). Each architecture is presented with a brief discussion of its historical motivation, followed by a definition, characteristic features, formal requirements, and an architecture diagram.

Figure 2 illustrates the general historical development and conceptual relationships among these architectures. The evolution begins with data warehouses, which emerged in the 1990s to support analytical reporting and business intelligence through centralized, materialized, and integrated repositories [27]. An important precursor was the federated database system of the 1980s, which offered virtual integration of heterogeneous databases without physically centralizing the data [4]. The Semantic Web, introduced around 2000 [28], laid the groundwork for knowledge representation and semantic integration. Around the same time, the term *big data* was introduced [29], emphasizing challenges in volume, velocity, and variety.

The 2000s and early 2010s saw a range of technological shifts: the rise of *NoSQL* database systems to address schema flexibility and scalability [30], the emergence of cloud storage platforms such as Amazon S3 [31], and the increasing relevance of data stream processing for real-time analytics [32]. The machine learning (ML) boom, accelerated by advances in deep learning from 2012 onward [33, 34, 35], further increased demand for architectures capable of handling heterogeneous, high-volume, and rapidly changing data [36]. In response, data lakes emerged around 2010-2015 [37], offering scalable storage with schema-on-read and ELT paradigms. These were later extended into semantic data lakes, which incorporate ontologies and metadata for improved data understanding and integration [38].

As data became increasingly distributed across departments, organizations, and jurisdictions, new architectural paradigms emerged to address governance, sovereignty, and scalability. Data mesh introduces domain ownership and a "data-as-a-product" philosophy to promote scalability and accountability within organizations [39]. Around 2019, data fabric was proposed [40] as a metadata-driven integration layer that leverages AI for governance and enables self-service access across heterogeneous environments. More recently, the data lakehouse concept has gained traction [41], combining the reliability and ACID guarantees of data warehouses with the flexibility of data lakes to support unified batch and real-time analytics.

Overall, the figure highlights a clear evolution from centralized and structured systems toward decentralized, federated, and semantically enriched architectures. This trend reflects the increasing heterogeneity of data sources and the distributed nature of modern enterprises. Nevertheless, data warehouses, lakes, and lakehouses remain central in practice due to their maturity and strong theoretical foundations; especially in scenarios where decentralization is not a primary requirement and user groups are relatively homogeneous [39].

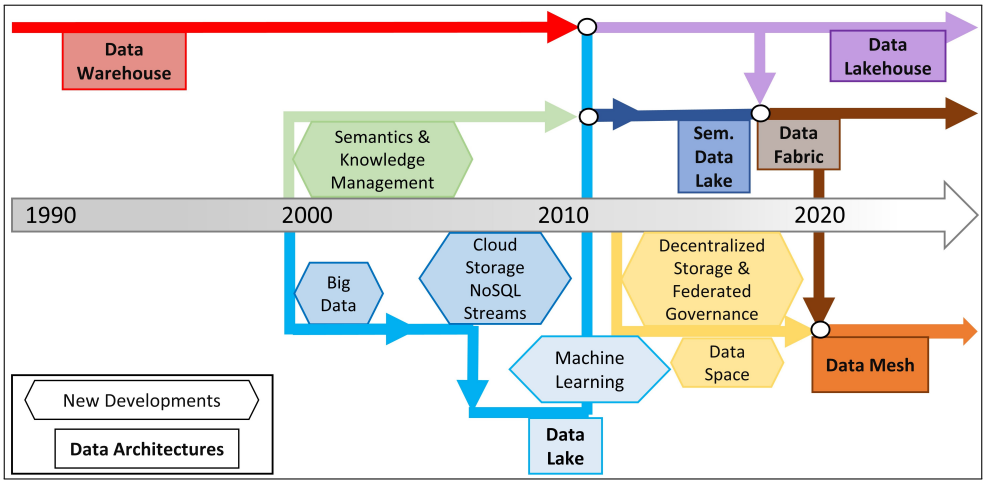

Figure 2: Evolution of the various discussed data architectures

3.1. Data Warehouse

In the 1990s, data warehouses [27] emerged as a standard solution for enterprise-wide analytical data management. They enabled organizations to run overnight ETL processes (Extract-Transform-Load) that extracted data from heterogeneous sources, transformed it to meet analytical requirements, and loaded it into a centralized, materialized repository. This architecture supported business intelligence by providing a consistent and integrated view of enterprise data.

Traditionally based on relational database technology, data warehouses are the most established form of analytical data architecture [27] and remain widely used in industry. They have evolved to support cloud-native implementations that offer greater scalability and lower operational costs. For instance, *Apache Hive* [42] introduced data warehouse functionality for big data by enabling SQL queries on top of *Hadoop*, while *Snowflake* [43] exemplifies a commercial, cloud-native data warehouse. Despite these innovations, data warehouses primarily support predefined analytical queries for reporting purposes and are less suited for exploratory or unstructured analytics such as text or web mining [44, 45].

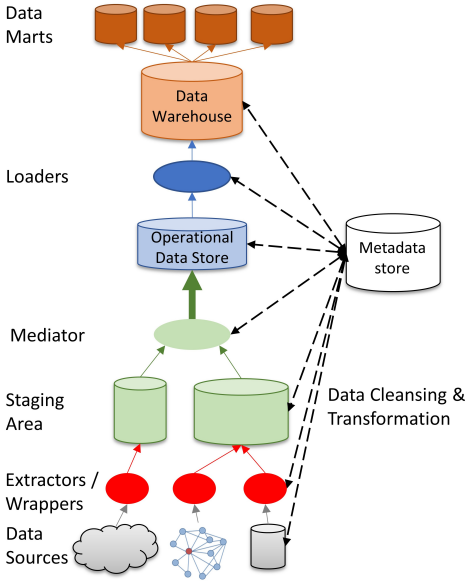

Figure 3: Data warehouse architecture based on [47, 50]

3.1.1. Characteristics

Data warehouses are designed to manage structured data and often rely on multi-dimensional data models for complex analytical tasks. They typically ensure atomicity, consistency, isolation, and durability (ACID) properties and offer advanced capabilities such as time travel, data governance, and zero-copy cloning [41].

For the definition we rely on Inmon [27]:

Definition 1. [**Data Warehouse**] "*A data warehouse is a subject oriented, integrated, non-volatile, and time variant*" data architecture *"in support of management's decisions."* [27]

Subject-oriented refers to the purposeful organization of data to support business analysis and decision-making. *Integration* is the central aim: although data is collected from diverse systems, the warehouse presents a unified view to the user. *Non-volatility* indicates that data is not regularly updated by operational systems but remains stable to support long-term analysis. *Time-variance* reflects the inclusion of historical data, enabling trend analysis and retrospective insights.

In practice, the term "data warehouse" may also refer to centralized repositories that include little or no historical data. However, the core idea remains that it provides business users and decision-makers with consistent, reliable access to analytical data [46].

The architecture, illustrated in Figure 3 [47, 48, 36], revolves around ETL processes that consolidate data from various sources. While the core storage is generally relational, the staging area may involve other systems, including NoSQL stores [49]. To support performance and modular access, data marts are employed as thematic subsets of the warehouse, optimized for *Online Analytical Processing* (OLAP) [44]. Data sources

are independent systems from which data is extracted. The staging area serves as a buffer for cleansing and transforming data before integration. A mediator integrates this harmonized data into a consistent, unified store. The operational data store holds historical data independently of specific use cases. The data warehouse acts as the central, structured repository optimized for analysis. Data marts are thematic subsets designed to support modular and secure data access. Finally, the metadata system manages schema and lineage information, enabling effective data governance and understanding.

3.1.2. Requirements

The following technical requirements capture the essential features of a data warehouse ([27, 44, 50]):

DW1 — Schema-on-write: data must conform to a predefined schema before it is written and stored. Hence, an ETL process consolidating data from heterogeneous sources is required to transform the data into the desired target schema.

DW2 — Structured data model: data is organized into a structured schema following a relational or multidimensional model.

DW3 — Consistency guarantees: provide means for checking and enforcing the consistency of structure and content across data collections.

DW4 — Optimize for read-heavy workloads: optimize query performance, making it ideal for analytical and reporting tasks.

DW5 — Time-variant data storage: include historical data, supporting trends and time-based analysis by retaining snapshots of data over time.

DW6 — Query language: offer a declarative, structured query language like SQL.

DW7 — OLAP: enable Online Analytical Processing (OLAP) by supporting multidimensional queries for business intelligence.

3.2. Data Lake

Since their emergence, numerous proposals have addressed data lake architectures, in particular, challenges, concepts, components, and implementations [51, 52, 53, 22, 54, 55, 56, 57, 58]. In addition, major IT companies have introduced commercial tools for data lakes [59, 60, 61]. In the following, we focus on widely recognized concepts and representative architectural patterns discussed in current academic literature or industrial implementations [62, 63, 64].

In 2010, James Dixon [37] introduced the term data lake as a solution for managing raw data from a single source while accommodating diverse user needs, because "*data is of a scale or daily volume such that it won't fit technically and/or economically into a relational DBMS*" [37]. This concept stood in stark contrast to the so-far dominant data warehouse systems. By preserving data in its raw format, data lakes bypass or delay the costly preprocessing steps. Dixon motivates the choice for the word lake with an analogy: "*If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state.* " [37].

In 2014, Gartner [65] raised several criticisms of data lakes, warning that the indiscriminate ingestion of disparate data could lead to a so-called data swamp, making the data unusable without proper metadata management and governance [51]. Further, concerns included insufficient attention to data security and privacy oversight. In response to these critiques, Dixon revisited the data lake concept [66].

With Hadoop at the core, a key requirement was that a data lake should offer low-cost, easily accessible storage that supports a **schema-on-read** approach, meaning the metadata can be populated over time and transformations to apply the schema are only performed once required. This approach allows a data lake to automatically extract metadata from raw data and identify patterns, with users contributing additional descriptive information, such as domain-specific knowledge and attribute linkages. Hence, the idea is that the interaction between users and the data lake continually enhances (meta)data quality and value.

3.2.1. Characteristics

The classical ETL approach was to limited to handle the requirements at the time, because transforming data to fixed schemas and rigid data models leads to limited flexibility in data management. Additionally, some information might be lost during the process, as it might not fit into the target schema and scaling often required a substantial financial investment, both in terms of initial hard- & software costs and ongoing maintenance [62].

In contrast, data lakes load **raw data** from multiple sources into a unified management system with a heterogeneous storage layer. They provide flexible data access, automated and efficient data preparation workflows, and a series of data pipelines for a wide range of analytical applications of which end-to-end machine learning workflows become increasingly important [63, 68]. Data is transformed only if required for a specific application or query (schema-on-read). Thus, data lakes apply an Extract-Load-Transform (**ELT**) approach. As data heterogeneity and the number of sources increase, data lakes follow a data-first, schema-on-read approach, reflecting the shift from frequent, schema-driven access toward more flexible, on-demand usage [69].

Query processing needs to deal with heterogeneous data sources in an on-the-fly manner [63], as data is not transformed into a uniform format before it is loaded.

Further, data lakes are defined by their ability to scale to massive volumes, all while offering **low-cost storage** solutions typically based on Hadoop, enabling organizations to store petabytes of data [62].

