Weaving Architectural Patterns: Data Fabric, Lakehouse, and Mesh

By Barry Devlin

Published by Denodo, October-December 2021

Weaving Architectural Patterns: I – Data Fabric

Since its first incarnation almost 35 years ago in my IBM Systems Journal article, the data warehouse (DW) has remained a key architectural pattern for decision-making support. By *decision-making support*, I mean everything from simple reporting and querying to AI-based predictive analytics.

Of course, the first DW architecture was designed for queries and reports. A variety of additional concepts of varying breadth and quality—such as data mining, the logical data warehouse, and data lakes—have expanded the scope of the original DW thinking or have sought to displace it entirely. None have succeeded in killing off the DW. Now, over the past couple of years, three new adversaries have emerged: data fabric, data mesh, and the data lakehouse.

Will any or all of them kill off the data warehouse? It's a fun question, but the wrong one! A better question—and one with a more useful answer—is: how can they complement DW thinking, and how could they improve decision-making support in an era of rapidly expanding digital business? We'll look at each of these new approaches in this series. In this post, we'll focus on data fabric.

What is a Data Fabric?

Data fabric in its current meaning dates to 2016 and in particular to a Forrester Wave report on "Big Data Fabric." The *Big* has been replaced by *Enterprise* in the latest, 2020 version of the report, reflecting the industry-wide shift from big data to all data. A 2019 definition of data fabric as "a distributed data management platform, where the sole objective is to combine various types of data storage, access, preparation, analytics, and security tools in a fully compliant manner, so that data management tasks become easy and smooth" shows the broad scope of the framework but suggests the challenges in specifying it in practice.

However, Denodo's Ravi Shankar explained it well in his 2017 article, describing six functional layers:

1. Data ingestion from every potential source, of every possible type and structure of data and information
2. Processing and persistence in any form of data store, such as a cloud-based repository, Hadoop, database system, or file
3. Orchestration of cleansing and transformation processes, to enable integration with other data from multiple sources
4. Data discovery, using data modelling, data virtualization, and other tools, to enable data to be accessed and integrated correctly and usefully across different sources or "silos"
5. Data management and intelligence to define and enforce data governance and security
6. Data access, delivering data to businesspeople or their applications

This list immediately and clearly exposes the fundamental meaning of data fabric. In every way, it can be compared directly with the concept of the *logical data warehouse* promoted by Gartner, among others, since the early 2010s. Furthermore, data virtualization lies at its heart in layer 4,

providing the technology to access and join data across multiple sources. The remaining layers reprise the basic functionality of a data warehouse or, to a lesser extent, a data lake.

What's New with Data Fabric in 2021?

Given that it has been around for five years, one might ask why it has seemingly become one of the flavors *du jour* now. For example, data fabric is listed as one of Gartner's Top 10 Data and Analytics Trends for 2021. Logical data warehouse and data virtualization are well-known and widely implemented. So, what is new?

The key realization expressed in the data fabric concept is that the real-time integration of data envisaged in layers 3 and 4 above is actually highly complex to define and manage over time in an environment where data sources change rapidly and unpredictably in both content and structure.

This leads to the inclusion of "active metadata" to drive AI algorithms that can simplify and automate the design and operation of integration and discovery functions. Active metadata means that metadata can change and grow automatically as the environment evolves. This is achieved via the provision of advanced analytics over a "connected knowledge graph"—a deep ontology of all the information/data in the environment stored and managed in a graph database/engine.

As Forrester's 2020 report phrases it, this amounts to "a unified, intelligent, and integrated end-to-end platform to support new and emerging use cases." The key word here is *intelligent*, emphasizing artificial intelligence function to automate multiple aspects of defining and operating the environment, including "process integration, transformation, preparation, curation, security, governance, and orchestration to enable analytics and insights quickly."

