



Department of Management Science & Technology

MSc in Business Analytics

**« Introducing a novel model for Data Portability in
Heterogeneous Data Environments »**

By

Lida Vratsanou

Student ID Number: p2822207

Name of Supervisor: Damianos Chatziantoniou

March 2025

Athens, Greece



Abstract

Modern data ecosystems are characterized by high complexity, rendering data portability difficult, a foundational principle of regulations such as the GDPR. Traditional data management approaches like data warehousing and ETL pipelines cannot provide the flexibility required for frictionless data transfers, user control, and interoperability. This thesis presents Data Virtual Machines (DVMs) as a new graph-based conceptual model that enables efficient, scalable, and user-oriented data portability for heterogeneous data.

The research begins with the exploration of the technical and non-technical data portability challenges such as usability, interoperability, extensibility, scalability, regulatory compliance, and data security. It then examines related work on personal data control, integration of data, virtualization, and sector-specific frameworks and identifies gaps in aspects covered by DVMs. The contribution of the thesis is the presentation of DVMs, with their structure, query language, and key ideas as a practicable solution for data portability. DVMs follow a data-driven and flexible modeling approach compared to traditional rigid schemas. They allow schema reorientation and query optimization in order to achieve any-entity view and model polymorphism.

The DataMingler tool is a significant practical contribution and through a real-life use case we will illustrate its feasibility for data portability for the financial sector. Its multiple strengths include the ability to derive the schema from the data, visually represent queries and effectively integrate data, through a simple interface suitable for both technical and non-technical users. The thesis thus illustrates how DVMs simplify data extraction, transformation, and transfer, thus making data more accessible, governed, and compliant.

This thesis positions DVMs as a scalable, flexible, and regulatory compliant paradigm for future-proof data portability solutions, bridging the gap between technical sophistication and end-user empowerment.

Keywords: Data Portability, Data Virtual Machines, Data Virtualization, Heterogeneous Data Integration, Query Processing, Regulatory Compliance, DataMingler



Table of Contents

1. Introduction	4
1.1 Introduction to Data Portability.....	4
1.2 Data Portability Challenges.....	4
1.3 Objective of Thesis.....	5
2. Technical and Non-technical requirements.....	6
2.1 Technical Requirements for Data Portability	6
2.1.1 Understandability and Usability.....	7
2.1.2 Extensibility and Scalability	14
2.1.3 Linkability and Interoperability	21
2.2 Non-Technical Requirements for Data Portability.....	26
2.2.1 Regulatory Compliance and Data Governance.....	26
2.2.2 Data Security and Privacy.....	30
3. Related Work	33
3.1 Personal Data Control	33
3.2 Data integration and Transfer.....	37
3.3 Data Integration and Virtualization	38
3.4 Sector-specific frameworks.....	40
4. The Data Virtual Machine (DVM).....	43
4.1 Theoretical foundation	43
4.2 Structure.....	45
4.3 Key Concepts.....	47
4.4 Query Language.....	49
4.4.1 Operators	50
4.4.2 Query tree	56
4.4.3 Query evaluation and optimization	58
4.5 Any-Entity View and Model Polymorphism.....	60
4.6 Practical Implementation: DataMingler Tool	61
4.7 Conclusion.....	65
5. DVM: Data Portability Applications.....	66
6. DVM: Use Case	69
6.1 Introduction to data management for regulatory reporting.....	69
6.2 Integrating Data in Data Mingler	73
6.3 Executing Queries in Data Mingler.....	78
6.4 Use Case results.....	84
6.5 Future Direction.....	85
7. Conclusion.....	87
References	89



List of Figures

Figure 1: DVM-modeling vs traditional ER-modeling. Reproduced from [101]	45
Figure 2: A customer entity with several attributes. Reproduced from [101]	46
Figure 3: A simple DVM example. Reproduced from [101]	47
Figure 4: Key-list structures to represent edges of DVMs. Reproduced from [101].....	48
Figure 5: Aggregation Function example	51
Figure 6: Filtering Function example	52
Figure 7: Mapping Function example.....	53
Figure 8: Joining two consecutive edges to one. Reproduced from [102]	54
Figure 9: RollupJoin example.....	54
Figure 10: ThetaCombine example	56
Figure 11: An example of a dataframe query. Reproduced from [102]	57
Figure 12: Breakdown of dataframe query	58
Figure 13: Creating JSON documents for different DVM's nodes, conceived as entities. Reproduced from [100]	61
Figure 14: DataMingler's Architecture. Reproduced from [99].....	62
Figure 15: Data Canvas: Managing DVMs. Reproduced from [99].....	63
Figure 16: QueryBuilder: Dataframe queries. Reproduced from [99].....	64
Figure 17: Addition of Database resource	74
Figure 18: Addition of Excel resource	74
Figure 19: Addition of CSV resource	74
Figure 20: Available resources in Data Canvas	75
Figure 21: Configuration of CSV	75
Figure 22: Use Case DVM.....	77
Figure 23: Use Case Query Builder.....	78
Figure 24: Use Case Query 1	80
Figure 25: Use Case Query 1 Tree.....	80
Figure 26: Use Case Query 1 Transformation.....	81
Figure 27: Use Case Query 2 Tree.....	82
Figure 28: Use Case Query 3 Tree.....	83