Apache Spark is a system frequently used for the implementation because it can process queries over heterogeneous data sources. For example *Constance* [22] implements a polystore

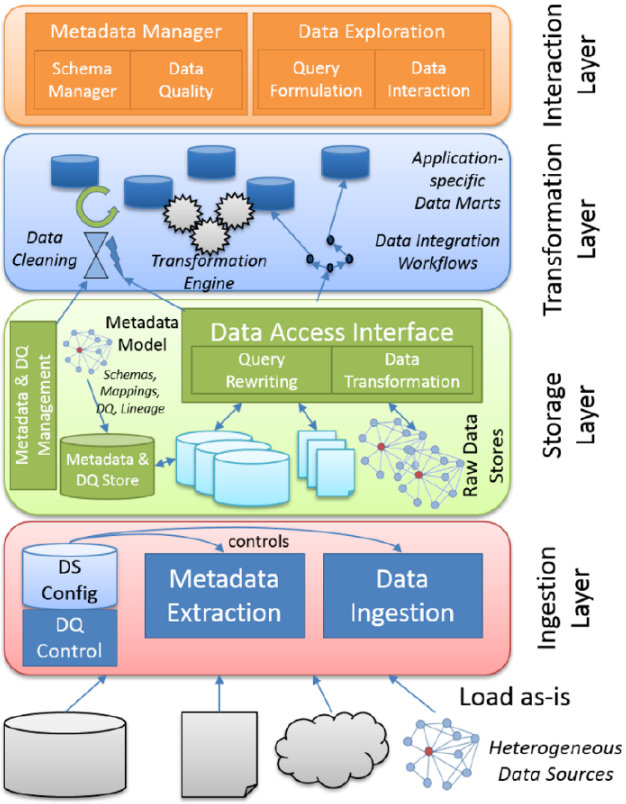

Figure 4: Four-layered data lake architecture [67]

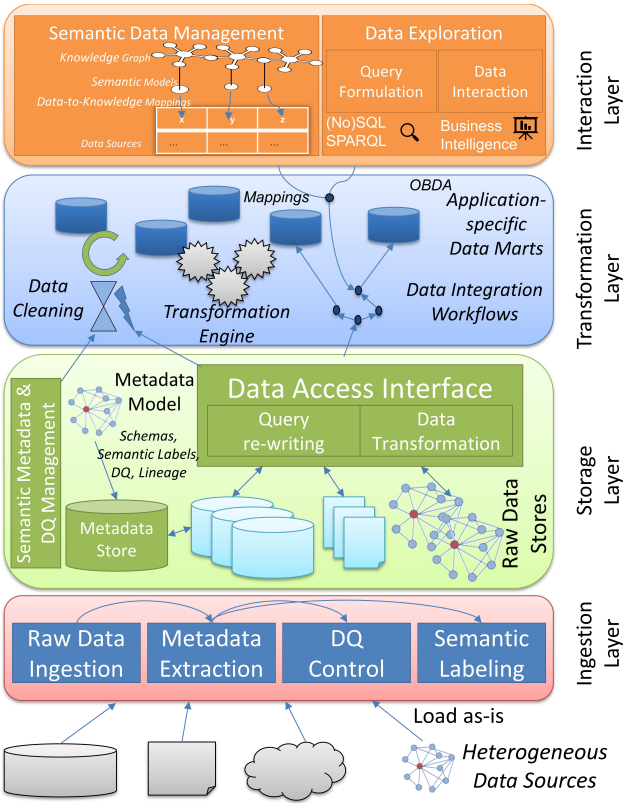

Figure 5: Semantic data lake architecture [38]

by storing diverse raw data according to its original format: relational (e.g., *MySQL*), document-based (e.g., *MongoDB*), and graph databases (e.g., *Neo4j*).

A key feature is the use of metadata to manage, organize, and analyze the stored data, because otherwise the data is hardly usable as their structure and semantics are not known. Hence, **metadata extraction** or **metadata enrichment**, the process of automatically discovering metadata information of a dataset, is essential for accessing datasets in a later phase. This includes all types of metadata, the more the better: structural, but also descriptive information as well as relationships to other datasets. Metadata models answer the question of which information to collect and how to structure and organize it and there numerous proposals in recent years [68, 53, 70, 71, 72, 73].

Additionally, data lakes aim to enable data exploration and discovery, allowing analysts to navigate and examine heterogeneous data sources without needing predefined schemas or structures and this is a complex challenge and subject to ongoing research [74, 15, 75, 21, 76]. Rencently, the increasing need for supporting machine learning processes inside a data lake system is emphasized [63, 64].

We adopt our definition for data lakes from [38] and added further technical requirements:

Definition 2. [**Data lake**] A data lake is a data architecture that manages heterogeneous data sets in their original structures. It should provide functions to extract data and metadata from heterogeneous sources, manage the data efficiently in heterogeneous storage systems, and query and transform the data in a scalable way. Furthermore, a uniform query processing mechanism is necessary, enabling users to query data seamlessly across diverse formats and sources without requiring extensive pre-processing or schema transformations [51, 63].

3.2.2. *Requirements*

DL1 — Schema-on-read: allow raw data of any format to be stored. Schema definitions are only required on access to the data.

DL2 — Data ingestion at scale: support diverse data ingestion methods, such as batch processing, streaming, and real-time data flow for arbitrary heterogeneous data sources.

DL3 — Scalability and cost-effective storage: handle vast amounts of data with elasticity to expand or contract storage based on needs for cost efficiency.

DL4 — Data governance and metadata management: provide a robust data catalog with automated metadata extraction, searchable indexes, tags, and descriptions for datasets ensuring data quality, lineage for data discovery to locate, understand, and utilize data independently of analysis.

DL5 — Unified query mechanism: enable querying of heterogeneous data sources using a unified query interface

allowing users to perform federated queries across structured and unstructured data with on-demand data transformations.

Azzabi et al. [62] outline how the initial *mono-zone architecture* had notable limitations and hence data pond and zone-based architectures were introduced. Zone-based architectures include transient landing, raw, trusted, and refined zones, along with a sandbox zone for exploratory analysis. Each zone has a specific function based on its refinement level, from raw data ingestion and preliminary processing to serving compliant, quality-checked data to end-users [77, 78, 79, 80, 81, 82]. A *four-layered data lake architecture* was proposed in [67] and is shown in fig. 4. It resembles some ideas of zone-based architectures, such as a raw data 'zone' at the storage layer, and refined data in data marts at the transformation layer. In addition, it emphasizes the role of metadata and quality management on each of the layers.

The *Ingestion Layer* loads data "as-is" into the data lake while maintaining its original format and structure thereby facilitates the acquisition of raw, heterogeneous data from various data sources, including databases, files, and streaming platforms. It is responsible for metadata extraction to provide insights into the structure, quality, and other properties of the incoming data.

The *Storage Layer* provides a centralized repository for storing raw data and metadata retaining the original format of ingested data. It manages metadata and quality information, enabling efficient querying and processing and ensures that raw data remains immutable, supporting traceability and reproducibility.

The *Transformation Layer* is responsible for the integration and preparation of data transforming raw data into refined datasets to meet analytical requirements, it facilitates query rewriting and data transformation tasks, such as data cleaning, enabling seamless data retrieval and manipulation.

The *Interaction Layer* focuses on enhancing the accessibility and usability for end-users acting as the user-facing interface. It comes with some form of metadata manager that handles schema mapping, data quality assessment, and metadata management as well as data exploration to enables users to explore datasets, query the data lake, and interact with application-specific workflows.

3.3. Semantic Data Lake

Various strategies for the use of semantic data management have been introduced [57, 83, 38, 36], which underscore the necessity of integrating big data with semantic web technologies into *semantic data lakes*. A semantic data lake facilitates the derivation semantic models from raw datasets, thus providing a more conceptual data description that includes concepts and their interrelations, and linking data sets to their relevant concepts [84]. Recently, with the growing demand for knowledge management, knowledge graphs have also been integrated into data lake architectures [85, 86].

At the company *Bosch*, the potential of semantic technologies for data management has already been exploited at scale in production [87]. Dibowski et al. [85, 84] describe a metadata model represented as an ontology based on W3C recommendations for the data catalog management and provenance control of a semantic data lake. Ingested data assets are **enriched with semantics** by aligning, annotating, and enriching the input data with semantics. The development of knowledge graphs required an immense initial effort by the company but promised a long-term solution for a series of sophisticated challenges. In addition, they utilize a new concept called ontology-driven, self-adaptive frontends, in which user interfaces for data catalogs can dynamically render information from a knowledge graph. A similar concept was presented in [88]. This is useful because changes in the underlying ontology do not require any changes at the frontend. The *ESKAPE* system from 2017 [89] can be viewed as a very first prototype in this direction and the term was coined by Pomp et al. [90] and Mami et al. [91] for the first time in 2018.

3.3.1. Characteristics

We presented a thorough review of the field in [38] from which we derive the following definition and requirements:

Definition 3. [**Semantic data lake**] Semantic data lakes are a specific form of traditional data lakes that extend the capabilities through a *semantic layer* that enriches and connects the stored data *semantically*. The *semantic layer* equips data sets in the lake with connections from the data set's metadata to conceptual/logical models, which encapsulate knowledge potentially external to the content of the data, such as domain knowledge.

3.3.2. Requirements

The following, together with the requirements for a traditional data lake (see section 3.2), confine the semantic data lake.

SDL1 — Creation of semantic labels: enable the mapping of data attributes and other schema elements to elements of a KG (might be internal or external, domain-specific).

SDL2 — Semantic relationships: represent any form of hierarchical, generic, or pre-defined semantic relationships (semantic connections between data sets, e.g., for provenance or governance).

SDL3 — Metamodel as ontology: the metamodel of the lake itself is modeled as ontology using semantic web technologies.

SDL4 — Semantics-based data access: the ability to query and access data using semantic-based query languages like SPARQL, i.e., abstracting from physical storage and allowing to interact with the data conceptually using techniques like ontology-based data access [92].

In [38], we introduce a semantic data lake architecture (see fig. 5) based on the four-layered architecture. Dibowski et al. [85] describe a semantic data lake as a "*specific form of data lakes in which a semantic layer on top enriches and connects the data semantically*". This definition suggests to extend the

four-layered architecture by a fifth layer. In our view, proper semantic data management is only possible if it is an integral part of the data architecture. Therefore, incorporating semantics demands adjustment across the entire functionality, starting possibly already at the ingestion, where users (or AI-based technologies) annotate a dataset with semantic labels, before it is fully available inside the lake for accessing. Thus, each layer of the architecture has been extended with enhanced metadata functions with semantic capabilities. For instance, semantic labeling in the ingestion layer assigns semantic labels to metadata elements. Data quality management benefits from an expanded semantic metadata repository managed within the storage layer, which supports functions such as ontology-based data access (OBDA) [38], aiding in data usage and interpretation. Consequently, the interaction layer incorporates additional features, such as exploration of the knowledge graph and the semantic model, semantic query formulation using languages like SPARQL, and tools for refining semantic mappings and models.

3.4. Data Lakehouse

Although data warehouses and data lakes serve different analytical purposes, organizations frequently require features of both worlds simultaneously [3]. Especially the necessity of generating business reports and crafting ML models in a single platform gave rise to the *data lakehouse*. Armbrust et al. [41] recognize that many organizations host an expensive two-tier architecture which is highly complex for users. Data is first ingested into lakes, and then again fed via an ETL process into data warehouses, creating complexity, delays, and new failure modes. Consistency is one of the challenges, i.e., keeping the data lake and warehouse in a consistent state is difficult and costly. Data is stale, new data is loaded to the data warehouse only with a delay due to the ETL process. Finally, both architectures provide limited support for advanced analytics.

The typical data warehouse supports well-designed business intelligence (BI) queries, which are known in advance. The multidimensional data model of data warehouse is optimized to support these queries. However, ad-hoc queries to extract large of amounts of data for a new purpose, e.g., training data for ML models, are less efficient in a data warehouse context. Data lakes, on the other hand, lack the rich and matured features from data warehouses, such as ACID transactions, data versioning and indexing [41]. In response, Databricks, the company founded by the creators of *Apache Spark*, introduced the lakehouse architecture to combine the best features of both systems. Building on their expertise with Spark's scalable data processing capabilities, they introduced *Delta Lake* in 2019 [93] as an open-source storage layer that brings ACID transactions, schema enforcement, and data versioning to data lakes.

3.4.1. Characteristics

Since key characteristics of data warehouses and data lakes often conflict (particularly in terms of data access, data independence, and storage formats) combining them typically requires compromises on both sides. For instance, a data warehouse that supports direct access to storage, as in data lakes, must sacrifice data independence and adopt open file formats. Conversely, a data lake can introduce warehouse-like management features only by restricting users' flexibility in storing and accessing data, enforcing structured protocols for read and write operations [94]. The data lakehouse architecture merges the flexibility of data lakes, which support the large-scale storage of various data formats, with the transactional integrity of data warehouses by incorporating layered components.