Architectural and Product Considerations

Data fabric emphasizes the ongoing shift to a hybrid, multi-cloud/on-premises data processing platform. As with most platforms defined by big analyst firms, such as Gartner and Forrester, data fabric is framed around the offerings of a variety of software vendors, who then compete to include (usually by acquisition) the additional function demanded by the platform. Such function accretion enables larger data management vendors to support the entire architecture.

However, implementers of data fabric should focus more closely on the key functional enablers and the vendors that offer the best support for them. In the case of data fabric, data virtualization is central, as it was in the logical data warehouse, and continues to be even more important in data fabric.

The second key functional enabler is comprehensive, active metadata. This has been a long-term challenge for data warehousing and, even more so, for data lakes. Many metadata management / data catalog products have emerged over recent years, often focusing on the metadata collection issue. Many have been acquired by various larger data management vendors. More recently, knowledge graph approaches have taken center-stage. Vendors such as Cambridge Semantics and Stardog have been capitalizing on the popularity of knowledge graphs as used by Google and Facebook and are positioning themselves as key to data fabric implementation.

Conclusion

Data fabric, as an architectural platform, has a solid historic foundation and rich product-set availability in its core storage and data virtualization functional areas. However, the key novel functionality of using AI and knowledge graphs to automate management, and its use via active metadata, is still an emerging product area.

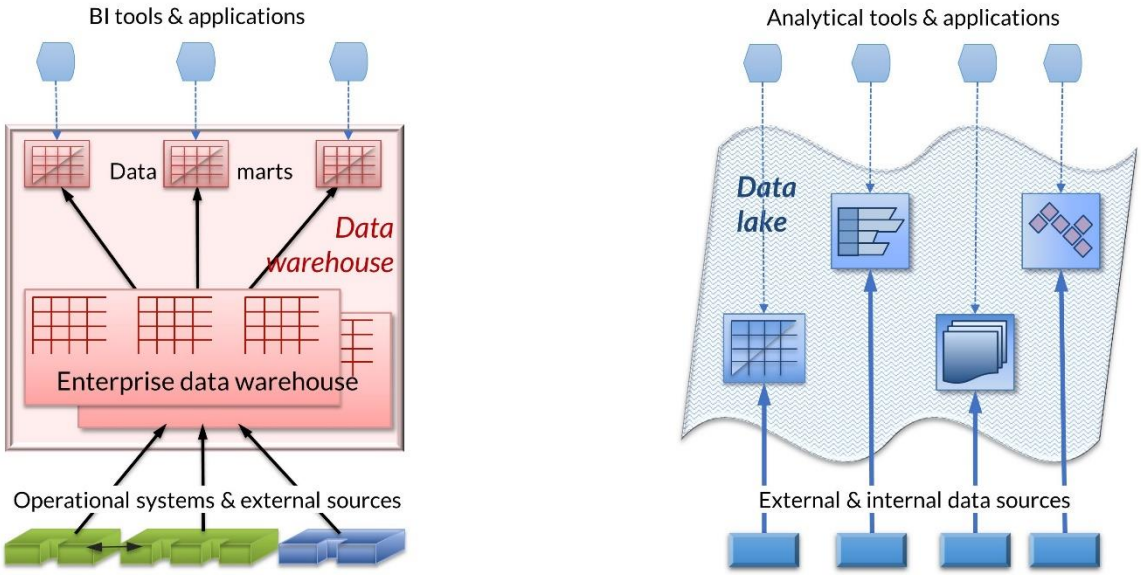
In subsequent parts, I'll be looking at the two other emerging architectural patterns: data mesh and the data lakehouse.

Weaving Architectural Patterns II – Data Lakehouse

In the previous section, I described the data fabric architectural pattern. Before turning to the data lakehouse, let's first look at what I mean by an *architectural pattern*.

