International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 15(2023) pp. 1-12

Submitted: 8 Feb, 2023; Accepted: 3 Apr, 2023; Publish: 9 June, 2023

Data Virtualization Enabling Distributed Data Architectures: Data Fabric and Data Mesh

Montasser AKERMI[1], Mohamed Ali HADJ TAIEB[1], and Mohamed BEN AOUICHA[1]

[1]Data Engineering and Semantics Research Unit, Faculty of Sciences of Sfax,
University of Sfax, Sfax, Tunisia
montaccep@gmail.com,
{*mohamedali.hajtaieb, mohamed.benaouicha*}*@fss.usf.tn*

Abstract: **Organizations are facing increasing challenges in effectively managing and utilizing data as a strategic asset within the rapidly evolving data landscape. Traditional monolithic data architectures are struggling to keep pace with the heterogeneous nature and high velocity of generated data, making it necessary to explore alternative paradigms. This article investigates the potential of data virtualization as a data integration pattern, discussing its core capabilities and features, as well as its role in enabling distributed data architectures such as data fabric and data mesh. Furthermore, the article provides a comparison of common data integration patterns and highlights the advantages of data virtualization especially in the distributed data architecture.**

Keywords: Data Virtualization, Data Integration, Data Architecture, Data Fabric, Data Mesh

I. Introduction

This article extends upon the findings presented in the conference paper (Akermi et al., 2023) to provide a comprehensive overview of data virtualization and its role in enabling distributed data architectures.

Data virtualization has emerged as a modern approach to data integration that provides a unified solution to managing and utilizing all the data within an organization (Van der Lans, 2012). With an increasing number of structured, semi-structured, and unstructured data sources being generated and used, data integration has become a major challenge for organizations. Traditional data integration patterns, such as Extract, Transform, Load (ETL) and Enterprise Service Bus (ESB), are limited in their ability to provide real-time access to diverse data sources, often resulting in delayed data delivery and decreased data accuracy (Naeem et al., 2022). Data virtualization, on the other hand, provides a real-time solution through the use of a virtual data layer. This layer acts as an abstraction between the data consumers and the underlying data sources, providing a unified view of the data (Muniswamaiah et al., 2019b).

Data virtualization offers several advantages over traditional data integration patterns. First, it eliminates the need for data movement, reducing the complexity and cost associated with data integration. Instead of copying data from source systems to a central repository, data virtualization enables real-time access to data sources, providing an up-to-date view of the data. Secondly, it offers increased flexibility and scalability, allowing organizations to easily add new data sources and remove existing ones, without affecting existing data consumers. Finally, data virtualization enables organizations to leverage the power of their data, providing faster and more accurate data delivery, which can lead to increased efficiency and competitiveness.

Moreover, data virtualization is key to enabling distributed data architectures, such as data fabric (Li et al., 2022). These architectures aim to facilitate access and sharing of disparate data sources, leading to more streamlined data management and increased efficiency (Machado et al., 2022). Data virtualization provides real-time access to data across disparate systems, without having to move the source data to a new repository, making it an ideal solution for organizations seeking to implement a distributed data architecture.

The article aims to provide a better understanding of data virtualization and its potential to enhance data management capability and enable distributed data architectures. Section 2 compares common data integration patterns, while Section 3 focuses on data virtualization and its key capabilities and features. In Section 4, the potential of data virtualization in enabling distributed data architectures, such as data fabric and data mesh, is discussed. The article concludes with a summary of its findings and suggestions for future work in this field.

II. Comparison of Common Data Integration Patterns

Data integration (DI) can be classified into physical integration and virtual integration (Doan et al., 2012). Physical data integration involves combining data from multiple sources into a single, unified location or repository, typically through the use of extract, transform, load processes or similar methods. This approach often requires the data to be physically moved or copied from its original location to the new repository, resulting in the creation of multiple replicas of the same

data. Virtual data integration, on the other hand, uses techniques such as data virtualization to provide a unified view of data from multiple sources without the need for physical replication. This approach allows users to access and query data in real-time, regardless of where it is physically stored, providing a more efficient and cost-effective solution for data integration.

When evaluating data integration patterns, several factors should be considered in order to determine their suitability. These include *ease of use*, *maintainability*, *performance*, *error handling* capabilities, *reusability*, and *extensibility*. Ease of use can be measured by the simplicity of setup, configuration, and management for both information technology (IT) teams and business users. Maintainability refers to the ease of updating, scaling, and modifying the integration method to align with changing business requirements. Performance is determined by the speed and efficiency of data processing and integration. Error handling capabilities refer to the effectiveness of detecting and correcting errors in the data. Reusability pertains to the ability to apply the integration method to multiple scenarios or use cases. Extensibility is the capability of accommodating new data sources, formats or integration scenarios.

It is important to note that this comparison is based on a general overview and may vary depending on the specific implementation and vendor of each data integration pattern.

Table 1 shows a comparison of six common data integration patterns; ETL, ESB, Change Data Capture (CDC), Data as a Service (DaaS), Streaming Data Integration (SDI), and Data Virtualization (DV). The rating ranges between 1 to 5 stars.

Fig. 1 illustrates this comparison in terms of overall score and agility. The overall score, represented on the x-axis, is calculated based on the aggregate of the previously discussed six criteria. The agility score, represented on the y-axis, reflects the ability of each data integration pattern to handle changes easily, and quickly adapt to new use cases, data sources, and changing business requirements.

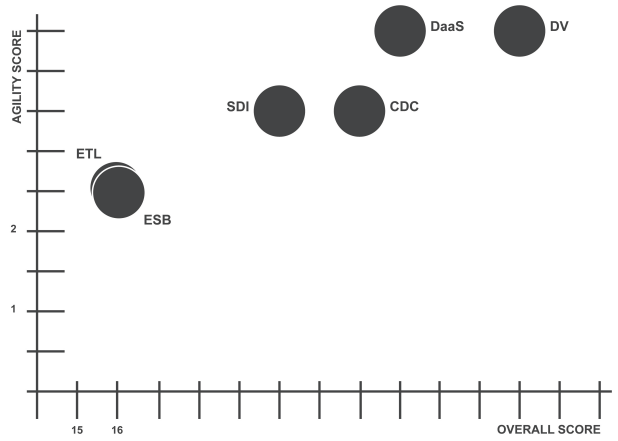


Figure. 1: A comparison between common data integration patterns

When evaluating a DI pattern for agility, it is important to consider the following factors:

- *Modularity*: The ability to add, remove, or modify data sources, data formats, or integration scenarios without disrupting existing integration processes.
- *Scalability*: The ability to handle an increasing volume and velocity of data without a significant impact on performance.
- *Automation*: The ability to automate data integration processes to improve efficiency and reduce the cost of integration.
- *Self-service capabilities*: The ability for business users to integrate data without IT teams involvement, which can help speed up the integration process.
- *Flexibility in data mapping*: The ability to map data between different structures and formats easily, which can help adapt to changing data structures.