Characteristic for a data lakehouse is a **transactional layer**, which ensures data integrity through ACID properties (Atomicity, Consistency, Isolation, Durability). This layer organizes the data and manages metadata, enforcing schema, supporting data versioning, and ensuring data lineage, which enhances overall data quality. Here, technologies like Delta Lake, Apache Hudi, and Iceberg used together with Spark on top of scalable and cloud-native storages like HDFS form the core [95]. Thus, the lakehouse has a **relational format** as its final structure, on top of heterogeneous sources, as most analytical and visualization tools are optimized for relational data, enabling faster analysis. Schneider et al. [95] conduct a thorough review on the data lakehouse and find four common definitions in current literature and argue that each one of them has certain shortcomings. Hence, they propose the following and derive technical requirements from it. The following are direct quotes from [95]:

Definition 4. [**Data Lakehouse**] *"A data lakehouse is a"* data architecture *"that leverages the same storage type and data format for reporting and OLAP, as for data mining and machine learning, as well as streaming workloads"*.

3.4.2. Requirements

DLH1 — ***"Same storage type and data format****: all (meta)data is stored on a single type of storage (no polystore), using a uniform open data format. Hence, a data lakehouse must have an inherently low heterogeneity in terms of the utilized technologies and data formats.*

DLH2 — ***Relational data collections****: this strong requirement comes from the need to support reporting and OLAP workloads, which require a higher degree of structure than would be provided by other types of abstractions, such as graphs or documents.*

DLH3 — ***Query language****: offer a declarative, structured data query language for the data collections.*

DLH4 — ***ACID guarantees****: provide means for checking and enforcing the consistency of structure and content across data collections, and ensure isolation and atomicity for all operations that modify or access the data concurrently.*

DLH5 — ***Direct read access****: provide unmediated, direct, and random read access to all (meta)data stored on the underlying storage system without needing to export and transform it first, and leverage open, standardized file formats.*

DLH6 — ***Unified batch and stream processing****: support processing in large-scale batches as well as at a high rate and in real-time."*

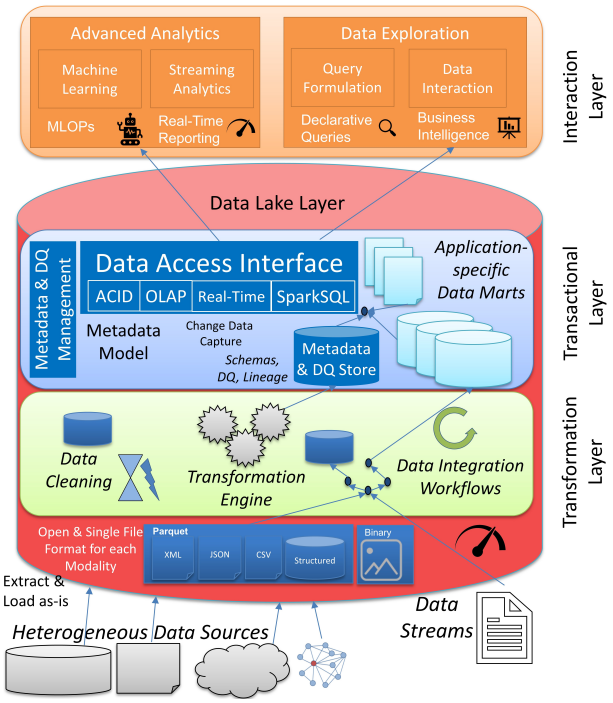

Figure 6: Data Lakehouse architecture inspired by [67, 41]

We derive a data lakehouse architecture (see fig. 6) from the four-layered data lake architecture and the findings of Harby et al. [96] and Azzabi et al. [62]. In the figure, the ingestion layer is replaced by a composite *Data Lake Layer*. This layer is composed of an ingestion layer, to load data and extract metadata, as well as a storage layer. What differs is that the storage is composed of single scalable storage technology, for example *HDFS* or *Amazon S3*. This foundational layer serves as the primary storage for raw, structured, semi-structured, and unstructured data. It enables scalable, cost-effective data storage and accommodates diverse data sources, serving as a central repository for all ingested data. Further, the layer utilizes a single open data format (such as Parquet) for each modality.

The *Transformation Layer* is responsible for the ETL pipelines and thus dedicated to extracting data from various sources, transforming it into structured formats, and loading it for analysis. It standardizes and cleanses data to make it ready for analytics, enabling compatibility and consistency across data sources and preparing it for data access. *Transactional Layer* supports data operations and guarantees ACID compliance, enabling reliable and transactional processing, often needed for complex analytical queries (OLAP) supporting accurate and real-time analytical processing. The transformation as well the transactional layer are embedded into the data lake layer to emphasize the vision that the lakehouse is a uniform system, and does not resemble a two-tier architecture composed of an individual data lake and data warehouse and thus no data replication or pipelines for transferring between the platforms are necessary. Finally, the *Interaction Layer* serves as the user interface ensuring that insights can be easily generated, accessed and utilized by business users and data scientists.

3.5. Data Fabric

The concept of data fabrics was coined by NetApp in a white paper from 2016 as a "vision for data management [...] that seamlessly connects different clouds, whether they are private, public, or hybrid environments" [97]. Back then, the term meant a combination of different data storage and integration techniques, however, without details about a concrete architecture.

In 2019, Gartner defines a data fabric as a design concept for attaining reusable and augmented data integration services, data pipelines and semantics for flexible and integrated data delivery [98]. Hence, a data fabric is a technical architecture that brings together heterogeneous data that spans across multiple data sources; it allows organizations to monitor and manage data regardless of the location, considering appropriate data governance and data cataloging [99, 100, 101]. Then in 2021, Gartner [102] redefines a data fabric as a "design concept that serves as an integrated layer (fabric) of data and connecting processes". In their data fabric architecture, Gartner emphasizes key characteristics, such as the use of AI for metadata management and data cataloging as well as knowledge graphs and ontologies to connect and enrich data sources with business related concepts. They introduce the notion of **"passive vs active metadata"**, i.e., data models, schema definitions, glossary, various logs are considered as passive metadata, whereas the continuous analysis of data usage with are described as active metadata describing actual experiences. The conversion of passive metadata into active metadata is interpreted as automating the collection and management of metadata [98]. By now, major vendors in data science have embraced the term; both vendors of semantic technology (e.g., Franz Inc., Ontotext, Pool Party, Stardog, and Top Quadrant) as well as more traditional vendors such as Microsoft, Oracle and Tibco [98].

3.5.1. Characteristics

At present, the particular mechanisms, standards, and technologies for the implementation of a data fabric architecture still have at be clarified as there is not a reference architeture or system [97, 103, 101]. For example, some authors advocate for the presence of **data virtualization**, i.e., a software architecture that allows applications and end users to access data from multiple sources as if it were stored in a single location, as a critical component of a data fabric [104, 105, 100]. In fact, the term "fabric" metaphorically represents the interwoven threads/fibers of a textile, signifying how different data sources, technologies, and workflows can be interconnected into a cohesive whole. As noted by *CData Software* [106]: "*If we take the metaphor to its conclusion, data virtualization can weave the threads (data) to make the finished cloth (data fabric).*" Others see data virtualization as optional [107] or do not mention it at all [108]. In our requirements definition below, we consider data virtualization as a key requirement for data fabrics. Data fabrics should be used as an additional integration layer on top of other existing data platforms.

Similarly, various proposals include the use of knowledge graphs, ontologies and semantic web technologies explicitly

[98] and others do not mention them at all [109]. DeBellis et al. [98] provide strong arguments in favor of using semantic web technologies because they ensure reuse and interoperability of data.

Modeling metadata for a data fabric requires a complex, highly interconnected model that typically spans multiple organizational, geographic, and business boundaries. Obviously, relational data models are too rigid to easily store this type of data. Knowledge graphs are more suitable to represent such information because they are more flexible, heterogeneous, formal as well as fast and scalable [98]. Martinez-Casanueva et al. [110] present *CANDIL*, a data architecture to convert heterogeneous data into an RDF-based representation using mappings. They call their platform a data fabric; however, besides the use of semantic data management techniques, we do not recognize other technical features in their approach. Therefore, we consider it as an ontology-based data integration pipeline.

A dominant theme in the data fabric paradigm is the idea of the **polystore**, i.e., combining heterogeneous storage databases, potentially hosted in the cloud [103, 111, 108, 112, 113]. Similarly, data storage is not meant to be centralized to a single location for access, but decentralized and distributed [99, 107]. Data exploration and access is mediated by the fabric through centralizing metadata only [109, 100].

To facilitate this type of accessing, discovering, and understanding of data, but also for automating data integration and governance activities as well as privacy, and compliance-related topics, the data fabric puts metadata as the most critical element [99]. Hechler et al. [97] differentiate the data fabrics paradigm from other data architectures by the predominant **use of AI to automate the metadata creation and management**, which is inline with the idea of active metadata.

According to Hechler et al. [97], ensuring trustworthy AI by broadening the scope of traditional data governance toward unified AI governance is another requirement for data fabrics. This is a relatively new area that goes beyond the inclusion of AI artefacts into the governance realm, because it requires to detect bias, drift, and decreasing accuracy and precision and needs to propose corrective actions to realign AI artefacts to business goals.

Furthermore, so-called self-service capabilities play a critical role that allow users to independently perform tasks that would traditionally require IT experts. Hechler et al. [97] provide concrete examples: (1) access various assets through the knowledge catalog based on roles, without needing additional credentials, (2) metadata includes details about how to access assets (e.g., (No)SQL, REST APIs, etc.), (3) a holistic view of AI/ML assets (4) metadata generation through AI to automate tasks like resource allocation and semantic annotations, (5) semantic search uncovers correlations and recommends relevant assets for business purposes, (7) data curation tasks such as data quality assessment and refinement are manageable by users without expert intervention. These examples require the creation of standardized data pipelines based on comprehensive metadata management for the automated and systematic transformation, cleansing, and integration of data [114, 113, 108].

An increasingly important pipeline is the support of the life cycle of AI processes [97], i.e., the management of the in- & outputs and the creation of ML models. However, it is important to note that, although researchers have recognized the potential [101], more traditional BI and data science activities, such as reporting, modeling and analysis tools are not considered the core scope of a data fabric, but rather fall under the responsibility of the data consumers [103].

For a definition we follow Blohm et al. [114]:

Definition 5. [**Data Fabric**] A Data fabric is a data architecture "*for attaining reusable data integration services, data pipelines and semantics for flexible and integrated data delivery [40]. It builds on the analysis, creation, and usage of metadata [103, 100], which are modeled as a knowledge graph.*" [114]

3.5.2. Requirements

Li et al. [100] identify key enabling technologies and Serra et al. [107] provide key features. Based on all of the previous findings we provide technical requirements for data fabrics:

DF1 — Polystore and Decentralized Storage: ensures data can reside across data lakes, data warehouses, or other systems, reducing redundancy and enhancing scalability.

DF2 — Data Virtualization: abstracts data complexity allowing seamless access to distributed sources via real-time querying and integration without requiring a central data repository.

DF3 — AI-Driven Metadata Management: metadata management must be (semi-)automated using AI to generate, enrich, and maintain metadata dynamically. This supports efficient data discovery, lineage tracking, and governance, while reducing manual intervention.

DF4 — Self-Service Capabilities: standardized pipelines driven by metadata enable automated, systematic data transformation, cleansing, and integration, allowing users to perform exploration, curation, and analysis tasks in GUI-based workflows.

