LUPINE PUBLISHERS

Current Trends in Computer Sciences & Applications

DOI: 10.32474/CTCSA.2023.03.000157

ISSN: 2643-6744

Research Article

Navigating the Data Architecture Landscape: A Comparative Analysis of Data Warehouse, Data Lake, Data Lakehouse, and Data Mesh

Benjamin Wong*

Department of Computing, Hong Kong Polytechnic University

***Corresponding author:** Benjamin Wong, Department of Computing, Polytechnic University, Hong Kong

Received: August 21, 2023 **Published:** October 18, 2023

Abstract

In the rapidly evolving field of data management, numerous terminologies, such as data warehouse, data lake, data lake house, and data mesh, have emerged, each representing a unique analytical data architecture. However, the distinctions and similarities among these paradigms often remain unclear. The present paper aimed to navigate the data architecture landscape by conducting a comparative analysis of these paradigms. The analysis a identified and elucidated the differences and similarities in features, capabilities, and limitations of these architectural constructs. The study outcome serves as a comprehensive guide, assisting practitioners in selecting the most suitable analytical data architecture for their specific applications.

Introduction

In the dynamic world of data science and machine learning, the foundation of meaningful analysis lies in robust data architecture. Over the past few decades, a multitude of analytical data architectures have been proposed and established, ranging from traditional, such as data warehouse (DWH), to more modern approaches, such as data mesh. However, understanding their nuances, similarities, and differences can be challenging.

In the present study, we aimed to provide a systematic overview and a comparative analysis of various analytical data architectures, including DWH, data lake, data lake house, and data mesh. We delved into the details of classic data warehouse (both Kimball and Inmon styles) [1,2], Gartner's logical DWH and data fabric concepts [3,4], Dehghani's data mesh proposal [5], Linstedt's data vault [6], data lake, and lambda and kappa architectures, and the data lake house [7], which is characterized by data bricks. All of these architectural proposals are shown in Figure 1 on an(approximate) timeline according to their creation or publication.

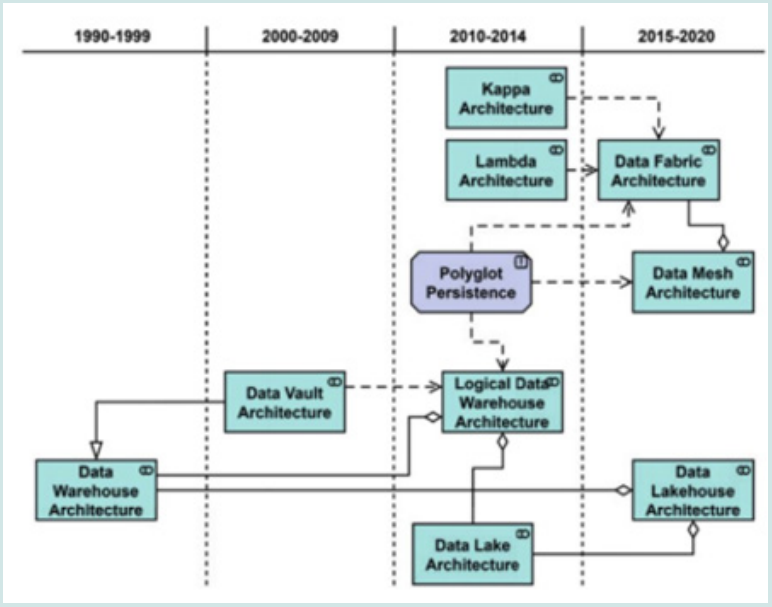

Figure 1: Architectural paradigms and their emergence.

Given the vast array of data architectures, maintaining a clear overview is challenging. Drawing from the concept of patterns and pattern systems familiar in software development, we attempt to establish a similar approach for data architectures. Our goal is to provide a guide for practitioners in selecting the most suitable analytical data architecture for their specific needs and situations.