A. Extract, Transform, Load

The extract, transform, load process; first coined in the 1970s; is a widely-used data integration pattern that involves extracting data from a source, transforming it into a format required by the final data repository, and then loading it into that repository. This process is commonly used in data warehouses (Muniswamaiah et al., 2019a) and has been the main method of integrating data for decades. However, with the advent of big data and the need for real-time data integration, organizations have begun to explore alternative methods, such as ELT (extract, load, transform), which loads data before making any transformations, and data virtualization, which allows for real-time access to data without the need for replication. Despite its widespread use, ETL has its limitations, such as the need for up-front transformation and the complexity of maintaining multiple ETL processes. As a result, organizations are increasingly looking for more efficient and effective data integration patterns that can handle the high volume and variety of data in today's digital landscape.

In terms of *ease of use* and *maintainability* criteria, the process of extraction, transformation, and loading of data can be complex, and may require specialized technical knowledge to set up and maintain. It may be more complex for business users or those with less technical expertise. ETL can handle large volumes of data, but it can have performance issues when dealing with high-velocity data streams. In term of *error handling*, ETL typically include built-in error handling capabilities, but the level of error handling can vary depending on the specific implementation and vendor. As for the *reusability* and *extensibility*, ETL may be relatively reusable and extensible, as it can be configured to handle different data sources and integration scenarios, but the level of reusability and extensibility can vary depending on the specific implementation and vendor.

B. Enterprise Service Bus

Enterprise Service Bus was originally coined by analysts from Gartner in 2002, as organizations began to adopt service-oriented architecture (SOA) and web services as a means of integrating their systems (Menge, 2007). ESB does not involve moving the data, instead, it uses a message bus to

Table 1: Comparison of the various data integration patterns

	Ease of use	Maintainability	Performance	Error handling	Reusability	Extensibility
ETL	★★	★★	★★★	★★★	★★★	★★★
ESB	★★	★★	★★★	★★★	★★★	★★★
CDC	★★	★★★★	★★★★	★★★★	★★★★	★★★★
DaaS	★★★★	★★★★★	★★★	★★★	★★★★	★★★★
SDI	★★	★★	★★★★★	★★★★	★★★	★★★★
DV	★★★★★	★★★★★	★★★★★	★★★	★★★★	★★★★

facilitate the interactions of applications and services. Applications are connected to the message bus. This allows them to communicate and exchange messages in real-time. Applications in ESB are decoupled, therefore, no need for one application to know about, or depend on other applications (Menge, 2007).
Data integration challenges such as those faced by ETL and ESB patterns can be addressed by integrating a data virtualization layer. While ETL processes are suited for operational scenarios, they are not well-suited for analytical use cases due to their batch-oriented nature and difficulty in maintenance over time. Similarly with ESB, while it is effective in moving away from point-to-point integration and offering real-time interaction between applications, it is not capable of integrating application data for analytical use cases. Data virtualization can provide an effective solution for integrating data from various sources in real-time and delivering analytical use cases.
ESBs can be relatively *easy to use* for IT teams with the appropriate technical expertise. However, they may be more complex for business users or those with less technical expertise. They can also be complex to *maintain*, especially if the ESB process is customized and the data sources are diverse. ESB processes may require frequent updates and maintenance to handle changes in data sources and integration scenarios. ESBs can handle large volumes of data, but they can have *performance* issues when dealing with high-velocity data streams. However, the performance can be improved through the use of different techniques such as parallel processing and optimized data structures. Same as ETL, ESBs may include built-in *error handling* capabilities. They can handle errors during the data integration process, but they may require manual intervention. As for *reusability* and *extensibility*, they are almost the same as with ETL, ESBs can be relatively reusable and extensible.

C. Change Data Capture

Change Data Capture refers to the process of identifying and capturing changes made to a specific data source and propagating these changes to other systems in real-time. This enables real-time data integration and ensures consistency and accuracy of data across different systems. CDC emerged in the early days of database management systems, when the primary means of capturing changes to a database was through database triggers. It is used in several applications such as, database replication, live data monitoring, real-time data warehousing, and for event-driven architectures (Eccles et al., 2010).
CDC is typically implemented using log-based or timestamp-based methods. This process eliminates the need for full refreshes of data, and allows the target system to be updated as soon as a change occurs in the source system. This is particularly useful in scenarios where data is rapidly changing and consistency is crucial (Schmidt et al., 2015).
However, implementing CDC also poses some challenges, such as the complexity of setting up and maintaining CDC processes, and the potential for increased data duplication and data inconsistency if not implemented properly.
CDC can be relatively *easy to use* for IT teams. However, it may be more complex for business users or those with less technical expertise. CDC solutions are typically less complex to *maintain* than ETL and ESB solutions, they may require less frequent updates and maintenance. CDC integration pattern is typically faster and more efficient than ETL and ESB, as they only process changes in the data rather than the entire dataset. It can handle high-velocity data streams, but may have *performance* issues when dealing with very large volumes of data. CDC method has been shown to be more efficient than traditional ETL or ESB methods. This is due to the fact that CDC only captures and records changes in specific sets of data, rather than the entire dataset. As a result, the scope of potential errors is reduced, and the process of identifying and resolving errors is typically faster. Additionally, CDC solutions often include built-in *error handling* capabilities, which further streamlines the process of dealing with errors. CDC solutions can be relatively *reusable* and *extensible*, as they can be configured to handle different data sources and integration scenarios. They can be adapted to different contexts and use cases. As a result, CDC solutions can be expanded and customized to meet the evolving needs of an organization.