The Power of a Pattern

An architectural pattern, or logical architecture, describes a system's major functional components—data/information, process, or both—in a generic, simplified way, often focused on some set of business and/or environmental characteristics that are its key drivers. It avoids specific products, attempting to be as vendor neutral as possible. The figure below shows typical architectural patterns for a basic hub-and-spoke data warehouse and a data lake. As well as allowing a brief discussion of the principles of architectural patterns, it offers a perfect starting point for describing the data lakehouse, which—as the name implies—attempts to combine the two worlds.



Different arrow and widget designs represent different types of function and storage. The different types of data stores—relational, flat file, graph, key-value, and so on—have different symbols, and it is clear that a data lake contains many different storage types, while the data warehouse is fully relational. The heavy black arrows show bulk data transfer between data stores, whereas the light blue dashed arrows show access to data stores from applications. The black arrows into and out of the enterprise data warehouse (EDW) denote strong transformation function (schema on write) and are arranged in funnel patterns to imply consolidation of data in the EDW. These functions are largely IT controlled. The solid blue feeds to the data lake denote minimum levels of transformation of the incoming data and are operated independently, often by data scientists or business departments. Data in a lake is therefore often inconsistent—the so-called data swamp—and it is up to the users to manage its meaning (schema on read) and—a more difficult proposition—quality.

Introducing the Data Lakehouse

The data lakehouse concept was introduced early in 2020 by Databricks, a company founded in 2013 by the original creators of Apache Spark™, Delta Lake and MLflow. According to Databricks, the data lakehouse is a platform that "combines the best elements of data lakes and data warehouses—delivering data management and performance typically found in data warehouses with the low-cost, flexible object stores offered by data lakes." It builds on a data lake foundation because, according to the authors of a recent lakehouse Q&A, they often contain more than 90% of the data in the enterprise. The lakehouse therefore attempts to eliminate or, at least, greatly reduce the common approach of copying subsets of that data to a separate data warehouse environment to support traditional BI that data lake tools struggle to do.

According to the original paper defining the architecture, a lakehouse has the following features:

1. ACID transactions for concurrent data read and write supporting continuous update
2. Enforcement, evolution, and governance for schemata such as star and snowflake
3. Support for using BI tools directly on the source data, reducing staleness and latency, and eliminating the copies of data found in a combined data lake and warehouse solution
4. End-to-end streaming to serve real-time data applications
5. Open and standardized storage formats, such as Parquet, with a standard API supporting a variety of tools and engines, including machine learning
6. Storage decoupled from compute, for scaling to more concurrent users and larger data sizes
7. Support for diverse workloads, including data science, AI, SQL, and analytics
8. Support for diverse data types ranging from structured to "unstructured" data

The first four points are directly related to the goal of supporting modern data warehousing from a data lake base, while the remainder relate more to "big data" applications.

As is often the case with vendor-defined architectures, no architectural pattern is offered for the lakehouse. Rather, a product or set of products is suggested as implementing the concept. In this case, the complementary/competing open-source products/projects mentioned are: Delta Lake (an open-source storage layer supporting ACID transactions), Apache Hudi (a platform for streaming data with incremental data pipelines), and Apache Iceberg (an open table format for very large analytic datasets). Apache Spark provides an underlying platform.

Data Lakehouse—Questions Arising

Although building on top of the data lake, the features described and the products mentioned focus heavily on the ingestion, management, and use of *highly structured* data, as is the case with a data warehouse. In this respect, the data lakehouse appears to build a data warehouse on a different platform than traditional relational databases, while enabling AI and analytic tools to use the same underlying data. The extension of good data management and manipulation practices beyond relational data to structures such as Parquet, ORC (Optimized Row Columnar), and even CSV files is welcome and valuable in reducing the prevalence of data swamps. Although traditional relational databases would seem to fit neatly, they are largely excluded from consideration, because of concerns about "vendor lock-in" typical of open-source vendors.

ACID transaction support is strongly emphasized. This is driven by the growing need to support streaming inputs from clickstreams and the Internet of Things, as well as the need to offer version management for more file-based ingestion. The business focus is thus very much slanted toward

analysis of data from external Internet-sourced data, in large volumes and at high velocity. Similar technology considerations pervade traditional data warehouse thinking, although the focus there is often more on data from operational systems.