We introduce a data fabric architecture in fig. 7, based on Hechler et al. [97], Macias et al. [108] and Priebe et al. [103]. Heterogeneous data sources are first ingested into storage. We illustrate how the data fabric architecture utilizes various storage technologies in a decentralized polystore. Those may include entire data lakes, warehouses, file storages like HDFS or cloud storages like S3. As the fabric's main concern is not the storage itself, but the associated metadata for merging the various parts into a unified view, we decided to leave this part out of the core data fabric area. Similarly, we view the transformation layer for data processing at scale as belonging to the storage component. The fabric is responsible for coordinating the required operations. However, it is ultimately the underlying storage, that is responsible to execute resource-intensive data processing operations. This opposes Priebe et al. [103], who include storage and integration and interoperability into the fabric's core functionality. Even in their illustration one can make a case to divide a data fabric into a storage & operations component and a

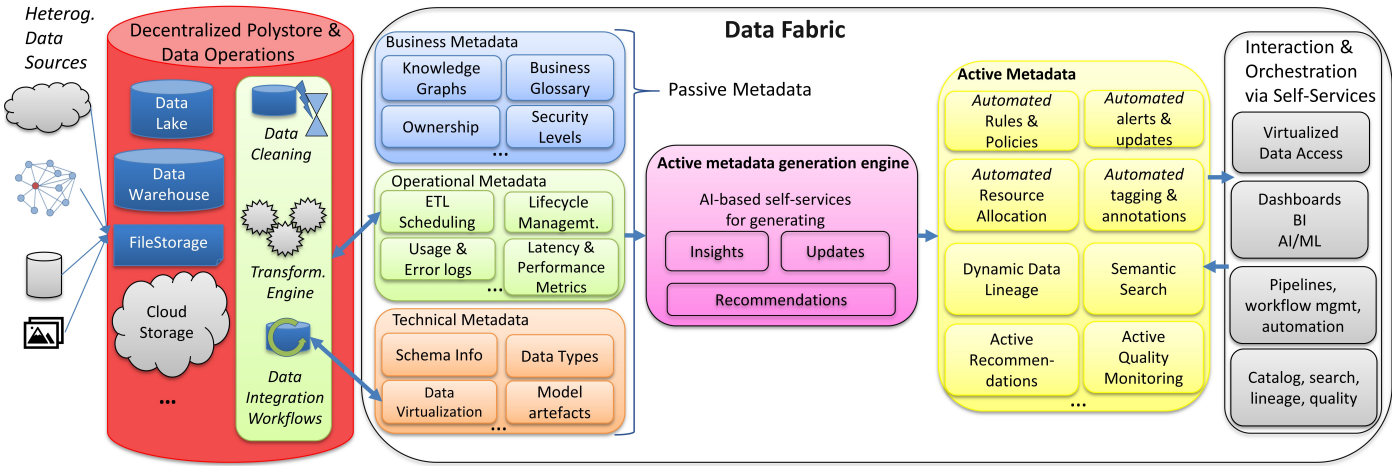

Figure 7: Data fabric architecture inspired by [103, 97]

knowledge catalog section. Essentially, we view the data fabric as knowledge catalog that connects various underlying data storages through comprehensive metadata management capable of data access and transformation via predefined standardized data pipelines.

Following the illustration for active metadata by Hechler et al. [97], for the data fabric we first deal with passive metadata divided into the three categories for metadata introduced by [115]. Business metadata serializes the KGs and business or domain related knowledge. The operational metadata orchestrates ETL processes and other data processing operations on the underlying storage, collects logs and manage the data lifecycle. Technical metadata extracts schema information, data types, etc. Therebym it enables data virtualization and collects metadata about the connections between analytical models and their artefacts and training datasets.

These are then converted into active metadata by the responsible generation engine. Those include tools powered by AI methods to enable users to discover insights, perform updates on the data, and receive recommendations. These AI methods are not meant to be crafted by users, but rather the data fabric should provide ready-to-use self-services that are sufficiently abstracted and simplified to make anyone capable of using them. The self-services enable a range of data discovery methods, such as semantic search, recommendations, automated semantic labeling, tagging, and annotations.

Finally, the interaction layer needs to provide an intuitive GUI that enables unfamiliar users to operate and utilize the self-services for integrating, searching and transforming data and orchestrating workflows.

3.6. Data Mesh

Centralized data teams, while highly specialized, often lack business domain expertise, creating bottlenecks in scaling data-driven decision-making across the enterprise [116, 114]. As different business units generate data in a decentralized manner, traditional architectures managed by central IT departments struggle to manage increasing volume and variety [117, 39, 118]. On the other hand, if the organization is divided in multiple business units to manage datasets specific to their domain, a lack of transparency and communication can lead to replicated effort and data silos. Here, a self-serve platform and employing proper standards through federated governance may enable reuse of datasets by others. Furthermore, centralized data management raises concerns over data ownership and governance. Without clear responsibility for data quality and maintenance, organizations risk compromised data integrity. Shifting ownership back to the domains creates a strong sense of responsibility over the created data that will improve data quality [39, 99].

For such reasons, Dehghani [118, 39] introduces the so-called data mesh, which is not (yet) a typical data architecture with concrete implementations, or infrastructure specifications, but rather "a socio-technical data approach" [39]. Interestingly, the concept draws inspiration from microservice architectures in software engineering [119]. While traditional data platforms remain centralized, a data mesh applies the microservice paradigm to data architectures, where each domain functions as an independent service. Just as microservices emerged to overcome the limitations of monolithic applications, the data mesh addresses the constraints of centralized data structures.

Another major motivating factor is the historical focus of data and analytics research on technological components rather than usage-driven approaches [120, 111], which led to overlooking the significance of federated architectures, governance, and organizational design [114]. This is the main reason for bottlenecks in centralized data platforms. Most studies have centered on designing technical solutions rather than addressing the organizational, procedural and governance challenges inherent in large-scale data management. Araujo et al. [121] even go as far as calling the data mesh a paradigm shift in the field of data architectures. In this new paradigm, data is seen as the main concern of a truly data-oriented organization, and the pipelining tools and specific architectures itself are seen as a secondary concern.

Several publications mention a close relationship between the data fabric and the data mesh [97, 111, 114, 107, 99]. James Serra [107] describes data fabrics as an architectural and tech-

nological framework that operates within or across domains in a data mesh, supporting its components and enabling synergy rather than competition. While a data mesh offers a decentralized, domain-oriented blueprint, data fabrics provide the underlying infrastructure to implement it. Multiple data fabrics can coexist within a mesh, and this integration approach also extends to other data architectures like data warehouses or lakehouses.

Despite the theoretical foundation of a data mesh and related frameworks, research on their real-world application remains limited. While studies have explored technological architectures, privacy concerns, and individual case studies, there is little systematic analysis of how industry implements these concepts [99]. However, Araujo et al. [121] outline two industrial use cases:

Zalando [122], a leading European fashion platform, initially relied on a centralized data lake but faced challenges such as lack of data ownership, poor data quality, and scalability bottlenecks as data sources and consumers grew. To address these issues, the company implemented a data mesh, introducing decentralized data ownership, domain-oriented teams, and a shift toward treating data as a product rather than a by-product. This transition enabled interoperability and decentralizing data responsibilities and improved governance through a metadata layer.

Netflix [123, 124, 125], serving over 150 million global users, generates petabytes of data daily and sought to integrate its studios into a unified system capable of handling this massive volume efficiently. Due to redundant pipeline efforts, maintenance overhead, lack of best practices, high latency, and error correction issues, Netflix implemented a data mesh that abstracts pipeline complexity for users. The infrastructure allows users to build pipelines while accessing a metadata catalog and standardized processes without needing deep technical knowledge.

3.6.1. Characteristics

Four principles define the data mesh as originally introduced by Dehghani [118]:

DM1 — Data-as-a-product: creation, management, and distribution of data products via a catalog. Data providers are responsible for publishing and provisioning them and collaborate with data owners and stewards for maintenance.

DM2 — Decentralized data ownership and architecture: each domain has ownership over its data products, ensuring that domain teams are responsible for the quality, maintenance, and lifecycle of their data products.

DM3 — Self-Service capabilities: a self-service data platform must be established to enable domain teams to create, manage, and consume data products without requiring extensive technical expertise.

DM4 — Federated governance: a federated governance model balances decentralized autonomy with global governance standards, ensuring that policies are enforced consistently across all domains while allowing flexibility for domain-specific rules.

The data-as-a-product principle frames data products as deliberately distributed and interconnected across individual "mesh nodes" [121, 126]. To give an example for a data product, consider the *MLFlow* model export functionality that outputs not only the trained model parameters, but also essential metadata, dependencies, and governance information. Each exported model includes environment specifications such as a `requirements.txt` or `conda.yaml`, ensuring reproducibility across different infrastructure setups. Additionally, *MLFlow* tracks experiment metadata, including training hyperparameters, dataset versions, and performance metrics, allowing for model lineage and governance.

Eichler et al. [127] recognize that data providers often lack incentives to share their data, because they face additional effort without immediate benefits, discouraging participation. To address this, they subdivide data products further into data assets. Data assets are raw, registered data in a company's catalog, while data products are refined versions with defined metadata, usage policies, and provisioning options. The transformation process follows three scenarios: (1) a provider directly registers data-as-a-product, (2) a consumer requests access to a data asset, prompting its transformation, or (3) a steward or another employee enriches metadata and submits it for approval. This staged approach reduces the initial burden on providers, ensuring they only need to add metadata when data is actually requested.

Domain-oriented ownership shifts data responsibility to individual business domains, allowing them to manage and utilize their own data according to their specific operational logic. This model fosters agility and autonomy by structuring data management around product-focused teams [121, 99]).

Self-service platforms serve as a technical abstraction layer, enabling unified access to data products while reducing complexity for users [128, 99]. By providing a standardized infrastructure, they eliminate redundant technical efforts and simplify the management of data products across their lifecycle [39, 99, 129].

Finally, federated computational governance establishes global governance standards that ensure interoperability and secure data access while enabling decentralized and autonomous governance by distributed teams. By balancing centralized standards with decentralized decision-making, federated governance ensures that data remains accessible and governed in a structured yet flexible manner [39, 99, 128]. Policies are also designed to balance local, i.e. domain-oriented, autonomy with global standardization. The federated governance model oversees security policies (defining authentication, authorization, and access control), interoperability policies (standardizing schema definitions, data contracts, and API structures), communication policies (guidelines for cross-domain data sharing), and documentation policies (requiring metadata enrichment, lineage tracking, and accessibility compliance)

For a definition we follow Blohm et al. [114] once more:

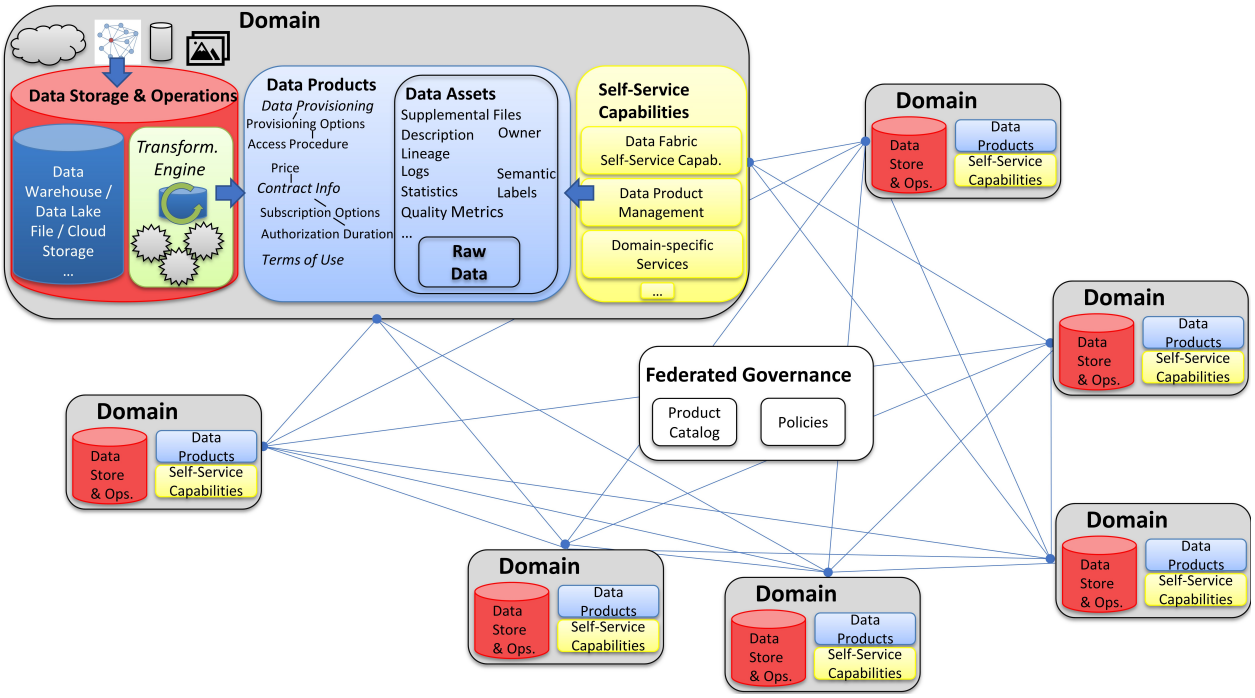

Figure 8: Data mesh architecture inspired by [127, 130, 97, 111]