1. Introduction

1.1 Introduction to Data Portability

Data portability is one of the pivotal features of modern data governance. The aim is to empower individuals by ensuring control over their personal data while also promoting competition in the digital markets. The General Data Protection Regulation (GDPR) formally establishes this right. According to Article 20, users can request their data in a "structured, commonly used and machine-readable format" and directly transmit between controllers. The core philosophy of data portability is that individuals, rather than organizations, are in control of personal data. Any unnecessary technical or procedural barriers should be lifted from data utilization and transfer.

Data portability is a necessity for user empowerment. The ability to move personal data promotes user autonomy and transparency of data handling. With the vendor lock-in effect reduced, consumers are able to make informed decisions on where to store data and how, without the risk of losing information. It also closely aligns with broader trends about digital rights, protection of the consumer and data democratization. As the true value of personal data is being recognized, businesses must adapt to the new expectations.

1.2 Data Portability Challenges

Data portability offers clear benefits, but its implementation brings forth challenges for both users and businesses alike. The requirements of the two are different, with individuals wishing to exercise their right in a simple way and businesses trying to ensure regulatory compliance. Heterogeneous data environments are particularly complicated due to high diversity in data formats and systems. Issues of interoperability, technical limitation and security are expected to occur, rendering current solutions insufficient.

To achieve data portability, organizations should have systems capable of efficient, secure and scalable data sharing. Such a shift benefits not only consumers but also pushes organizations toward more user-friendly data management practices. The process of finding efficient and secure ways for data management and sharing paves the way for



innovation. However, investment in new technologies can be costly and complicated. Additionally, easy data transfers free users from persistent monopolistic practices in digital markets. As a result, competition is enhanced, and users can choose a service suitable for their needs.

1.3 Objective of Thesis

The main goal of this thesis is to analyze the challenges that data portability entails and examine how Data Virtual Machines (DVM) can make its implementation feasible. Such challenges will be investigated thoroughly, both technical as well as non-technical, along with proposal of potential solutions. Focusing on the DVM framework, a graph-based conceptual data model, this research aims to examine its usage towards data portability. The thesis will focus on presenting a case study of the functionality of DVM and how it is applied in a real-life example to critically evaluate its efficiency.

This thesis also seeks to bridge the gap between theoretical principles and practical implementation by identifying key areas for improvement and innovation. By exploring the DVM's extensibility, scalability, and user-centric design, the thesis aims to contribute toward a robust, adaptive model of data portability. Finally, the thesis hopes to yield findings useful for crafting solutions that would be empowering for users, improve the level of data governance, and align with dynamic technological and regulatory changes.



2. Technical and Non-technical requirements

Data portability, especially within environments characterized by high heterogeneity, presents multiple challenges, both technical and non-technical. Nowadays, organizations deal with increasingly varied data sources, ranging from traditional databases to flat files and unstructured data. As a result, ensuring seamless data transfer between systems has become considerably more complicated. Fast and efficient transfer of data is not just a preference, but rather a critical requirement for businesses using multiple platforms for sharing data across various departments or even organizations.

From the technical perspective, any data portability solution should be able to support different data formats and the increasingly growing data volumes. At the same time, flexibility is key in order to handle additional data sources without significant reconfiguration. In parallel, it must ensure that both technical and non-technical users can easily understand and interact with the data. Beyond the technical requirements, non-technical factors are just as important. Data security and privacy have become of utmost importance with regulations such as the GDPR. Organizations should also consider user consent and data ownership, besides being compliant with the applicable legal frameworks.

This section summarizes the major technical and nontechnical requirements and challenges encompassing data portability frameworks. It also proposes solutions for the realization of effective data portability within modern, data-driven environments.