D. Data as a Service

Data as a Service is a method of delivering data as a service to different users, applications, and systems. It is a form of real-time data integration that allows organizations to access and use data from various sources, such as databases, cloud storage, or data lakes, as a service, through a set of APIs or web services (Jiang et al., 2012). DaaS emerged in the early days of cloud computing and Software as a Service in the 2000s, when organizations began to offer a variety of software applications as a service over the internet (Wang et al., 2010).
DaaS offers several advantages to organizations when it comes to data integration. It provides a simplified and unified way of accessing data, eliminating the need for data replication and movement. DaaS also allows for flexibility in data usage, as organizations can access more data and more sources as their needs grow, without the need to move or replicate data. However, implementing DaaS also poses certain challenges such as ensuring data security, data governance and data quality, as well as dealing with data latency,

data availability, and data integrity issues.
DaaS can be relatively *easy to use* for business users or IT teams. DaaS solutions are less complex to *maintain* than on-premise solutions, as the vendor is responsible for maintaining the service. DaaS solutions may require less frequent updates and maintenance. They can handle large volumes of data, but they can have *performance* issues when dealing with high-velocity data streams. DaaS solutions are often built on cloud infrastructure, which can provide scalability and performance benefits. However, the performance of DaaS solutions can be affected by the quality of the internet connection. DaaS platforms may include built-in *error handling* capabilities, but the level of error handling may be limited. DaaS solutions can be relatively *reusable* and *extensible*, as they can be configured to handle different data sources and integration scenarios. However, the level of reusability and extensibility can vary depending on the vendor.

E. Streaming Data Integration

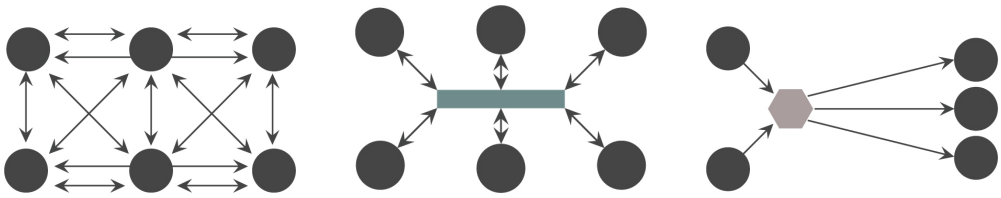
SDI is a real-time data integration pattern that involves the continuous and concurrent collection, processing, and analysis of data as it is generated from various sources (Tatbul, 2010). This method enables organizations to process and analyze data in near real-time, allowing for real-time decision making. It is particularly useful in scenarios where data is rapidly changing and time-sensitive, such as fraud detection (Carcillo et al., 2018), anomaly detection (Zhang et al., 2020), and real-time analytics (Bou et al., 2021). The history of streaming data integration can be traced back to the early days of data integration, when data integration was typically done in batches and on a scheduled basis.
The use of streaming data integration allows organizations to process and analyze data as it is generated, rather than waiting for batches of data to be collected and processed, leading to improved efficiency and better decision-making. Additionally, it enables organizations to scale their data processing and analysis capabilities as the volume and velocity of streaming data can grow rapidly.
However, it is important to note that streaming data integration also poses certain challenges such as ensuring data quality, data security, and data governance, dealing with data latency, data availability and data integrity issues. It also requires specialized skills and expertise, and it can be complex to set up and maintain. Therefore, organizations must carefully consider these challenges and have appropriate measures in place to mitigate them in order to effectively implement streaming data integration.
Streaming data integration can be relatively *easy to use* for IT teams with the appropriate technical expertise. However, it may be more complex for business users. It may be difficult to *maintain*, especially when dealing with customized integration process and diverse data sources. Streaming data integration process may require frequent updates and maintenance to handle changes in data sources. Streaming data integration solutions are designed to handle high-velocity data streams in real-time, they can handle large volumes of data and provide low-latency data processing. They can handle errors during the data integration process, as they typically include built-in error handling capabilities. Streaming data integration solutions can be *reusable*, but not as flexible as DaaS or CDC. As for *extensibility*, they can be relatively extensible, as they can be configured to handle different data sources and integration scenarios.

F. Data Virtualization

Data virtualization is a data integration pattern that allows for the creation of a virtualized data layer that integrates data from a wide variety of structured, semi-structured, and unstructured sources in real-time, without the need for physical replication of data. This virtualized data layer provides a single, unified, and integrated view of data that can be accessed on-demand by various data consumers, such as applications, processes, data scientists, and business users. The data virtualization layer contains only the metadata required to access each of the connected data sources, as well as any governance policies and security rules that may be applied. The use of data virtualization enables faster access to data, reduces data duplication, decreases costs and increases agility in data management.
In terms of *ease of use* and *maintainability*, data virtualization has been demonstrated to be relatively easy to use and maintain, as it abstracts the complexity of the underlying data sources, makes it easy for applications to access and query data, and reduces the need for manual intervention. As for *performance*, data virtualization can provide high performance, as it allows multiple data sources to be queried in parallel and can also cache data for faster access. However, in terms of *error handling*, data virtualization can handle errors during the data integration process, but it may require manual intervention to resolve errors. For *reusability* and *extensibility*, data virtualization can be relatively reusable and extensible, as it can be configured to handle different data sources and integration scenarios.

G. Data Integration from an Architecture Perspective

When examining data integration from an architectural perspective, it is common to encounter three primary models: point-to-point interaction, hub-and-spoke, and publish-subscribe, as illustrated in Fig. 2. These models represent architectural patterns for designing and implementing data integration systems.
The *point-to-point* interaction model is a relatively simple approach that involves connecting each system with every other system with which it needs to exchange data (See Fig. 2a). While this model is relatively straightforward to implement, it can become increasingly complex and difficult to maintain as the number of systems increases.
The *hub-and-spoke* model, on the other hand, centralizes data integration through the use of a central hub that acts as an intermediary between all connected systems (See Fig. 2b). This model allows for greater control and management over data integration processes, but it can also become a bottleneck if not properly implemented and scaled.
The *publish-subscribe* model is a more advanced approach that allows systems to subscribe to specific data events (See Fig. 2c). This enables more flexible and scalable data integration, but it can also be challenging to implement and requires specialized infrastructure.
It is important to consider that each model has its own set of advantages and disadvantages and the choice of architec-



(a) Point-to-point interactions (b) Hub-and-spoke interactions (c) Publish-subscribe interactions

Figure. 2: Architectural patterns for designing data integration systems

ture will depend on the specific requirements and constraints of the organization. Furthermore, best practices and patterns such as data quality, data security, and data governance should be taken into account in order to ensure that the integrated data is accurate, consistent and secure.