The lakehouse description provides little evidence of how the governance and version management of more loosely structured data/information—such as text, audio, image, and video—is to be achieved. This omission is concerning, given the volumes of such data/information and its importance in AI.

Perhaps most concerning of all is the underpinning notion that ***all*** the data a business needs can be housed in the same (lake-based) platform and processed by a single set of tools. Of course, the notion speaks to the desire for good governance and security, reduced data management, and lower storage costs. However, experience has proven that data monoliths have their own serious drawbacks: inability to meet all business needs equally, complex and lengthy implementation, and ongoing inflexibility to change. Based on prior knowledge of data warehouses and data lakes, it seems likely that the lakehouse will also need to adapt to be part of a logical, distributed approach, just one of a variety of data stores accessed via data virtualization technology.

Conclusion

The data lakehouse is very clearly a pattern coming from teams experienced in open-source / extended Hadoop / cloud-centric data storage and delivery systems. This focus may be appropriate for enterprises with more distributed environments, where the data center of gravity leans heavily toward cloud. Enterprises with more on-premises, traditional environments—even if aiming toward significant cloud migration—would be advised to dig more deeply into the details of the lakehouse approach to understand its limitations. A clear and sufficiently detailed architectural pattern from the designers of the data lakehouse would be of substantial benefit.

In the next and final part of the series, I'll be looking at data mesh, an architectural pattern that starts from an uncommon point of view.

Weaving Architectural Patterns III – Data Mesh

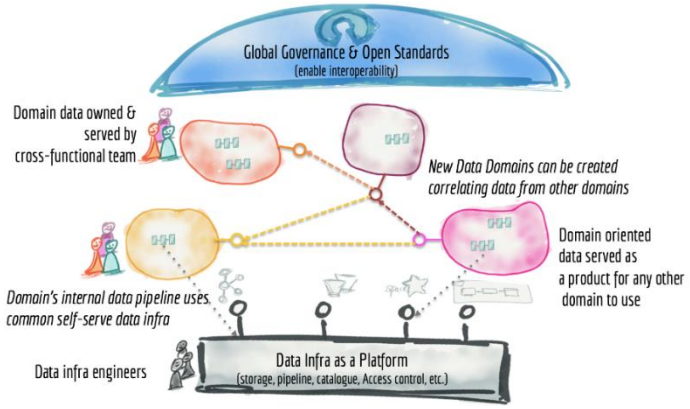
Software systems designers often structure their thinking around the underlying functional and data/information components of their desired applications. This approach—analogous to the scientific method of breaking a system into its smallest sub-parts in order to understand how it works—forms the basis of traditional architectural patterns, such as the data warehouse, as we saw in part II of this series. Data fabric, as seen in part I comes from similar thinking. The data lakehouse (also in part II) on the other hand, appears to derive its functional foundation more from existing products than decomposition of functionality.

Introducing the Data Mesh Approach

The designers of the data mesh approach start from yet another distinctly different place. Starting with her seminal article in May 2019, Thoughtworks' Zhamak Dehghani introduces data mesh as an alternative to "the centralized paradigm of a lake, or its predecessor data warehouse." Denodo's Alberto Pan offers a concise summary of the problems of a centralized architecture—limited business understanding in the data team, inflexibility of centralized systems, and slow, unresponsive data provisioning—that drove the evolution of data mesh. The solution, according to Dehghani is

"that the next enterprise data platform architecture is in the convergence of *Distributed Domain Driven Architecture*, *Self-serve Platform Design*, and *Product Thinking* with *Data*."