Definition 6. [**Data Mesh**] "*Data mesh is a socio-technical, decentralized, distributed concept for enterprise data management [99]. It is characterized by the four principles of data-as-a-product, domain orientation, self-service platforms, and federated computational governance [131].*"

Figure 8 illustrates the data mesh architecture. Van der Werf et al. [132] identify four archetypes, out of which the fine-grained fully federated mesh or short pure data mesh is considered to be the most theoretical and decentralized but also the most mature version. We have chosen this pure form in which individual nodes directly exchange data products and deliberately left out a central distribution layer. Every atomic team or business unit is responsible for a single domain and the data products arising from it. While this approach provides high flexibility and strong domain specialization, it might also introduce challenges such as wasting resources, when data transformations are performed redundantly due to lack of communication or transparency between domains. Here, the maturity of the self-service capability and federated governance components are particularly relevant. In the figure, one node is enlarged representatively to display more details. In each domain heterogeneous raw data emerges that will be stored in a data storage and operations platform of choice. Those can be whatever is available and suitable in the specific domain, e.g. data warehouse/lake, cloud storages or file storages etc. Each domain is responsible for transforming and packaging raw data into data products. Data products are further subdivided into data assets and raw data as inspired by Eichler et al. [127]. In each domain, data consumers and providers have access to self-service capabilities. In the figure, the individual nodes inherit self-service capabilities from a data fabric to represent the close relationship and potential synergy between the two paradigms. They are extended by domain-specific services as well as data product management where domain teams register their data products along with associated metadata, quality metrics, and access policies and facilitate the distribution to other domains. This data product management and distribution is overseen by a central data federated governance layer to enforce global governance standards consistently ensuring policies, interoperability and secure data access.

4. Evaluation

The evaluation begins with a definition of the criteria used to assess and describe the discussed architectures. The criteria are defined in a way that allows to assign a quantitative measure (see table 1). Finally, these evaluation criteria in conjunction with the technical requirements from section 3 are evaluated against real-world systems to find a suitable match.

4.1. Dimensional Analysis

The preceding literature review and the in-depth examination of the discussed data architectures helps us to derive the following dimensions. These aim to (a) describe the characteristics of a given architecture, (b) highlight similarities and differences between them, and (c) provide a framework for assigning existing systems to one of the six categories in section 3:

Control, Governance & Trust.

D1 – Architectural Paradigm: Measures how centralized or decentralized the data architecture is in terms of control,

storage infrastructure, and processing. Higher scores reflect distributed, domain-driven models that promote autonomy.

D2 – Governance & Ownership: Examines the model of data control from centralized to federated ownership. Higher values indicate domain-level accountability and governance.

D3 - Security & Trust Measures how much the architecture gives priority to support secure access, data protection, auditability, and trust-building mechanisms such as data provenance, policy enforcement, or verifiable sharing.

Data Modeling & Understanding.

D4 – Data Formats: Evaluates how much the architecture considers flexibility in handling different data formats. Higher scores indicate native support for polystores and unstructured, semi-structured, and structured data.

D5 – Metadata Management: Assesses how metadata is handled from manual and static to dynamic, automated, and lineage-aware systems. Advanced setups foresee the use of AI or active metadata.

D6 – Knowledge Management: Assesses the extent to which the architecture supports modeling, managing, or leveraging domain knowledge alongside data. This includes formal approaches (e.g., SDM) and informal ones, such as business glossaries, tagging systems, or e.g., enterprise LLM-based chatbots [133] with richer data understanding, reasoning, and integration.

Data Processing.

D7 – Integration Focus: Captures how central integration is to the architecture's design. A high score means data integration is a foundational concern rather than an afterthought.

D8 – Data Virtualization: Describes how much focus the architecture places on abstracting storage details to enable querying flexibly and transparently without having to move data physically.

D9 – ML & Analytics Support: Captures the degree of support for ML and data science. Higher scores indicate built-in tools and pipelines for advanced analytics beyond traditional BI.

4.2. Evaluation of Real-World Systems

The Semantic Data Reservoir. (SEDAR) [15] has been designed as a modular system using open-source big data technologies, making it suitable as a workbench for research on scalable data architectures. Originally a classical data lake, SEDAR has evolved to support semantic data management (SDM) and ontology-based data access (OBDA) [83, 38]. SEDAR streamlines the ML life cycle using MLOps principles [135], supports AutoML [68] and is applied in production as part of a cyber-physical system for an Industry 4.0 case [136]. In the latest progress, a novel LLM-driven natural language interface has been added, which is designed to complement the user interface and to lower the technical barrier for users. The chat-based interface enables users to perform complex data management tasks using everyday language by leveraging the ability of LLMs to convert user queries into actions executed on the system.

From a control and governance perspective (D1–D3), SEDAR initially exhibits centralized characteristics. Security and policy enforcement remain basic. In terms of data modeling and understanding (D4–D6), SEDAR supports heterogeneous formats, semantic enrichment, and the use of ontologies and knowledge graphs as first-class citizens, making it highly suitable for knowledge-driven applications. Metadata management is automated and partially AI-augmented. Data processing (D7-D9) in SEDAR treats integration as a foundational concern, leverages Apache Spark for scalable processing, supports OBDA, and implements the full ML lifecycle following MLOps principles.

SEDAR partially overlaps with data lakehouse and warehouse paradigms due to its support for relational processing, SQL-based querying, and ML workflows. However, it does not enforce schema-on-write DW1 and the same storage type and data format across sources DLH1 and does not prioritize OLAP-centric optimization DW7. Given its semantic foundation, integration-centric design, and AI-enhanced capabilities, SEDAR increasingly fulfills key **data fabric** criteria (DF2-DF4), although currently SEDAR implements a centralized storage layer DF1. Hence, it moves away from the scope of classical data lakes, warehouses, and lakehouses by offering semantic interoperability, self-service capabilities and advanced automation across the data lifecycle.

The Stackable Data Platform. Stackable[1] is an open-source modular data platform built on Kubernetes that integrates a curated set of big data and data science tools into a cohesive platform-as-code architecture. It provides production-ready deployments of components like Kafka, Trino, Open Policy Agent, Hive via Kubernetes operators, with strong support for declarative configuration and infrastructure management.

From a control and governance perspective (D1–D3), Stackable promotes a hybrid governance model. While orchestration and policy enforcement can be centralized (e.g., via Kubernetes and Open Policy Agent), the platform enables domain-level autonomy in configuring and managing services, fostering decentralized ownership. It supports advanced access control through integration offering a relatively mature security and trust enforcement model. In terms of data modeling and understanding (D4–D6), Stackable is tool-agnostic and does not prescribe a specific data model. It supports diverse data types through connectors and engines, but leaves semantic enrichment and metadata management to external services. Similarly, ontologies and

[1] https://stackable.tech/, accessed on 02.05.2026

#	Dimension	1 = Low	3 = Medium	5 = High
			Control, Governance & Trust	
D1	Architectural Paradigm	Centralized	Federated	Decentralized
D2	Governance & Ownership	Centralized	Shared or hybrid	Federated / Domain-oriented
D3	Security & Trust	Minimal/static security	Moderate control with some auditing	Integrated, policy-enforced, trust-aware
			Data Modeling & Understanding	
D4	Data Formats	Structured only	Multi-format, some flexibility	Polystore / All formats
D5	Metadata Management	Manual / fixed	Automated extraction / tagging	AI-driven, dynamic, lineage-aware
D6	Knowledge Management	None	Basic domain modeling (glossaries, tags, rules)	Integrated knowledge layer (ontologies/KGs, LLMs)
			Data Processing	
D7	Integration Focus	Not central to the architecture	Localized or externalized	Core architectural concern
D8	Query Mechanisms	Local SQL only	SQL + NoSQL / partial federation	Federated / virtualized querying
D9	ML & Analytics Support	BI only	Some support for ML	Full ML pipeline and data science support

Table 1: Evaluation Dimensions for Data Architectures on a scale from 1 to 5, mapped to General Requirements

knowledge graphs are not natively integrated but can be layered in through additional tooling. Regarding data processing (D7–D9), Stackable enables large-scale data integration and analytics through its orchestration of distributed compute engines like Spark and Flink, and supports virtualization and query federation via Trino. However, it lacks global semantic abstraction and unified orchestration of ML workflows. While the platform supports full ML pipelines via integrations, MLOps is not a primary design focus.

Stackable does not conform strictly to data lakehouse or data fabric paradigms due to its lack of unified metadata, active governance layers, or enforced semantic models. However, it aligns most closely with **data mesh** principles particularly through its domain-oriented deployment patterns (DM2), emphasis on infrastructure-as-code, and federated governance (DM4). While it lacks the organizational enforcement, its architectural modularity and governance extensibility make it a strong enabler for infrastructure-centric data mesh implementations. If these are extended by self-service capabilities (DM3) in the future, possibly by integrating corresponding external services, then Stackable covers a mature technological base for data meshes.

Microsoft Fabric. Microsoft Fabric[2] is Microsoft's platform-as-a-service offering, unifying components from Power BI, Azure Synapse, Azure Data Factory, and OneLake. Designed for end-to-end data and AI workloads, it provides services for data engineering, data science, real-time analytics, and business intelligence — with strong emphasis on cloud-native scalability, collaboration, and ease of use.

From a control and governance perspective (D1–D3), Microsoft Fabric implements a federated orchestration model layered over centralized infrastructure. While the platform itself is centrally managed (e.g., via Azure), governance can be delegated across workspaces and domains. It offers comprehensive role based access management (RBAC), lineage tracking, and sensitivity labeling features. Data access and governance is handled by Microsoft Purview, though policy-based data sharing is still evolving. In terms of data modeling and understanding (D4–D6), Fabric supports a wide range of data formats promoting a polystore architecture via OneLake, where all data can be virtually accessed through a unified namespace. Metadata extraction and enrichment are automated and AI-augmented, particularly through Microsoft Copilot. While domain modeling support exists (via data cataloging, tagging, glossaries), deeper semantic constructs such as ontologies or knowledge graphs are not first-class features. Regarding data processing (D7–D9), Fabric offers highly integrated support for ETL pipelines, real-time streaming, and analytical processing. It supports distributed querying across files and tables, but semantic virtualization and ontology-based abstraction are not core design elements. Fabric fully integrates AutoML and MLOps capabilities enabling full ML lifecycle support from within the platform.

Microsoft Fabric aligns closely with the **data fabric** paradigm (DF1, DF3 & DF4), particularly in its unified infrastructure, active metadata, and orchestration of cross-role workflows. While data virtualization can be configured via so-called *shortcuts*, it falls short of the semantic richness for advanced data virtualization (DF2). Overall, Microsoft Fabric represents a commercially mature, enterprise-oriented fabric implementation that balances manageability and innovation, with strong vertical integration across Microsoft's data ecosystem.