H. Disadvantages of Traditional Approaches to Data Integration

Traditional approaches to data integration, such as ETL and ELT methods, have several limitations that can impede their effectiveness in certain use cases.

One of the challenges associated with traditional approaches to data integration is the significant time investment required for data transformation and data mapping. This is due to the lack of reuse of existing transformation specifications, leading to the need to repeatedly implement such specifications. This can impede the development process and ultimately result in a prolonged time to market.

Another limitation of traditional approaches to data integration is the difficulty in maintaining consistency of reports. The synchronization of various implementations of transformation specifications can be challenging, as changes to a specification may not always be reflected consistently across all implementations. This lack of consistency in transformation specifications can lead to inconsistencies in the final reports, which can undermine the integrity of the data and the overall usefulness of the data-driven insights. Additionally, ensuring that all implementations are updated in a timely and consistent manner is a complex task.

The presence of multiple copies of the same data, also known as data duplication, is a prevalent issue in traditional data integration approaches. The duplication of data occurs when data is copied from source systems to data stores. Additionally, data may also be duplicated within each data repository for reasons such as performance optimization. This can result in a large number of copies of the same data being stored across different data architectures, potentially numbering in the tens or even hundreds. This can lead to issues such as data inconsistency, increased storage costs, and difficulties in maintaining data quality and integrity.

Data quality is a critical consideration when implementing data integration solutions. Traditional approaches, such as ETL and ELT methods, can introduce potential data quality issues due to the replication and transformation of data. The more data is copied or transformed, the higher the risk of low data quality. Factors such as data redundancy, inconsistencies, and errors can negatively impact the integrity and accuracy of the integrated data. As a result, it is important to implement measures and best practices to ensure data quality throughout the data integration process.

The adoption of traditional approaches to data integration may result in increased costs associated with development. This can be attributed to the implementation of various technologies, the duplication of data storage and computing resources, and the need for additional maintenance. The lack of sharing or reuse of implementations can contribute to these increased costs, as they may lead to unnecessary duplication of effort.

The complexity of implementing and maintaining traditional data integration approaches can present a significant challenge, particularly as the number and diversity of data sources, as well as integration requirements, increases. This can impede the ability to effectively manage and integrate data, resulting in inefficiency and limitations in scalability.

Traditional data integration approaches, such as ETL and ELT, often rely on physically moving and replicating data, which can result in delays in the availability of integrated data. These approaches can be resource-intensive and may not be optimized for large-scale data integration, leading to inefficiencies in terms of cost and performance. They may face limitations in terms of scalability, and may not be able to effectively accommodate the growth of an organization.

Traditional approaches may not provide adequate controls for data quality, security, and governance, which can lead to inaccurate or inconsistent data. These limitations in governance can have a negative impact on the integrity and reliability of integrated data.

In the next section, we explore the concept of data virtualization as a modern data integration pattern that addresses these challenges.

III. Data Virtualization

Data virtualization is a modern approach to data integration that aims to provide unified access to a diverse range of structured, semi-structured, and unstructured data sources. It utilizes an abstraction layer, referred to as a virtual data layer, to deliver real-time data services (Mousa & Shiratuddin, 2015) to a variety of data consumers, such as applications, processes, and users as illustrated in Fig. 3. Unlike traditional data integration patterns, data virtualization does not copy data but instead creates a view of the integrated data, keeping the source data in its original location (Satio et al., 2016), resulting in reduced costs, minimized data replication, and minimal data latency. It has the ability to replace traditional data integration patterns by reducing the need for replicated data marts and warehouses (Mousa et al., 2014), while also being highly complementary to other data integration patterns such as ETL and ESB. Data consumers query the virtual data layer, which retrieves data from various sources, hiding

the location and implementation of the physical data, as well as the complexity of accessing it (Bogdanov et al., 2020a). The virtual data layer functions as a single data repository and contains only the metadata of each data source, as well as any global instructions such as data governance policies and data security rules.

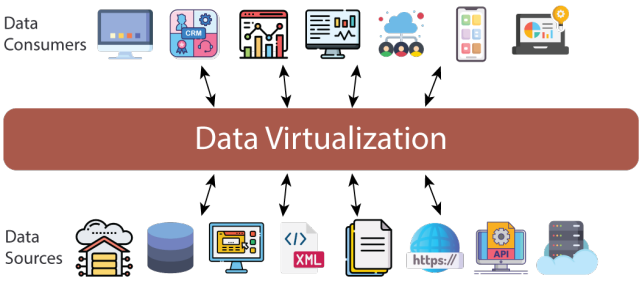


Figure. 3: Data virtualization integrates data from multiple sources and delivers it to different data consumers (Akermi et al., 2023)

In the following sections, we explore how data virtualization complements ESB and ETL processes. Additionally, an overview of data virtualization's core capabilities and features will be provided.

A. *Data Virtualization Complements ETL and ESB processes*

In the context of data integration, the ETL process has been widely adopted for moving data to other repositories, such as data warehouses. However, ETL can present challenges when attempting to integrate data from cloud-based sources (Miller, 2018). In order to overcome these challenges, data virtualization can be utilized as a complement to ETL processes. By integrating a data virtualization layer, it is possible to enable real-time data integration from multiple sources, connect on-premise and cloud-based data sources without the need to move all data to a single repository, unify data across data warehouses and new on-premise or cloud-based data stores, and access data faster than with traditional ETL processes (Guo et al., 2015). Similarly, data virtualization can also assist in the integration of disparate and complex data sources into ESB systems (Miller, 2018).
As shown in Table 2, data integration patterns can be applied to different use cases, and data virtualization may not always be the best approach for a specific problem. Shraideh et al. (Shraideh et al., 2019) developed a structured and systematic decision support system that considers 15 critical success factors to determine whether ETL, data virtualization, or a hybrid solution of both patterns is the most suitable data integration approach.