2.1 Technical Requirements for Data Portability

The goal is to develop a data model that efficiently enables data portability. To achieve that, it must fulfill a number of technical and functional requirements, apart from overcoming major implementation challenges.

It should be understandable and usable, hence non-experts can intuitively interact with the data. The data model should be extensible, easily accommodating additional data types and functionalities, and scalable to support enlarged volumes of data without



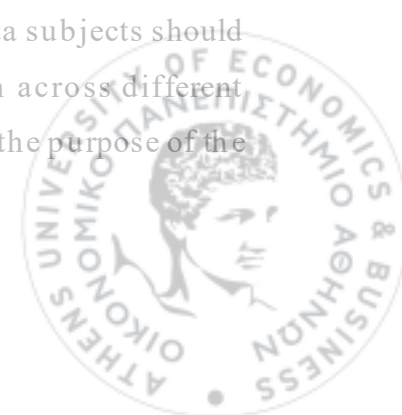
performance degradation. Additionally, it should be linkable and interoperable, enabling smooth data exchanges between different systems. For that, it should support distributed processing to handle voluminous data smoothly with no loss of system performance and reliability.

Building a data model for data portability is highly technical. It involves not only addressing the functional requirements identified earlier but also finding ways to make each of these aspects work practically. Each of the requirements addresses directly the central aim: creating a system where data can be ported across platforms without problems in a secure and efficient way, both for the user and the system.

2.1.1 Understandability and Usability

Data portability frameworks are necessary tools for empowering individuals with possession and control over their personal data in the digital ecosystem. Data portability should not be limited to technical users, all individuals should be able to access, manage and transfer their personal data with ease. A usable system is intuitive, efficient and user-friendly. An understandable system allows users to view and act upon their data meaningfully. However, application mechanisms for these frameworks have faced great challenges in reaching usability and understandability by non-experts. The General Data Protection Regulation does indeed support the right to data portability; yet such implementations often entail technical complexities, disjointed datasets, and unintuitive interfaces. Even the most sophisticated technical solution risks alienating non-technical users. As a result, the right to data portability becomes impracticable for many. Bridging the gap between technical design and user accessibility is fundamental for functional and inclusive frameworks. The following section discusses the user barriers, while proposing solutions focused on simplicity, clarity and usability.

The General Data Protection Regulation (GDPR) emphasizes that data subjects should have control over their personal data and be able to transmit them across different platforms or service providers without facing technical barriers. While the purpose of the

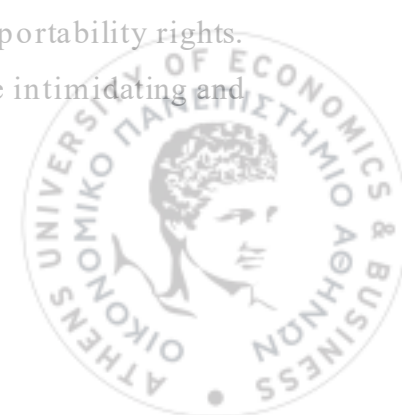


right is to empower users, it also presents quite a challenge: data models are required to be technically robust, yet intuitive enough for non-technical users. [1]

In practice, the implementation of data portability under GDPR is far from meeting its expectations. As highlighted in studies examining the application of the right to data portability, current systems repeatedly fall short of the expectations of the end-user. Unavailability of data is the obvious reason, but also a lack of intuitive interfaces. Responses to portability requests often use formats and procedures which are inaccessible or incomprehensible to consumers. This presents a more profound problem: even when data is technically in compliance with regulatory requirements, the lack of good tools and interfaces makes it practically impossible for users to engage meaningfully with it. Clear, simple, and consistent presentation of data can help bridge the gap between technical compliance and user empowerment. [2][3]