5. Related Surveys & Work

The ISO/IEC TR 20547-1:2020 standard [139] defines a reference architecture for big data systems, providing a high-level framework to guide the design of interoperable and scalable solutions. While DATER shares the goal of architectural classification, it places greater emphasis on comparative evaluation across diverse architectures based on concrete technical requirements. A whitepaper by Fraunhofer ISST explores the conceptual and technical intersections between data mesh and data spaces [140], highlighting their complementary nature in decentralized data ecosystems. It argues that data mesh offers organizational and governance principles within enterprises, while data spaces provide the infrastructure for secure, cross-organizational data sharing. [103] present architecture diagrams in ArchiMate notation [141] for data warehouse, data mesh, and data fabric based on [23]. [142] compare data spaces

[2] https://www.microsoft.com/en-us/microsoft-fabric, accessed on 02.05.2026

	Data Warehouse [134, 27, 45]	Data Lake [63, 51, 62, 64]	Semantic Data Lake [38, 84, 76]	Data Lakehouse [96, 95, 41]	Data Fabric [97, 114, 107]	Data Mesh [39, 107, 121]
Control, Governance & Trust						
D1	1 – Centralized relational	1 – Centralized raw storage	1 – Semantic layer on centralized DL	1 – Hybrid of DW & DL	3 – Federated integration & Active Metadata	5 – Decentralized domain–oriented
D2	1 – Centralized	1 – Centralized	1 – Centralized	1 – Centralized	3 – Policy–managed	5 – Domain ownership
D3	2 – Strong role–based security but typically centralized and static	1 – Security is often minimal or bolted–on; access control at storage layer	2 – semantic technologies to account for it, but not a primary focus	3 – security models via modern engines, but not the primary focus	5 – central	4 – Domain–level responsibility; requires strong governance frameworks to scale
Data Modeling & Understanding						
D4	1 – relational	5 – raw data in original format	5 – raw data in original format	3 – Mainly relational	5 – arbitrary format	5 – Data products of arbitrary format
D5	1 – Fixed, scoped model	3 – Automated metadata extraction	4 – Metadata enriched with semantics	3 – Automated metadata extraction	5 – AI–driven, automated	2 – Decentralized, on–demand
D6	1 – None	1 – None	5 – Ontologies & KGs	3 – Basic	5 – KG–based metadata	2 – local vocabularies, but no global semantic layer
Data Processing						
D7	5 – ETL	3 – on–demand integration (ELT)	4 – Semantic–level integration	2 – Mix between DL and DW	4 – Integration driven by metadata	2 – Local integration within data products
D8	2 – access remains tightly coupled to internal engine and physical schemas	3 – Raw file-based querying; flexibility via tools like Spark/Presto, but limited abstraction	5 – OBDA-based access over heterogeneous data using semantic models.	3 – Combines files and tables with SQL engines, but limited abstraction from physical storage	5 – virtualized access, logical layers, and cross-platform queries	3 – Varies by domain; considers APIs or connectors; lacks global query abstraction
D9	1 – Focus on reporting & BI	2 – Not the primary focus	1 – No focus	5 – Full ML lifecycle support	3 – Moderate	1 — Not a focus

Table 2: Evaluation of Data Architectures against requirements

	SEDAR [15]	Stackable [137]	Microsoft Fabric [138]
Control, Governance & Trust			
D1	1 – Centralized	5 – Decentralized	3 – centralized governance, but workspace-level autonomy & logical domains
D2	1 – Centralized control plane	4 – centrally policy-managed, but configurable from within domains	3 – centralized governance tools, but support for role- and workspace-based ownership models
D3	1 – Basic RBAC; not a strong focus	4 – sophisticated and scalable measures	3 – fine-grained access-policies, but policy-based data sharing are not core.
Data Modeling & Understanding			
D4	3 – Raw data retained; main support relational	3 – arbitrary data can be processed, but support mainly designed for relational model	5 – services integrate a variety of formats suitable for polystore-style needs.
D5	5 – Automatically enriched with semantics	1 – Outsourced to external tooling	5 – AI-augmented, consistent with modern active metadata
D6	5 – Ontologies & Knowledge Graphs	1 – None	3 – basic domain modeling capabilities such as tagging, schemas, and glossaries.
Data Processing			
D7	5 – Integration is central	2 – ingestion and ELT workflows, integration is often manual or tool-driven	5 – Integration is central
D8	5 – OBDA	3 – virtualization via Trino connectors, but no global query abstraction	4 – can be configured via OneLake, but no global semantic layer
D9	5 – Full ML lifecycle (incl. AutoML)	2 – possible, but not a special focus	5 – Full ML lifecycle

Table 3: Evaluation of real-world data architecture systems based on technical requirements.

to data warehouses, lakes and different types of database systems.

In his book [107], James Serra gives a high-level introduction into modern data architectures: data warehouse, lake, lakehouse, fabric and mesh. As a textbook, it is more oriented towards readers unfamiliar with the field, explaining many concepts in much detail, which we assume to be available with readers of this article. We draw inspiration from this work heavily but shift the focus towards more in-depth technical detail, comparisons, and add more architectures.

A systematic literature review by Ataei and Litchfield [18] reveals that many data architectures rely on monolithic data pipelines with central storage, which hinders scalability and maintainability. Moreover, they point out that most existing reference architectures align closely with traditional data warehousing concepts, often integrating data lakes in a way that does not fully address modern data processing challenges. A major conclusion from their work is the necessity for decentralized and distributed big data architectures suggesting that emerging paradigms such as data mesh could better support scalability, data ownership, and interoperability as compared to centralized architectures. However, they note that industry adoption of such architectures remains slow due to the high cost and complexity of implementation. Additionally, the study identifies a lack of rigorous evaluation methodologies, making it difficult to assess their effectiveness in practice. Our work builds upon these insights by providing a more in-depth technical comparison of modern data architectures.

6. Conclusion

In response to the growing complexity of modern data landscapes, this chapter provides a structured exploration and comparison of contemporary data architectures.

The study reveals that no single architecture universally satisfies all technical needs, underscoring the importance of context-aware selection and hybrid solutions. This led to convergence of architectural paradigms, e.g., lakehouses and fabrics blending the strengths of lakes and warehouses. This comes with a lack of consensus in the academic and industrial literature regarding the definitions and boundaries of and between certain architectures, indicating a need for a clear characterization of

data architectures that motivated this work.

Modern data ecosystems increasingly shift toward decentralization, which promote domain-oriented ownership, federated governance, and peer-to-peer data sharing. The increasing importance of metadata and semantic technologies across architectures, highlights another shift toward more intelligent, discoverable, and machine-interpretable data systems. Architectures like data mesh and data fabric reflect more than technical designs. They also embody operational principles, organizational models, and mark a third cultural shift in data management.

Funding

This work was supported by the German Federal Ministry of Education and Research (BMBF) under grant no. 13FH557KX0 (project i2DACH).

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT, DeepL, and Grammarly to improve phrasing and clarity. After using these tools, the authors reviewed and edited the content and take full responsibility for the final version.

References

[1] A. Tzanetos, G. Dounias, Introduction to the special issue "data in management and decision engineering", Data in Brief 55 (2024) 110711.

[2] G. Muller, A reference architecture primer, Eindhoven Univ. of Techn., Eindhoven, White paper (2008) 24.

[3] C. Gröger, There is no ai without data, Communications of the ACM 64 (11) (2021) 98–108.

[4] A. P. Sheth, J. A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Comput. Surv. 22 (3) (1990) 183–236. doi:10.1145/96602.96604.
URL https://doi.org/10.1145/96602.96604

[5] G. M. Sang, L. Xu, P. de Vrieze, A reference architecture for big data systems, in: 2016 10th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2016, pp. 370–375. doi:10.1109/SKIMA.2016.7916249.

[6] G. M. Sang, L. Xu, P. De Vrieze, Simplifying big data analytics systems with a reference architecture, in: Collaboration in a Data-Rich World: 18th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2017, Vicenza, Italy, September 18-20, 2017, Proceedings 18, Springer, 2017, pp. 242–249.

[7] J. Van den Hoven, Data architecture: Blueprints for data., Information systems management 20 (1) (2003).

[8] D. Badampudi, C. Wohlin, K. Petersen, Experiences from using snowballing and database searches in systematic literature studies, in: Proceedings of the 19th international conference on evaluation and assessment in software engineering, 2015, pp. 1–10.

[9] B. Otto, The evolution of data spaces, in: Designing data spaces: The ecosystem approach to competitive advantage, Springer International Publishing Cham, 2022, pp. 3–15.

[10] J. Gessler, M. R. Cencic, C. Metzner, H. Wieker, K. Lindow, W. H. Schulz, Business models and organizational roles of data spaces: A framework for sustainable value creation, Data in Brief (2025) 111795.

[11] J. Lin, The lambda and the kappa, IEEE Internet Computing 21 (05) (2017) 60–66.

[12] D. Linstedt, M. Olschimke, Building a scalable data warehouse with data vault 2.0, Morgan Kaufmann, 2015.

[13] W. H. Inmon, D. Linstedt, Data architecture: a primer for the data scientist: big data, data warehouse and data vault, Morgan Kaufmann, 2014.

[14] M. Kiran, P. Murphy, I. Monga, J. Dugan, S. S. Baveja, Lambda architecture for cost-effective batch and speed big data processing, in: 2015 IEEE International Conference on Big Data (Big Data), 2015, pp. 2785–2792. doi:10.1109/BigData.2015.7364082.

[15] S. Hoseini, A. Ali, H. Shaker, C. Quix, Sedar: A semantic data reservoir for heterogeneous datasets, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 5056–5060.

[16] Plattform Industrie 4.0, Project gaia-x: A federated data infrastructure as the cradle of a vibrant european ecosystem, Tech. rep., Federal Ministry for Economic Affairs and Energy (BMWi), D-11019 Berlin, Germany (Oct. 2019).

[17] S. Venugopal, R. Buyya, K. Ramamohanarao, A taxonomy of data grids for distributed data sharing, management, and processing, ACM Computing Surveys (CSUR) 38 (1) (2006) 3–es.

[18] P. Ataei, A. Litchfield, The state of big data reference architectures: A systematic literature review, IEEE Access 10 (2022) 113789–113807.

[19] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz, B. Mitschang, The data lake architecture framework, in: BTW 2021, Gesellschaft für Informatik, Bonn, 2021, pp. 351–370.

[20] S. Geisler, M.-E. Vidal, C. Cappiello, B. F. Lóscio, A. Gal, M. Jarke, M. Lenzerini, P. Missier, B. Otto, E. Paja, et al., Knowledge-driven data ecosystems toward

data transparency, ACM Journal of Data and Information Quality (JDIQ) 14 (1) (2021) 1–12.

[21] K. M. Endris, P. D. Rohde, M. Vidal, S. Auer, Ontario: Federated query processing against a semantic data lake, in: Proc. DEXA, Vol. 11706 of LNCS, Springer, 2019, pp. 379–395. doi:10.1007/978-3-030-27615-7_{2}{9}.

[22] R. Hai, S. Geisler, C. Quix, Constance: An intelligent data lake system, in: Proc. SIGMOD, 2016.

[23] D. A. M. Association, et al., DAMA-DMBOK: Data management body of knowledge, Technics Publications, LLC, 2017.

[24] M. I. S. Oliveira, B. F. Lóscio, What is a data ecosystem?, in: Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age, dg.o '18, ACM, New York, NY, USA, 2018. doi:10.1145/3209281.3209335.
URL https://doi.org/10.1145/3209281.3209335

[25] C. J. Date, An introduction to database systems, Pearson Education India, 1977.

[26] N. Fathy, W. Gad, N. Badr, A unified access to heterogeneous big data through ontology-based semantic integration, in: 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), IEEE, 2019, pp. 387–392.

[27] W. H. Inmon, Building the data warehouse, John wiley & sons, 2005.

[28] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, I. Horrocks, The semantic web: The roles of XML and RDF, IEEE Internet Comput. 4 (5) (2000) 63–74. doi:10.1109/4236.877487.
URL https://doi.org/10.1109/4236.877487

[29] D. Laney, et al., 3d data management: Controlling data volume, velocity and variety, META group research note 6 (70) (2001) 1.