B. *Faster Data Access and Delivery*

Traditional data integration patterns, such as ETL, involve physically transferring data from multiple sources to multiple locations, such as data stores, databases, data warehouses, data lakes, data lakehouses, and cloud-based repositories. This manual process often results in multiple replications across the network, making the data architecture slower, more complex, and more costly (Gottlieb et al., 2019).
By incorporating a layer of data virtualization, organizations can achieve fast, efficient, and agile data integration solutions (Van der Lans, 2014), helping them to become data-driven. In the context of self-service business intelligence (Lennerholt et al., 2018), this eliminates the need to physically move and aggregate data locally. Instead, business users can connect multiple data sources to the data virtualization layer through pre-built data connectors, also known as adapters.
The data virtualization layer serves as a unified and rapidly accessible source of data for business intelligence reporting and analysis, addressing the high latency commonly associated with traditional data architectures. Moreover, the development of data services is faster, as the unified data layer eliminates the need for developers to connect to each individual data source in different formats and repositories.

C. *Self-Service Analytics*

Self-service analytics empowers business users to independently perform data analysis, freeing the data engineering team to focus on other critical aspects of data architecture. However, the realization of self-service analytics is often hindered by several challenges including widespread distribution of data across databases, data warehouses, cloud, and big data architectures, low data integrity resulting from a lack of a single authoritative source, high latency in accessing the data, and a lack of data lineage, which negatively impacts data quality and raises concerns about its credibility.
Data virtualization can overcome these challenges and makes self-service available to business users (Alagiannis et al., 2012; Chatziantoniou & Kantere, 2021). By this, cost and complexity are reduced, and replications are created only when it is necessary.

D. *Data Virtualization Core Capabilities*

An effective data virtualization layer should have; at least; the following core principles (Bogdanov et al., 2020c, 2020b):

- Pre-built connectors, which facilitate quick and efficient connection, exploration, and extraction of data from a variety of on-premise or cloud-based sources and data types.
- Self-service data services that provide a user-friendly interface, thereby hiding the underlying complexity from data consumers. By decoupling data sources and consumers, data services can be easily created without the need for involvement from the data engineering team.
- The establishment of a single logical data model, which is maintained through the automatic processing of data catalogs containing metadata, data classes, data clusters, and more.
- A unified approach to data governance, which includes a single entry point for data, metadata management, audit logging, security, and monitoring, as well as integration with external data management tools.

Table 2: Data integration use cases and different patterns

Use case	Patterns					
	Data virtualization	**ETL**	**ESB**	**CDC**	**DaaS**	**SDI**
Moving data between data repositories		X		X	X	X
Data unification	X					
Real-time reports and insights	X		X		X	X
Migrating data to the cloud		X		X	X	X
Self-service analytics	X				X	
360 customer view	X					
Data warehouses and data marts		X		X		
Logical data warehouses and virtual data marts	X					
Data warehouse offloading		X			X	
Logical data lakes	X					
Application synchronization		X	X	X		X

- The creation of a unified data layer, which is the centerpiece of the virtual data layer and serves to harmonize, transform, improve quality, and connect data across different data types.
- A universal mechanism for data publishing, which provides users with access to processed data through unified connected services.
- Agile and high-performance operations, achieved through the application of real-time optimizations to create a flexible workload (Earley, 2016; Gottlieb et al., 2019).

E. Privacy and Data Protection

Data virtualization, by its nature, inherently promotes data privacy. This is particularly relevant in the context of General Data Protection Regulation (GDPR), which requires data protection to be implemented "by design". Data virtualization allows organizations to fulfill this requirement by providing a flexible approach to data access and format.
Sources of data are not limited to predefined formats, nor are they required to be accessed through a specific method. Instead, data virtualization offers advanced data protection mechanisms such as data anonymization, immutability of data through refusal of signature, and end-to-end encryption for transmitted transactions. These mechanisms ensure that data remains secure and private, even as it is being accessed and used (Bogdanov et al., 2020c).

F. Data Services

Data virtualization provides a layer of abstraction that enables the creation of data services, simplifying the process of accessing, transforming, and integrating data from multiple sources.
Business-oriented services, such as querying all customers across all repositories or obtaining revenue data from the past five years, are common examples of the services offered by the data virtualization layer. These services enable business users to easily generate reports and insights without the need for technical expertise or the intervention of a data engineer. Operational-oriented services, such as updating a customer's address or email, are also available.
A data service consists of three components: an interface that manages incoming parameters and outgoing results, a logic component that handles data preparation specifications, and a source abstraction that makes the service independent of the underlying data source system. While the creation of these services may seem deceptively straightforward, they are responsible for much of the work involved in data preparation, including transformation, enrichment, joining, federation, synchronization, and historicization.
The complexity of the data services can vary, with some requiring access to multiple systems, others requiring data movement, and still others requiring the execution of machine learning algorithms. Despite this complexity, data virtualization enables it to be hidden from the end user.
The performance of a data service is largely dependent on the performance of the underlying systems, but it is also essential that the data virtualization layer knows the most optimized way to access these systems. This may involve sending requests for sets of data or individual records, depending on the nature of the request. The greatest performance challenge arises when data services need to join data from multiple systems. To address this challenge, the data virtualization layer minimizes the amount of data extraction and network movement by pushing down queries to the connected data sources as much as possible.
The data virtualization layer serves data services to consumers as an integrated system, enabling the creation of a 360-degree customer view, for example, while hiding the complexity of accessing data from disparate repositories. The abstraction provided by the data virtualization layer is achieved through the use of data services.

G. Virtual Tables

Data virtualization plays a crucial role in simplifying the development of service logic. At its core, data virtualization is comprised of virtual tables, which serve as a central component in this process. Virtual tables serve as a blueprint for transforming data sources into the desired format. They can be used to outline the preparation process for data, which can come from various other virtual tables. Additionally, virtual tables can be accessed through a range of interfaces, including REST, OData[1], and JDBC services, among others (see Fig. 4).
One of the key benefits of data virtualization is that it abstracts the location and implementation details of the underlying data sources from virtual table developers. This results in a simplified development experience, as developers

[1]Open Data Protocol

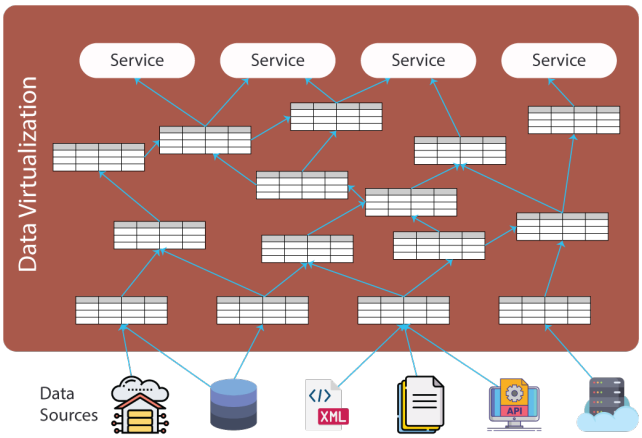


Figure. 4: Data virtualization and virtual tables (Akermi et al., 2023)

can access a single logical database rather than multiple data sources and services, and can interact with a single interface instead of multiple technologies and interfaces.