Background

a) Data Warehouse, Data Vault, and Data Lake

Data Warehouse (DWH), Data Vault and Data Lake the meanwhile just as "classic" Data Lake, are not discussed here for reasons of space. The corresponding architectural patterns, based on the structure presented in the previous section, can be found on the website mentioned. A distinction is also shown between Kimball and Inmon DWHs. Even if variants such as (near) real-time DWH and the like with stream processing have emerged, data integration in DWH usually occurs with batch- based ETL (or ELT) processing. Regarding storage technology, DWH is based on relational database technology. As shown in Figure 1, the data vault architecture is presented as a specialization of the DWH architecture. Contrarily, a data lake typically uses file/object storage (Hadoop, AWS S3, or Azure Blob Store in the cloud). Stream processing is added to data integration.

b) Logical Data Warehouse

Building on this, in the present study, we want to take explore further the concept of the logical DWH as the first architectural paradigm. This was presented by Gartner in 2012 [4]. It provides recommendations on how companies can set up needs-based data management for analytical ap- plications. According to the author, architectural approaches, such as DWHs, data lakes, and data virtualization, should not be considered competing solutions, but rather as complementary components of an overarching architecture. Gartner also explicitly mentioned sandboxes and stream processing as components of a logical DWH architecture. Regarding data models, Gartner introduced dimensional and data vault modeling. There is a focus on DWH automation, which is why we also included this as a capability in the area of "Data Modeling & Design."

c) Persistence Kappa Architectures

The combination of several repositories, i.e., different data storage technologies, has already been applied and named "polyglot persistence." This term was apparently first used by Scott Leberknight in 2008 and then by Martin Fowler in 2011. It basically means that every midsize company should combine technologies, such as distributed file or object storage systems and relational and graph databases, even within a single application, as needed. Polyglot persistence is more of a principle that advocates the use of a polyglot data store, i.e., a data storage solution that combines different storage technologies, as an architectural component, i.e., the lambda and kappa architectures. However, they are not full data architectures, but rather focus on the data integration function. Lambda combines batch processing in a so-called batch layer with stream processing in a so-called speed layer, whereas kappa relies exclusively on streaming (possibly through log processing or change data capture). Both architectures were developed and published in 2014/15.

d) Data Lakehouse

Even if the term "data lake house" appears to have been used earlier, it was coined clearly in a blog by Databricks in 2020 and developed (also sponsored by Databricks) by Inmon et al. [8]. According to this blog, a data lake house is defined as "a new, open paradigm that combines the best elements of data lakes and data

warehouses." In this respect, the basic idea is comparable to the previously described logical DWH according to Gartner. Contrarily, the Data Lake house (at least in the case of Databricks) does not rely on a combination of several storage technologies, but on file/object storage that has been expanded to include transaction consistency. Since the boundaries are more fluid as compared with those of logical DWH, a distinction is not made between data lake and DWH as architectural components, but between data records with raw and prepared data ("Curated Data").

e) Data Fabric and Data Mesh

The combination of different data storage and integration techniques, but not limited to concrete architecture archetypes, such as data lake or DWH, led to the term "data fabric," which was originally coined in 2015 by George Kurian of NetApp and then by Gartner [9]. According to the authors, data fabric is defined as "a design concept to achieve reusable and advanced data integration services, data pipelines, and semantics, aimed at flexible and integrated data delivery." Data fabric can be seen as the successor and generalization of logical DWH. It builds on the idea of polyglot persistence or polyglot data store that combines storage approaches, including relational databases, graph databases, and/or file/object storage. A main focus of the data fabric concept is metadata, which, according to Gartner, comprises an (extended) data catalog and a knowledge graph containing semantically linked metadata. The use of artificial intelligent or machine learning. Gartas. DerLans [10] differentiates transactional services, the logical DWH, reporting and analysis tools are not seen as the core of the data fabric concept but fall under the responsibility of the data consumer. Data fabric is basically already leading the basic idea of "Data Products" even if this terminology is not used explicitly. This finally resulted in the idea of a data mesh according to Dehghani [5]. Dehghani argues that the existing centralized and monolithic data management platforms, which lack clear domain boundaries and ownership of domain data, fail in large enterprises with a diverse number of data sources and consumers. In a data mesh, the domains must display their records as a domain host internal data and make them available as data products. Although the individual domain teams independently control the technology used to store, process, and deliver their data products, a common platform ensures uniform interaction with the data products. As with data fabric, there is a focus on metadata with a data catalog that provides a cross-domain inventory of available data representing products. As with data fabric, reporting and analytical tools are not the focus (hence "business intelligence & data science" is outside the data mesh architecture box). However, in contrast to the other data architecture paradigms presented, the data mesh concept takes the data sources into consideration. The operational data are served via operational data products (or their interfaces), similar to analytical data products.