[30] A. Moniruzzaman, S. A. Hossain, Nosql database: New era of databases for big data analytics-classification, characteristics and comparison, arXiv preprint arXiv:1307.0191 (2013).

[31] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Communications of the ACM 53 (4) (2010) 50–58.

[32] L. Golab, M. T. Ozsu, Data stream management, Springer Nature, 2022.

[33] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012).

[34] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[35] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[36] I. M. Putrama, P. Martinek, Heterogeneous data integration: Challenges and opportunities, Data in Brief 56 (2024) 110853.

[37] J. Dixon, Pentaho, hadoop, and data lakes | james dixon's blog, [Online; accessed 22-Oct-2024] (2010).
URL https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/

[38] S. Hoseini, J. Theissen-Lipp, C. Quix, A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes, Journal of Web Semantics 81 (2024) 100819. doi:https://doi.org/10.1016/j.websem.2024.100819.

[39] Z. Dehghani, Data Mesh: Delivering Data-Driven Value at Scale, O'Reilly Media, Sebastopol, CA, 2022.
URL https://www.oreilly.com/library/view/data-mesh/9781492092346/

[40] E. Zaidi, E. Thoo, G. De Simoni, M. Beyer, Data fabrics add augmented intelligence to modernize your data integration, With Eric Thoo. Gartner Grou 17 (2019).

[41] M. Armbrust, A. Ghodsi, R. Xin, M. Zaharia, Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics, in: Proceedings of CIDR, Vol. 8, 2021, p. 28.

[42] Y. Huai, A. Chauhan, A. Gates, G. Hagleitner, E. N. Hanson, O. O'Malley, J. Pandey, Y. Yuan, R. Lee, X. Zhang, Major technical advancements in apache hive, in: Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 2014, pp. 1235–1246.

[43] M. Hentschel, J. Dees, F. Funke, M. Heimel, I. Oukid, Building a data management system for the cloud: Lessons learned and future directions, Datenbank-Spektrum 25 (1) (2025) 17–28. doi:10.1007/S13222-025-00494-9.
URL https://doi.org/10.1007/s13222-025-00494-9

[44] S. Chaudhuri, U. Dayal, An overview of data warehousing and olap technology, ACM Sigmod record 26 (1) (1997) 65–74.

[45] R. Bose, Advanced analytics: opportunities and challenges, Industrial Management & Data Systems 109 (2) (2009) 155–172.

[46] A. A. Harby, F. Zulkernine, Data lakehouse: A survey and experimental study, Information Systems (2024) 102460.

[47] M. Jarke, M. A. Jeusfeld, C. Quix, P. Vassiliadis, Architecture and Quality in Data Warehouses: An Extended Repository Approach, Information Systems 24 (3) (1999) 229–253.

[48] R. Kimball, M. Ross, The data warehouse toolkit: The definitive guide to dimensional modeling, John Wiley & Sons, 2013.

[49] A. Davoudian, L. Chen, M. Liu, A survey on nosql stores, ACM Computing Surveys (CSUR) 51 (2) (2018) 1–43.

[50] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis (Eds.), Fundamentals of Data Warehouses, 2nd Edition, Springer-Verlag, 2003.

[51] C. Quix, R. Hai, Data lake, in: Encyclopedia of Big Data Technologies, Springer, 2019. doi:10.1007/978-3-319-63962-8_{7}{-}{1}.

[52] P. Sawadogo, J. Darmont, On data lake architectures and metadata management, JJIS (2021).

[53] E. Scholly, P. Sawadogo, P. Liu, J. A. Espinosa-Oviedo, C. Favre, S. Loudcher, J. Darmont, C. Noûs, Coining goldmedal: a new contribution to data lake generic metadata modeling, arXiv preprint arXiv:2103.13155 (2021).

[54] Y. Zhao, I. Megdiche, F. Ravat, Data lake ingestion management, arXiv preprint arXiv:2107.02885 (2021).

[55] M. Farid, A. Roatis, I. F. Ilyas, H.-F. Hoffmann, X. Chu, Clams: Bringing quality to data lakes, in: Proc. SIGMOD, 2016.

[56] S. Galhotra, U. Khurana, Semantic search over structured data, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Association for Computing Machinery, New York, NY, USA, 2020, p. 3381–3384. doi:10.1145/3340531.3417426.

[57] H. Dibowski, S. Schmid, Y. Svetashova, C. A. Henson, T. Tran, Using semantic technologies to manage a data lake: Data catalog, provenance and access control, in: SSWS@ISWC, 2020.
URL https://api.semanticscholar.org/CorpusID:229357020

[58] R. Eichler, C. Giebler, C. Gröger, H. Schwarz, B. Mitschang, Modeling metadata in data lakes - A generic model, Data Knowl. Eng. 136 (2021) 101931. doi:10.1016/J.DATAK.2021.101931.
URL https://doi.org/10.1016/j.datak.2021.101931

[59] A. Nambiar, D. Mundra, An overview of data warehouse and data lake in modern enterprise data management, Big data and cognitive computing 6 (4) (2022) 132.

[60] R. Ramakrishnan, B. Sridharan, J. R. Douceur, P. Kasturi, B. Krishnamachari-Sampath, K. Krishnamoorthy, P. Li, M. Manu, S. Michaylov, R. Ramos, et al., Azure data lake store: a hyperscale distributed file service for big data analytics, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 51–63.

[61] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, S. E. Whang, Managing google's data lake: an overview of the goods system, IEEE Data Eng. Bull. (2016).
URL http://sites.computer.org/debull/A16sept/p5.pdf

[62] S. Azzabi, Z. Alfughi, A. Ouda, Data lakes: A survey of concepts and architectures, Computers 13 (7) (2024) 183.

[63] R. Hai, C. Koutras, C. Quix, M. Jarke, Data lakes: A survey of functions and systems, IEEE TKDE 35 (12) (2023) 12571–12590.

[64] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, P. C. Arocena, Data lake management: challenges and opportunities, Proc. VLDB Endow. 12 (12) (2019) 1986–1989.

[65] I. Gartner, Gartner says beware of the data lake fallacy, [Online; accessed 29-Jun-2025] (2014).
URL https://www.gartner.com/en/newsroom/press-releases/2014-07-28-gartner-says-beware-of-the-data-lake-fa

[66] J. Dixon, Data lakes revisited, [Online; accessed 22-Oct-2024] (2014).
URL https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/

[67] M. Jarke, C. Quix, On warehouses, lakes, and spaces: The changing role of conceptual modeling for data integration, in: J. Cabot, C. Gómez, O. Pastor, M. Sancho, E. Teniente (Eds.), Conceptual Modeling Perspectives, Springer, 2017, pp. 231–245. doi:10.1007/978-3-319-67271-7_16.
URL https://doi.org/10.1007/978-3-319-67271-7_16

[68] S. Hoseini, M. Ibbels, C. Quix, Enhancing machine learning capabilities in data lakes with automl and llms, in: European Conference on Advances in Databases and Information Systems, Springer, 2024, pp. 184–198.

[69] M. Bacco, A. Kocian, S. Chessa, A. Crivello, P. Barsocchi, What are data spaces? systematic survey and future outlook, Data in Brief 57 (2024) 110969.

[70] C. Quix, R. Hai, I. Vatov, Metadata extraction and management in data lakes with GEMMS, Complex Syst. Informatics Model. Q. (2016).

[71] P. N. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, J. Darmont, Metadata systems for data lakes: models and features, in: European conference on advances in databases and information systems, Springer, 2019, pp. 440–451.

[72] R. Eichler, C. Giebler, C. Gröger, H. Schwarz, B. Mitschang, Handle-a generic metadata model for data lakes, in: International Conference on Big Data Analytics and Knowledge Discovery, Springer, 2020, pp. 73–88.

[73] P. Ouellette, A. Sciortino, F. Nargesian, B. G. Bashardoost, E. Zhu, K. Q. Pu, R. J. Miller, Ronin: data lake exploration, Proceedings of the VLDB Endowment 14 (12) (2021).

[74] N. W. Paton, J. Chen, Z. Wu, Dataset discovery and exploration: A survey, ACM Computing Surveys 56 (4) (2023) 1–37.

[75] M. N. Mami, D. Graux, S. Scerri, H. Jabeen, S. Auer, J. Lehmann, Squerall: Virtual ontology-based access to heterogeneous and large data sources, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II, Vol. 11779 of Lecture Notes in Computer Science, Springer, 2019, pp. 229–245.

[76] A. Bagozi, D. Bianchini, V. D. Antonellis, M. Garda, M. Melchiori, Personalised exploration graphs on semantic data lakes, in: Proc. OTM Conf., Vol. 11877 of LNCS, Springer, 2019, pp. 22–39. doi:10.1007/978-3-030-33246-4_2.

[77] B. Sharma, Architecting Data Lakes—Data Management Architectures for Advanced Business Use Cases, 2nd Edition, O'Reilly Media, Sebastopol, CA, USA, 2018.

[78] P. Patel, G. Wood, A. Diaz, Data lake governance best practices, Dzone Guide Big Data—Data Sci. Adv. Anal. 4 (2017) 6–7.

[79] F. Ravat, Y. Zhao, Data lakes: Trends and perspectives, in: Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30, Springer, 2019, pp. 304–313.

[80] M. Madsen, How to Build an Enterprise Data Lake: Important Considerations before Jumping, Third Nature Inc., San Mateo, CA, USA, 2015.

[81] P. Zikopoulos, D. DeRoos, C. Bienko, R. Buglio, M. Andrews, Big Data Beyond the Hype, 1st Edition, McGraw-Hill Education, New York, NY, USA, 2015.

[82] A. Gorelik, The Enterprise Big Data Lake, O'Reilly Media, Sebastopol, CA, USA, 2016.

[83] A. Paulus, A. Burgdorf, A. Pomp, T. Meisen, Recent advances and future challenges of semantic modeling, in: Proc. 15th IEEE ICSC, IEEE, 2021, pp. 70–75. doi:10.1109/ICSC50631.2021.00016.

[84] H. Dibowski, S. Schmid, Y. Svetashova, C. Henson, T. Tran, Using semantic technologies to manage a data lake: Data catalog, provenance and access control, in: Proc. Scalable Semantic Web Knowledge Base Systems Workshop, Vol. 2757 of CEUR WS, 2020, pp. 65–80.

[85] H. Dibowski, S. Schmid, Using knowledge graphs to manage a data lake (2021).

[86] A. Beheshti, B. Benatallah, R. Nouri, A. Tabebordbar, Corekg: a knowledge lake service, Proc. VLDB Endow. 11 (12) (2018) 1942–1945. doi:10.14778/3229863.3236230.
URL http://www.vldb.org/pvldb/vol11/p1942-beheshti.pdf

[87] E. G. Kalayci, I. Grangel-González, F. Lösch, G. Xiao, A. ul Mehdi, E. Kharlamov, D. Calvanese, Semantic integration of bosch manufacturing data using virtual knowledge graphs, in: Proc. ISWC, Vol. 12507 of LNCS, Springer, 2020, pp. 464–481.

[88] D. Calvanese, L. Ding, A. Mosca, G. Xiao, Realizing ontology-based reusable interfaces for data access via virtual knowledge graphs, in: Proc. Italian SIGCHI (CHItaly), ACM, 2021, pp. 35:1–35:5. doi:10.1145/3464385.3464744.

[89] A. Pomp, A. Paulus, S. Jeschke, T. Meisen, ESKAPE: information platform for enabling semantic data processing, in: Proc. ICEIS, SciTePress, 2017, pp. 644–655. doi:10.5220/0006324906440655.

[90] A. Pomp, A. Paulus, A. Kirmse, V. Kraus, T. Meisen, Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures, Technologies 6 (3) (2018) 86.