H. Query Pushdown

The utilization of query pushdown as an optimization strategy is a characteristic of data virtualization. This approach aims to minimize network traffic and maximize the use of the connected data sources by pushing down as much of the data processing as feasible to the underlying data source systems, such as databases, data lakes, or flat files. For instance, executing aggregations directly on a NoSQL database, which is known for its scalability and optimized performance (Alagiannis et al., 2012), is more efficient than carrying out the same operation on the data virtualization layer.
In the following section, we delve into the role of data virtualization in enabling distributed data architectures. Specifically, we focus on two use cases: data fabric and data mesh. Data virtualization provides a virtual layer for accessing data from disparate sources, enabling real-time data access and integration, as well as improved data governance, security, and performance optimization. These capabilities make data virtualization a critical component in the implementation of a data fabric or a data mesh.

IV. Distributed Data Architecture Enabled by Data Virtualization

Henderson et al. define Data architecture as follows ”identifying the data needs of the enterprise (regardless of structure), and designing and maintaining the master blueprints to meet those needs. Using master blueprints to guide data integration, control data assets, and align data investments with business strategy” (Henderson et al., 2017).
Data architecture, as a discipline, encompasses two key areas: data at rest and data in motion. The former pertains to the organization and storage of data within information systems, taking into account fundamental principles such as data clustering and storage formats. The latter, also known as data in transit, pertains to the patterns and principles governing the flow of data between systems. Both areas aim to ensure efficient and effective data management within an organization. The two main approaches to data architectures are monolithic and distributed. In this section, we explore both of these data architectures, their characteristics, and their use cases.

A. Monolithic Data Architectures

A monolithic data architecture is a centralized approach to data management in which all data is stored and managed within a single, large system. Organizations with simple data structures and low data volume often choose this architecture because it offers a straightforward and easily manageable solution.
In a monolithic data architecture, data is typically stored in a single database management system (DBMS), a data warehouse, a data lake, or a data lakehouse, and access to the data is controlled through a centralized application layer. This centralized control enables organizations to easily enforce data governance and data security policies, and monitor data quality.
One advantage of monolithic data architecture, such as data warehouse, is that it provides a single source of truth for all data in the organization. This helps to eliminate data silos and duplicated data, and it ensures that all business users have access to consistent, accurate data.
However, as the volume, velocity, and complexity of data grow, the limitations of a monolithic data architecture gradually emerge. For example, it can be difficult to scale the architecture to meet increasing data needs, it can result in slow performance and hinder data processing, and it can be inflexible, as changes to the architecture require changes to the entire system. Additionally, monolithic data architectures can be challenging to integrate with newer technologies and cloud-based systems, which can limit the organization to take advantage of the latest trends in data management.
Despite these challenges, monolithic data architecture remains a popular approach for organizations with small to medium-sized data requirements, where the benefits of a centralized data architecture outweigh the limitations (Harding, 2022).

B. Distributed Data Architectures

Distributed data architecture is a modern approach to data management that seeks to overcome limitations of monolithic data architecture by distributing data across multiple systems and data repositories. The aim of a distributed data architecture is to provide a more flexible and scalable solution for organizations with complex data structures and large data volumes. This architecture democratizes access to data and enables business users to engage in self-service analytics. Recently, data fabric and data mesh have emerged as new distributed data architectures. However, the role of data virtualization in enabling both remains unclear.

1) Data Fabric

Data Fabric is a data architecture paradigm (Li et al., 2022) and an emerging data management framework, or ”design” (Gartner, n.d.), that enables the seamless access and sharing of diverse data sources within an organization. It provides a flexible, scalable and efficient way of managing data

across multiple platforms, systems, and devices (Kuftinova et al., 2022b; Macías et al., 2023). Data Fabric is designed to handle large-scale data environments and to support the increasing demand for real-time access to data, regardless of its location or format (Kuftinova et al., 2022a). The main purpose of data fabric is to provide a single point of access for data, making it easier for data analysts and data scientists to access, share, and process data (Östberg et al., 2022). This eliminates the need for manual data integration and helps to improve data accuracy, consistency, and security.
The data fabric also enables organizations to leverage existing data sources and infrastructure, reducing the need for additional investment in data management tools and systems. Data Fabric is enabled by a combination of tools and technologies, including ETL processes, data warehouses, master data management (MDM) systems, metadata management systems, and data catalogs. Among these components, data virtualization plays a key role in the data fabric ecosystem.
In conclusion, data fabric plays a crucial role in realizing a distributed data architecture, offering organizations a comprehensive, flexible, and scalable approach to data management. Its ability to provide real-time access to data from disparate systems, without the need for relocating or duplicating data to another repository, is enabled by data virtualization. As a result, data fabric is an essential component for organizations seeking to unlock the full value of their data assets (Kuftinova et al., 2022b).

2) Data Mesh

Data mesh is a novel approach to data management that aims to address the challenges posed by monolithic data architectures. It is characterized by a decentralized and domain-driven approach to data ownership and management (Dehghani, 2019). Traditionally, data management was approached as a centralized function, where data was managed by a central data team (Machado et al., 2022) and distributed to business units as needed. This monolithic data architecture often resulted in challenges such as slow data access, difficulty in maintaining data quality, and a lack of data ownership among business units.
Data mesh, as defined by Dehghani, seeks to resolve these challenges by distributing data ownership across different data domains within an organization. Each data domain is responsible for managing and packaging its specific data as a product for distribution throughout the organization (Dehghani, 2019). This approach enables business units to engage in self-service analytics and makes data more accessible and valuable to the entire organization (Dehghani, 2022).
However, a successful data mesh requires seamless interoperability between data domains, to prevent fragmentation, duplication, and inconsistencies. Interoperability can be achieved through the use of a universal interoperability layer, which provides data standards and governance protocols (Dehghani, 2020).
In conclusion, data mesh offers a novel and decentralized approach to data management that can help organizations address the challenges posed by traditional monolithic data architectures. By distributing data ownership and enabling interoperability, it can make data more accessible, valuable, and actionable to the entire organization.