Methodology

In this study, we performed a comparative analysis to examine the various analytical data architectures. Our approach involves the following steps: Selection of Architectures: We focused on the following four key data architectures: DWH, data lake, data lake house, and data mesh. These architectures were selected due to their prominence in the field and their representation of both traditional and modern approaches to data management. Criteria for Comparison: We identified several criteria for comparison, including scalability, performance, data consistency, data integration, security, and cost. They were the chosen criteria because they represent key considerations in data architecture selection and implementation. Data Collection: For each architecture, we collected information from various sources, including academic papers, industry reports, and technical documentation to serve as basis for our comparison. Analysis: We conducted a detailed analysis of each architecture based on our chosen criteria, which involves comparing and contrasting the features, capabilities, and limitations among the architectures. Presentation of Findings: We presented our findings in a clear and structured manner, allowing for easy comparison among the different architectures. We also provided a discussion of the implications of our findings for the practitioners in the field. With this methodology, we aimed to provide a comprehensive and unbiased comparison of the selected data architectures. Our goal is to assist practitioners in making informed decisions about the most suitable architecture for their specific needs and applications.

Comparison

In the present article, we have made a first systematic presentation of important analytical data architectures in a common structural framework. Particularly, logical data warehouse and data mesh architectures were covered in more detail. Based on Inmon et al.'s [7] study findings, we compared for the first time according to the dimensions "Data Format," "Data Types," "Data Access," "Reliability," "Governance and Security," "Performance," "Scalability" and "Supported Use Cases." We added "Data Ingestion Processing" in the analysis because the architecture variants also differ significantly in their support for streaming and data virtualization. The original table shown in the source, which is limited to DWH, data lake, and data lake house, is naturally somewhat biased toward the data lake house. In Table I, we have attempted to somehow objectify this. On the above-mentioned website, we will cover other architectures and paradigms, such as the lambda and kappa architectures in detail, and present them as architecture patterns — up to a sample system similar to the software design patterns of the "Gang of Four" [11]. There will also be detailed templates with context, problem, and solution sections that will hopefully provide even better guidance for choosing the right architecture paradigms.

Table 1: Comparison of architecture paradigms considered.

	Data Warehouse	Data Vault	Data Lake	Logical Data Warehouse	Data Fabric	Data Mesh	Data Lakehouse
Data Format	Relational database	Relational database	File/object save to base more open file formats Structured data,	Different data formats with polyglot persistence Structured data in DWH, semi- structured, textual, and unstructured data in data lake	Different data formats with polyglot persistence Structured data	Different data formats with polyglot per-sistence	File/object save to base more open file formats
Data Storage	physical proprietary storage, structured data, limited support for semi-structured data	physical proprietary storage, structured data, limited support for semi structured data	Semi- structured data, textual data, unstructured (raw) data	Batch-/ETL cycles in DWH Streaming possible in data lake	Semi- structured data, textual data, unstructured (raw) data	structured data, semi-structured data, textual data, unstructured (raw) data	Structured data, semi-structured data, textual data, unstructured (raw) data
Data Ingression Processing	Usually daily, limitation by batch/ ETL cycles	Usually daily, limitation by batch/ETL cycles	Almost real time possible through streaming	Open APIs for file/object access in data lake SQL in DWI and limited across the board with data virtualization Poor quality and reliability in data lake	Up to real time possible through data virtualization (depending on data product) Structured	Up to real time possible through data virtualization (depending on data product)	Combination of batch/ETL cycles, possibly data streaming
Data Access	SQL	SQL	Open APIs for file/object access, limited SQL access	High Quality ACID transactions in DWH	Depends on implementation	Open APIs for operational data products, SQL (with data virtualization) for analytical data products	Open APIs for File/ object access, SQL
Data Consistency	High quality and reliability, ACID transactions	Medium quality in raw vault, high quality in business vault, ACID transactions	Minor quality and reliability		Depends on implementation	Depends on data product.	Minor quality and reliability in raw data, high quality, ACID transactions at processed data
Data Governance & Security	Fine-grained safety and governance (row/column level)	Fine-grained safety and governance (row/column level)	Weak safety and governance in file level	Fine granular in DWH Weak safety and Governance in data lake	Depends on implementation	Depends on data product	Fine-grained for SQL access, weak safety and governance at file/object access
Performance	High because it can be specifically optimized	High because it can be specifically optimized	Rather low as file/object based, depending on data usage (MapReduce, Spark)	High in DWH, rather low in data lake (depending on data usage)	Depends on implementation	Depends on data product	Medium since limited until long optimization possibilities
Scalability	Scaling becomes ex- potentially more ex- pensive	Scaling will be more expensive but easier to handle than in the classic DWH Classic Bl, reports, dashboards, SQL	Highly scalable for large Amounts of data at low cost	Scaling expensive in the DWH, cheap in data lake	Depends on implementation	Depends on data product	Highly scalable for large amounts of data at low cost
Business Use Cases	Classic BI, reports, dashboards, SQL		Data science, especially machine learning	Diverse, from classic Bl to self- service BI to machine learning	Data lake product, classic Bl, self-service Bl, machine learning, also operational applications	Data lake product, classic Bl, self- service Bl, machine learning, also operational applications	Diverse, from classic Bl to self-service Bl to machine learning