[91] M. N. Mami, H. Jabeen, S. Auer, "teach me to fish": Querying semantic data lakes, https://www.researchgate.net/publication/322526357_'Teach_me_to_fish'_Querying_Semantic_Data_Lakes, accessed: 2024-11-04 (2018).

[92] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 5511–5519. doi:10.24963/IJCAI.2018/777.
URL https://doi.org/10.24963/ijcai.2018/777

[93] M. Armbrust, T. Das, L. Sun, B. Yavuz, S. Zhu, M. Murthy, J. Torres, H. van Hovell, A. Ionescu, A. Łuszczak, et al., Delta lake: high-performance acid table storage over cloud object stores, Proceedings of the VLDB Endowment 13 (12) (2020) 3411–3424.

[94] J. Schneider, C. Gröger, A. Lutsch, H. Schwarz, B. Mitschang, Assessing the lakehouse: Analysis, requirements and definition, in: Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023), Scitepress 2023, 2023.

[95] J. Schneider, C. Gröger, A. Lutsch, H. Schwarz, B. Mitschang, The lakehouse: State of the art on concepts and technologies, SN Computer Science 5 (5) (2024) 1–39.

[96] A. A. Harby, F. Zulkernine, From data warehouse to lakehouse: A comparative review, in: 2022 IEEE International Conference on Big Data (Big Data), 2022, pp. 389–395. doi:10.1109/BigData55660.2022.10020719.

[97] Y. Hechler, M. Weihrauch, Y. C. Wu, Data fabric and data mesh approaches with ai, Berkeley, CA, USA: Apress Berkeley (2023).

[98] M. DeBellis, L. Pinera, C. Connor, Interoperability frameworks: Data fabric and data mesh architectures, in: Data Science with Semantic Technologies, CRC Press, 2023, pp. 267–286.

[99] J. Bode, N. Kühl, D. Kreuzberger, C. Holtmann, Towards avoiding the data mess: Industry insights from data mesh implementations, IEEE Access (2024).

[100] X. Li, M. Yang, X. Xia, K. Zhang, K. Liu, A distributed data fabric architecture based on metadate knowledge graph, in: 2022 5th International Conference on Data Science and Information Technology (DSIT), IEEE, 2022, pp. 1–7.

[101] A. Abu Rumman, L. Al-Abbadi, Structural equation modeling for impact of data fabric framework on business decision-making and risk management, Cogent Business & Management 10 (2) (2023) 2215060.

[102] R. Thanaraj, M. Beyer, E. Zaidi, What is data fabric design?, https://info.cambridgesemantics.com/hubfs/What_Is_Data_Fabric_Gartner.pdf, accessed on 22.11.2024 (2021).

[103] T. Priebe, S. Neumaier, S. Markus, Finding your way through the jungle of big data architectures, in: 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 5994–5996. doi:10.1109/BigData52589.2021.9671862.

[104] N. Kuftinova, O. Maksimychev, A. Ostroukh, A. Volosova, E. Matukhina, Data fabric as an effective method of data management in traffic and road systems, in: 2022 Systems of Signals Generating and Processing in the Field of on Board Communications, IEEE, 2022, pp. 1–4.

[105] M. AKERMI, M. A. H. TAIEB, M. B. AOUICHA, Data virtualization enabling distributed data architectures: Data fabric and data mesh, International Journal of Computer Information Systems and Industrial Management Applications 15 (2023) 12–12.

[106] C. Software, Data fabric vs. data virtualization, accessed: 2024-11-22 (2024).
URL https://www.cdata.com/blog/data-fabric-vs-data-virtualization

[107] J. Serra, Deciphering Data Architectures, O'Reilly Media, Inc., 2024.

[108] A. Macías, D. Muñoz, E. Navarro, P. González, Data fabric and digital twins: An integrated approach for data fusion design and evaluation of pervasive systems, Information Fusion 103 (2024) 102–139.

[109] S. A. Rieyan, M. R. K. News, A. M. Rahman, S. A. Khan, S. T. J. Zaarif, M. G. R. Alam, M. M. Hassan, M. Ianni, G. Fortino, An advanced data fabric architecture leveraging homomorphic encryption and federated learning, Information Fusion 102 (2024) 102004.

[110] I. D. Martinez-Casanueva, L. Bellido, D. González-Sánchez, D. Lopez, Candil: A federated data fabric for network analytics, Future Generation Computer Systems 158 (2024) 98–109.

[111] P. Strengholt, Data Management at Scale, " O'Reilly Media, Inc.", 2023.

[112] M. M. Alvord, F. Lu, B. Du, C.-A. Chen, Big data fabric architecture: How big data and data management frameworks converge to bring a new generation of competitive advantage for enterprises, Enterprise Architecture Professional Journal (2022).

[113] S.-J. Moon, S.-B. Kang, B.-J. Park, A study on a distributed data fabric-based platform in a multi-cloud environment, International Journal of Advanced Culture Technology 9 (3) (2021) 321–326.

[114] I. Blohm, F. Wortmann, C. Legner, F. Köbler, Data products, data mesh, and data fabric: New paradigm (s) for data and analytics?, Business & Information Systems Engineering (2024) 1–10.

[115] C. Diamantini, P. L. Giudice, L. Musarella, D. Potena, E. Storti, D. Ursino, A new metadata model to uniformly handle heterogeneous data lake sources, in: Proc. ADBIS Short Papers & Workshops, Vol. 909 of CCIS, Springer, 2018, pp. 165–177. doi:10.1007/978-3-030-00063-9_{1}{7}.

[116] V. K. Butte, S. Butte, Enterprise data strategy: a decentralized data mesh approach, in: 2022 International Conference on Data Analytics for Business and Industry (ICDABI), IEEE, 2022, pp. 62–66.

[117] W. Brackenbury, R. Liu, M. Mondal, A. J. Elmore, B. Ur, K. Chard, M. J. Franklin, Draining the data swamp: A similarity-based approach, in: Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA '18, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3209900.3209911.

[118] Z. Dehghani, How to move beyond a monolithic data lake to a distributed data mesh (May 2019).
URL https://martinfowler.com/articles/data-monolith-to-mesh.html

[119] A. Ashraf, A. Hassan, H. Mahdi, Key lessons from microservices for data mesh adoption, in: 2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 2023, pp. 1–8. doi:10.1109/MIUCC58832.2023.10278300.

[120] A. Abbasi, S. Sarker, R. H. Chiang, Big data research in information systems: Toward an inclusive research agenda, Journal of the association for information systems 17 (2) (2016) 3.

[121] I. A. Machado, C. Costa, M. Y. Santos, Data mesh: Concepts and principles of a paradigm shift in data architectures, in: M. M. Cruz-Cunha, R. Martinho, R. Rijo, D. Domingos, E. Peres (Eds.), CENTERIS 2021 - International Conference on ENTERprise Information Systems, Vol. 196 of Procedia Computer Science, Elsevier, 2021, pp. 263–271. doi:10.1016/J.PROCS.2021.12.013.
URL https://doi.org/10.1016/j.procs.2021.12.013

[122] M. Schultze, A. Wider, Data mesh in practice: How europe's leading online platform for fashion goes beyond the data lake, Online video, available at: https://www.youtube.com/watch?v=eiUhV56uVUc [Accessed 06.02.205] (2020).

[123] J. Cunningham, Netflix data mesh: Composable data processing - justin cunningham, Online video, available at: https://www.youtube.com/watch?v=TO_IiN06jJ4 [Accessed 06.02.205] (2020).

[124] G. Pires, M. Cho, M. Liu, S. Jain, Streaming sql in data mesh, Netflix TechBlog, accessed: 2025-02-06 (November 2023).
URL https://netflixtechblog.com/streaming-sql-in-data-mesh-0d83f5a00d08

[125] A. Nguonly, A. Magalhães, O.-I. Nwoke, S. Afshar, S. Das, T. Liu, W. Liu, Y. Zeng, Data movement in netflix studio via data mesh, Netflix TechBlog, accessed: 2025-02-06 (July 2021).
URL https://netflixtechblog.com/data-movement-in-netflix-studio-via-data-mesh-3fdc

[126] J.-P. Soininen, G. Laatikainen, What is a data space–logical architecture model, Data in Brief (2025) 111575.

[127] R. Eichler, C. Gröger, E. Hoos, H. Schwarz, B. Mitschang, From data asset to data product - the role of the data provider in the enterprise data marketplace, in: J. Barzen, F. Leymann, S. Dustdar (Eds.), Service-Oriented Computing - 16th Symposium and Summer School, SummerSOC 2022, Hersonissos, Crete, Greece, July 3-9, 2022, Revised Selected Papers, Vol. 1603 of Communications in Computer and Information Science, Springer, 2022, pp. 119–138. doi:10.1007/978-3-031-18304-1_7.
URL https://doi.org/10.1007/978-3-031-18304-1_7

[128] A. Wider, S. Verma, A. Akhtar, Decentralized data governance as part of a data mesh platform: Concepts and approaches, in: 2023 IEEE International Conference on Web Services (ICWS), 2023, pp. 746–754. doi:10.1109/ICWS60048.2023.00101.

[129] I. Araújo Machado, C. Costa, M. Y. Santos, Advancing data architectures with data mesh implementations, in: International Conference on Advanced Information Systems Engineering, Springer, 2022, pp. 10–18.

[130] A. Dolhopolov, A. Castelltort, A. Laurent, Implementing federated governance in data mesh architecture, Future Internet 16 (4) (2024) 115.

[131] Z. Dehghani, Data mesh principles and logical architecture, accessed: 18 Jan 2024 (2020).
URL https://martinfowler.com/articles/data-mesh-principles.html

[132] D. van der Werf, J. Moreira, J. P. S. Piest, Towards a data mesh reference architecture, in: International Conference on Enterprise Design, Operations, and Computing, Springer, 2024, pp. 339–353.

[133] Y. Yun, H. Ye, X. Li, R. Li, J. Deng, L. Li, H. Xiong, Eicopilot: Search and explore enterprise information over large-scale knowledge graphs with llm-driven agents, arXiv preprint arXiv:2501.13746 (2025).

[134] C. J. Quix, Metadatenverwaltung zur qualitätsorientierten informationslogistik in data-warehouse-systemen, Ph.D. thesis, Bibliothek der RWTH Aachen (2003).

[135] S. Alla, S. K. Adari, What is mlops?, in: Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure, Springer, 2020, pp. 79–124.

[136] S. Hoseini, G. Zhang, D. Polke, A. Surjana, L. Wagner, C. Schmitz, C. Quix, Coatings intelligence: Data-driven automation for chemistry 4.0, in: 2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS), IEEE, 2024, pp. 1–6.

[137] S. D. P. Team, Interview on the stackable data platform, Interview with the Stackable team, conducted by the author, personal communication via phone interview (May 2025).

[138] D. Ghosh, Mastering Microsoft Fabric, Springer, 2024.

[139] International Organization for Standardization, ISO/IEC TR 20547-1:2020: Information technology — Big data reference architecture — Part 1: Framework and application process, https://www.iso.org/standard/71275.html, accessed: 2025-06-02 (2020).

[140] N. Jahnke, K. Bendiek, M. Klueß, Data Mesh und Data Spaces – Zwei Konzepte für dezentrale Datenökosysteme, Whitepaper Fraunhofer ISST, https://www.bk.admin.ch/dam/bk/de/dokumente/dti/DatenoekosystemSchweiz/Grundlagen/Fraunhofer-ISST_Data-Mesh-und-Data-Spaces_Whitepaper%20.pdf.download.pdf/Fraunhofer-ISST_Data-Mesh-und-Data-Spaces_Whitepaper%20.pdf, accessed: 2025-06-02 (2023).

[141] A. Josey, ArchiMate® 3.1-A Pocket Guide, Van Haren, 2019.

[142] M. Atzori, A. Ciaramella, C. Diamantini, B. Martino, S. Distefano, T. Facchinetti, F. Montecchiani, A. Nocera, G. Ruffo, R. Trasarti, et al., Dataspaces: Concepts, architectures and initiatives, in: CEUR workshop proceedings, Vol. 3606, CEUR-WS, 2024.