C. The Role of Data Virtualization in Distributed Data Architectures

Data virtualization is a technology that enables real-time access to data from disparate sources, without having to physically move the source data to a centralized repository. This technology creates a virtual data layer that provides access to data from multiple sources, and it is typically used in the context of distributed data architectures such as data fabric and data mesh (Biggenden, 2022).
In a data fabric, data virtualization plays a crucial role in facilitating access and sharing of disparate data sources. Data virtualization enables data fabric by providing real-time access to data across multiple systems, without having to first move the source data to a new repository. This allows for a seamless access to data, without the need for data consumers, e.g. data analysts and business users, to know where the data is stored. Data virtualization provides the necessary interoperability, governance, and security for data fabric.
In a data mesh architecture, data virtualization enables the seamless interoperability of data domains, providing a universal interoperability layer that provides data standards and governance protocols. Data virtualization provides a virtual data-access layer between data sources and domain-specific data consumers, and helps establish the foundation for data mesh configurations. This technology enables data consumers to gain access to only the data they need, when they need it, reducing the need for data replication.
Data virtualization provides advanced performance optimization and self-service search and discovery using data catalogs, and these features are critical for distributed data architectures like data fabric and data mesh. In summary, data virtualization plays a crucial role in enabling distributed data architectures, providing real-time access to data, reducing data replication, and improving data governance and security.

V. Conclusions and Future Work

Data virtualization has emerged as a decoupling technology that offers a number of key features and capabilities that address many of the challenges associated with traditional data integration approaches. By providing a unified view of data, regardless of its source or format, data virtualization increases the efficiency of data operations, minimizes replications, reduces complexity and cost, and enables faster and more flexible access to data.
One of the major benefits of data virtualization is its ability to enable real-time data delivery and self-service analytics, which empowers different users to access and manipulate data without the intervention of the data engineering team. This shifts the focus from data storage to data usage and from moving data to connecting data. In this model, data is not moved from the source system to the target system, but rather the target system can request data on demand and use the services offered by the source system to manipulate it.
Data virtualization is also playing a key role in enabling new, distributed data architectures, such as data fabric and data mesh. These architectures allow for the creation of a highly flexible, scalable, and decentralized data infrastructure that can accommodate the growing volume and variety of data

that organizations must manage and analyze.

Data virtualization offering organizations a powerful and flexible solution for effectively leveraging and unlocking the value of their data. With its ability to support real-time data delivery, self-service analytics, and distributed data architectures, data virtualization is poised to play a critical role in driving the next information revolution and facilitating the creation of new and innovative data-driven solutions.

Future work in the area of data virtualization could include the integration of active metadata, which involves the use of dynamic metadata that automatically updates based on changes in the underlying data sources.

The impact of data virtualization on data privacy and security could also be studied in depth to further strengthen its role in modern data management.

Future work could also include exploring the integration of data virtualization with modern data stacks. Research in this area would also help organizations to better understand the evolving requirements of modern data management, and how data virtualization can play a critical role in meeting these demands, in order to stay ahead of the curve in a rapidly changing data landscape.

Finally, continued exploration of the use of data virtualization in distributed data architectures, such as data fabric and data mesh, could yield valuable insights into how these approaches can be used to support the efficient and effective management and utilization of data at scale.

References

Akermi, M., Hadj Taieb, M. A., & Ben Aouicha, M. (2023). Data virtualization layer key role in recent analytical data architectures. *Intelligent Systems Design and Applications: 22nd International Conference on Intelligent Systems Design and Applications (ISDA 2022) Held December 12-14, 2022*. https://doi.org/10.1007/978-3-031-27440-4_153

Alagiannis, I., Borovica, R., Branco, M., Idreos, S., & Ailamaki, A. (2012). Nodb: Efficient query execution on raw data files. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 241–252.

Biggenden, L. (2022, August). Data fabric, data mesh, and data virtualisation. https://www.nephostechnologies.com/blog/data-fabric-data-mesh-and-data-virtualisation/

Bogdanov, A., Degtyarev, A., Shchegoleva, N., & Khvatov, V. (2020a). On the way from virtual computing to virtual data processing. *CEUR Workshop Proceedings*, 25–30.

Bogdanov, A., Degtyarev, A., Shchegoleva, N., Khvatov, V., & Korkhov, V. (2020b). Evolving principles of big data virtualization. *Computational Science and Its Applications – ICCSA 2020*, 67–81.

Bogdanov, A., Degtyarev, A., Shchegoleva, N., Korkhov, V., & Khvatov, V. (2020c). Big data virtualization: Why and how? *CEUR Workshop Proceedings (2679)*, 11–21.

Bou, S., Kitagawa, H., & Amagasa, T. (2021). CPiX: Real-time analytics over out-of-order data streams by incremental sliding-window aggregation. *IEEE Transactions on Knowledge and Data Engineering*, *34*(11), 5239–5250.

Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF: A scalable framework for streaming credit card fraud detection with spark. *Information Fusion*, *41*, 182–194.

Chatziantoniou, D., & Kantere, V. (2021). DataMingler: A novel approach to data virtualization. *Proceedings of the 2021 International Conference on Management of Data*, 2681–2685.

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

Dehghani, Z. (2020, March). Data mesh principles and logical architecture. https://martinfowler.com/articles/data-mesh-principles.html

Dehghani, Z. (2022, April). *Data mesh* (1st ed.). O'Reilly Media, Inc.

Doan, A., Halevy, A., & Ives, Z. (2012). *Principles of data integration*. Elsevier.

Earley, S. (2016). Data virtualization and digital agility. *IT Professional*, *18*(5), 70–72.

Eccles, M. J., Evans, D. J., & Beaumont, A. J. (2010). True real-time change data capture with web service database encapsulation. *2010 6th World Congress on Services*, 128–131.

Gartner. (n.d.). Definition of data fabric - gartner information technology glossary. https://www.gartner.com/en/information-technology/glossary/data-fabric

Gottlieb, M., Shraideh, M., Fuhrmann, I., Böhm, M., & Krcmar, H. (2019). Critical success factors for data virtualization: A literature review. *The ISC International Journal of Information Security*, *11*(3), 131–137.