Discussion

The comparative analysis presented in this paper provides a comprehensive overview of the current landscape of data architectures. However, as with any rapidly evolving field, new questions and areas for future research emerge.

1. Interoperability and Integration: As organizations often employ multiple data architectures, it is unclear how these different architectures can interoperate and integrate with each other. Future research should explore strategies and technologies to ensure effective data integration across different architectures.

2. Evolution of Data Architectures: As data management continues to evolve, the architectures underpinning it will also evolve. How the emerging technologies and trends, including artificial intelligence and edge computing, might shape the future of data architectures should be investigated in the future.

3. Performance Benchmarking: Although this paper has compared the theoretical aspects of different data architectures, empirical studies that could benchmark their performances under different scenarios would be valuable, as such studies could provide more concrete data to guide practitioners.

4. Security and Privacy: With the increasing importance of data privacy and security, future research should focus on these aspects in the context of different data architectures. How can these architectures ensure data security and comply with privacy regulations?

5. Adoption Challenges: Implementing a new data architecture is not without its challenges. Future work should explore these challenges in more detail and develop strategies to overcome them.

6. Impact of Data Architecture on Data Science: How does the choice of data architecture impact the work of data scientists? This is an important question that deserves further exploration.

To summarize, although this paper provides a comprehensive comparison of different data architectures, it also opens up several avenues for future research. As we continue to explore the complex landscape of data architecture, ongoing discussion and research are crucial to keep up with the rapid pace of change in this field.

Conclusion

Our analysis has illuminated the distinct characteristics and similarities among four key data architectures (data warehouse, data lake, data lake house, and data mesh). Although DWHs offer a traditional, structured approach to data management, Data Lakes provide a more flexible and scalable solution for storing raw data. Contrarily, data lake houses attempt to combine the best of both worlds, offering the structured analysis of DWHs and the flexibility of data lakes. Lastly, data mesh represents a paradigm shift toward a more decentralized approach to data architecture, focusing on domain-oriented data ownership. However, a one-size-fits-all solution in data architecture is lacking. The choice of architecture depends on various factors, including the nature of the data, specific use case, scalability requirements, and existing IT infrastructure, among others. Therefore, practitioners should carefully consider these factors when selecting the most suitable data architecture for their specific needs. In conclusion, as the field of data management continues to evolve, so too will the architectures that underpin it. We hope that the results of our comparative analysis will serve as a valuable guide for practitioners when navigating the complex landscape of data architecture, aiding them in making informed decisions that best suit their specific applications.

References

1. R Kimball and M Ross (2013) The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling.
2. W H Inmon (2005) Building the Data Warehouse.
3. E Zaidi, E Thoo, G De Simoni, M Beyer (2019) Data Fabrics Add Augmented Intelligence to Mod- ernize Your Data Integration, Gartner, Tech. Rep :00450706.
4. Understanding the logical data warehouse: The emerging practice, 2012.
5. How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh - martin fowler.
6. D Linstedt and M Olschimke (2015) Building a Scalable Data Warehouse with Data Vault 2.0.
7. B. Inmon (2021) Building the Data Lakehouse, pp. 2021-2021.
8. B Inmon, M Levins, and R Srivastava, Building the Data Lakehouse.
9. E. Zaidi (2019) Data Fabrics Add Augmented Intelligence to Modernize Your Data Integration.
10. R Van Der and Lans, Logical Data Fabric to the Rescue: Integrating Data Warehouses, Data Lakes, and Data Hubs.
11. E. Gamma (1995) Design Patterns: Elements of Reusable Object-Oriented Software pp. 2021-2021.

DOI: 10.32474/CTCSA.2023.03.000157