Dehghani admits this to be a buzzword-laden definition. However, at its core is the concept of domain-driven design, a seemingly simple idea that the structure and language of software should match the "business domain" in which it operates. The approach has been successfully applied in (operational) application development, has influenced the decomposition of applications into services based on business domains, and has driven the rise of the microservices architecture. In data mesh, the concept is applied to data, leading to a data architecture where business units[1] take responsibility for the data they need, from initial creation to consumption. This is a model build on the ownership of data as "products" by business units. Dehghani says: "Instead of *flowing* the data from domains into a centrally owned data lake or platform, domains need to *host and serve* their domain datasets in an easily consumable way," leading to a self-service mindset for data use. The figure below shows Dehghani's very high-level architectural pattern for a data mesh.



Of particular interest is that the technical infrastructure is reduced to a single box, placing the focus firmly on the concept of data domains and how they are created, owned, and used by businesspeople. Functional components and data stores are omitted. The architecture majors on how data should be managed by domain and how other domains request or "pull" data from the creative domains. Observe also the "Global Governance and Open Standards" construct. This is a box seldom seen in functional architectures because so much of it depends on organizational constructs and largely manual processes.

Dehghani's concise definition of the building blocks of a data mesh platform is: "*distributed data products* oriented around domains and owned by *independent cross-functional teams* who have embedded data engineers and data product owners, using common *data infrastructure* as a platform to host, prep and serve their data assets… an intentionally designed distributed data architecture, under centralized governance and standardization for interoperability, enabled by a shared and harmonized self-serve data infrastructure."

As defined in a subsequent article, a data product consists not only of data and metadata, but also of the code for creating it from its sources, allowing access, and enforcing policies, as well as the infrastructure to build and store it. This service-oriented architecture (SOA) approach is unique to data mesh over all the other patterns we have explored.

[1] I use the term "unit" loosely to represent any organizational structure formed around business process scope ranging from a specific task, such as order entry, to a complete function, such as marketing.

Data Mesh Discussion

A data architecture with governance in pole position can only be a good thing. So too is a focus on addressing the issues of siloed data storage and delivery schemes, particularly as the analytical environment becomes both more complex in structure and demands ever more timely delivery.

However, experienced data warehouse architects will probably be concerned about how data from different domains can be reconciled and made consistent, given the encapsulated nature of data products and their independent development by different teams. The traditional solution to this reconciliation problem is, of course, to bring data together in an enterprise data warehouse structure, an approach explicitly excluded by the data mesh. Another approach, however, can be used. This consists of an overarching, integrated data model combined with data virtualization technology, as outlined by Pan in the previously referenced blog post.

Both approaches require some level of up-front definition of an over-arching information or semantic model to ensure cross-domain interoperability. Dehghani does not discuss this topic. The risk, therefore, may be that siloed data pipelines are replaced by siloed data domains.

Conclusion

Data mesh offers a new and unique approach to the challenges of providing an environment for decision-making support and analytics systems. Its focus on delivering, managing, and using data in a domain-oriented paradigm offers much needed and novel insights into the organizational challenges of such systems. However, adopting a microservices approach to data may be too big a step for many more traditional data warehouse and lake shops. And product support for some aspects of the architecture is still in the early stages of evolution. As a result, data mesh may be more suited to larger businesses with complex data analytics needs and mature IT organizations.

As we've considered the three architectural patterns—data fabric, data lakehouse, and data mesh—over this and the previous two blog posts, it should be clear that there exist multiple approaches to stepping up to the promises of digital business and addressing the problems associated with delivering the actual data and information. Among these three approaches—as well as data warehouse, logical data warehouse, and data lake—none can be chosen as a clear winner, nor can any be dismissed out of hand. Each has its strengths and weaknesses; each addresses a particular aspect of the challenges of data management.

Data architects embarking on creating new or extending existing decision-making support and analytics environments would be well advised to study all these competing and complementary architectural patterns with a view to weaving the most appropriate solution for their specific situation.

Original posts at Denodo Data Virtualization Blog:

Part I, Data Fabric

Part II, Data Lakehouse

Part III, Data Mesh