Guo, S.-S., Yuan, Z.-M., Sun, A.-B., & Yue, Q. (2015). A new ETL approach based on data virtualization. *Journal of Computer Science and Technology*, *30*(2), 311–323.

Harding, C. (2022, August). Data integration – choosing the right approach. https://blog.opengroup.org/2022/08/09/data-integration-choosing-the-right-approach/

Henderson, D., Earley, S., & Sebastian-Coleman, L. (Eds.). (2017). *DAMA-DMBOK: Data management body of knowledge* (2nd ed.). Technics Publications.

Jiang, N., Xu, L., De Vrieze, P., Lim, M.-G., & Jarabo, O. (2012). A cloud based data integration framework. *Collaborative Networks in the Internet of Services*, 177–185.

Kuftinova, N. G., Maksimychev, O. I., Ostroukh, A. V., Volosova, A. V., & Matukhina, E. N. (2022a). Data fabric as an effective method of data management in traffic and road systems. *2022 Systems of Signals Generating and Processing in the Field of on Board Communications*, 1–4.

Kuftinova, N. G., Ostroukh, A. V., Filippova, N. A., Gaevskii, V. V., & Podgorny, A. V. (2022b). Integration of scalable IT architectures on the basis of data

fabric technology. *Russian Engineering Research*, *42*(11), 1199–1202.

Lennerholt, C., Van Laere, J., & Söderström, E. (2018). Implementation challenges of self service business intelligence: A literature review. *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS)*, *51*, 5055–5063.

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

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

Macías, A., Muñoz, D., Navarro, E., & González, P. (2023). Digital twins-based data fabric architecture to enhance data management in intelligent healthcare ecosystems. *Proceedings of the International Conference on Ubiquitous Computing & Ambient Intelligence (UCAmI 2022)*, 38–49.

Menge, F. (2007). Enterprise service bus. *Free and open source software conference*, *2*, 1–6.

Miller, L. C. (2018). *Data virtualization for dummies, denodo special edition*. John Wiley & Sons, Ltd.

Mousa, A. H., & Shiratuddin, N. (2015). Data warehouse and data virtualization comparative study. *2015 International Conference on Developments of E-Systems Engineering (DeSE)*, 369–372.

Mousa, A. H., Shiratuddin, N., & Bakar, M. S. A. (2014). Virtual data mart for measuring organizational achievement using data virtualization technique (KPIVDM). *Jurnal Teknologi*, *68*(3).

Muniswamaiah, M., Agerwala, T., & Tappert, C. (2019a). Data virtualization for analytics and business intelligence in big data. *CS & IT Conference Proceedings*, 297–302.

Muniswamaiah, M., Agerwala, T., & Tappert, C. (2019b). Data virtualization for decision making in big data. *International Journal of Software Engineering & Applications*, *10*, 45–53.

Naeem, M., Jamal, T., Diaz-Martinez, J., Butt, S. A., Montesano, N., Tariq, M. I., De-la-Hoz-Franco, E., & De-La-Hoz-Valdiris, E. (2022). Trends and future perspective challenges in big data. *Advances in Intelligent Data Analysis and Applications*, 309–325.

Östberg, P.-O., Vyhmeister, E., Castañé, G. G., Meyers, B., & Van Noten, J. (2022). Domain models and data modeling as drivers for data management: The ASSISTANT data fabric approach. *IFAC-PapersOnLine*, *55*(10), 19–24.

Satio, K., Maita, N., Watanabe, Y., & Kobayashi, A. (2016). Data virtualization for data source integration. *IEICE Technical Report*, *116*(137), 37–41.

Schmidt, F. M., Geyer, C., Schaeffer-Filho, A., DeBloch, S., & Hu, Y. (2015). Change data capture in NoSQL databases: A functional and performance comparison. *2015 IEEE Symposium on Computers and Communication (ISCC)*, 562–567.

Shraideh, M., Gottlieb, M., Fuhrmann, I., Kienegger, H., Böhm, M., & Krcmar, H. (2019). Decision support for data virtualization based on fifteen critical success factors: A methodology. *MWAIS 2019 Proceedings*.

Tatbul, N. (2010). Streaming data integration: Challenges and opportunities. *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, 155–158.

Van der Lans, R. F. (2012). *Data virtualization for business intelligence systems: Revolutionizing data integration for data warehouses*. Elsevier, Morgan Kaufmann.

Van der Lans, R. F. (2014). *Creating an agile data integration platform using data virtualization* (Technical Whitepaper). R20/Consultancy.

Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., & Fu, C. (2010). Cloud computing: A perspective study. *New Generation Computing*, *28*(2), 137–146.

Zhang, M., Guo, J., Li, X., & Jin, R. (2020). Data-driven anomaly detection approach for time-series streaming data. *Sensors*, *20*(19).

Author Biographies

Montasser AKERMI received his B.Sc. and M.Sc. degrees in Computer Science from the University of Sfax, Tunisia, in 2020 and 2022, respectively. He is currently pursuing his PhD in data management area focusing on data architecture and metadata management. He is now serving the Faculty of Sciences of Sfax, University of Sfax, Tunisia as an adjunct professor in Computer Science and Communications Department. His current research interests include data management, data architecture, data analytics, and big data. He is member of the Data Engineering and Semantics Research Unit at the Faculty of Sciences of Sfax.

Mohamed Ali HADJ TAIEB received the MSc from the Faculty of Sciences of Sfax and Master degree from the National School of the Engineers of Sfax, in Tunisia. He obtained the PhD degree at the University of Sfax in December 2014. He is assistant professor at the Faculty of Sciences of Sfax, Tunisia. His research interests include semantic similarity, semantic relatedness, knowledge representation, Big Data, social media, data management systems and graph embedding. He is member of the Data Engineering and Semantics Research Unit at the Faculty of Sciences of Sfax.

Mohamed BEN AOUICHA received the MSc from the Faculty of Sciences of Sfax, Master degree from the National School of the Engineers of Sfax in Tunisia. He obtained the doctorate degree from the University Paul Sabatier of Toulouse (France) / National School of the Engineers at the University of Sfax in January 2009, he obtained HDR (Habilitation for Supervising Research activities) diploma in computer systems engineering from University of Sfax. Currently, he is the head and the founder of the Data Engineering and Semantics Research Unit at the University of Sfax. He is the founder of a datacenter in the Faculty of Sciences of Sfax

and Chief Data Officer at the University of Sfax.