Challenges in Understandability and Usability

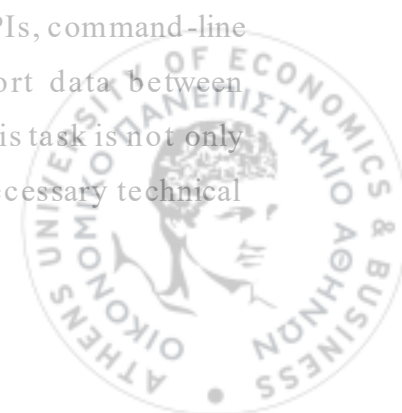
Most of the frameworks for data portability have been developed for technical users, imposing huge barriers to the majority of the users because of their lack of technical expertise. Such frameworks depend on machine readable data structures and formats like JSON, XML, or CSV, which are highly technical and require specific tools or knowledge for proper interpretation. Non-technical users who happen upon such formats are usually overwhelmed, having no ways of intuitively interacting with the data. With minimal knowledge, a simple XML can easily be read as it contains only limited information. However, the more information it contains, such as increased data volume, attributes or elements, the less readable and consequentially understandable such a file can be. It is therefore evident that machine-readability is not the same as user accessibility. It will be impossible for a user to extract any meaningful information from a machine-readable file with thousands of entries with cryptic field names and values, unless they are guided or have technical skills. This technical complexity introduces an accessibility gap, which, in fact, excludes non-technical users from fully exercising their data portability rights. The result is that what should be a user-centered process has become intimidating and inaccessible. [2][3][4][5]



Even when exports are successful, the contents inside could be non-contextualized. Let us take as example a CSV file with the following column names: "TRANS_ID", "SKU", and "QTY". From a first review, no additional information on what the fields represent, how they are structured or what they can be used for is available. Even the same column between providers may contain different information, such as metrics being expressed in actual numbers or percentages. Without sufficient context, users cannot understand their data or know how it could be used to meet their needs. Lack of clarity renders data portability frameworks useless, since data is technically available but practically incomprehensible. Clear explanations, including metadata enhancement, describing the purpose and relationships of the data are necessary to bridge this gap and make exported data actionable for users. [3][4]

Additionally, most of the interfaces that data portability frameworks offer lack simplicity and clarity and thus are not workable. User Interface (UI) design is the first thing that a user notices when first utilizing a product. It refers to its visual design, such as the colors and fonts used, or the position of the buttons and features. Next up, we have User Experience (UX), which refers to the feelings of the user while navigating through the product. In many systems, over-complicated UIs or badly designed UXs seem to puzzle users instead of facilitating them. That could be a result of multiple but confusing ways to access information, lack of clarity in data exploration and manipulation, or difficulty in finding the needed functionalities. The user might need to make various selections based on unclear options and ambiguous messages, like "structured data export" or "schema validation error". This level of difficulty leaves users confused about their actions. Most of the frameworks likewise don't give any on-point guidance, leaving users guessing what they're supposed to do. This does not only discourage users from using data portability features but also raises the likelihood of errors, including exporting incomplete or wrong datasets. [6][7][8]

Other characteristics of data portability frameworks are complicated APIs, command-line tools, or data transformation scripts. Without ways to directly port data between providers, the user is tasked with performing this action indirectly. This task is not only lingering but can also prove highly technical. Most users lack the necessary technical



knowledge required and the programming skills to effectively interact with the data. Even a seemingly simple task like transferring a contact list between systems might require writing code to do necessary data transformations. This challenge is further exacerbated by the unavailability of no-code or low-code platforms. With the introduction of intuitive tools, such as drag-and-drop interfaces, prebuilt templates, or guided workflows, a user can manipulate and port their data without barriers. In their absence, users continue to depend either on third-party services or technical professionals, which compromises their independence and denies inclusivity to data portability rights. [6][7][9][10]

Moreover, the data could be fragmented into multiple files, or even formats that do not give any comprehensive view whatsoever. For instance, exporting social media data will most likely result to receipt of different files for profile information, posts, photos, and interactions. The format and structure of these files can also vary heavily. The user is left to piece this fragmentation together manually, a process which is laborious and liable to induce errors. Fragmentation renders reuse of data more complicated and interpretation even harder. For instance, a user who wants to move data to another service might be forced to combine or otherwise restructure the fragmented files into a form acceptable to the new system. Even data exploration is not possible without the proper consolidation tool, as insights cannot be derived without first understanding and establishing connections between datasets. Not having a uniform and user-friendly way of presenting the data adds an unnecessary layer of complexity that is likely to turn users away from using data portability frameworks. [10]

Without filtering or summarization, the sheer volume of data produced by modern digital services is too great for users to practically review. For instance, exporting all the data from a fitness app may yield a dataset of thousands of daily activity logs, with multiple data points each. Although the information included is complete, it can overwhelm a user looking to review information for specific insights or import relevant data into another service. Without the functionality of filtering, summarizing, or prioritizing the data that users want to export, portability frameworks will lead to an overwhelming experience. Exported data can only be practically utilized when users are able to find the needed



information. Having the option to clearly filter out and summarize data is a necessary step toward making data portability feasible and user-friendly. [6][10]

The above challenges presented, preventing understandability and usability in data portability frameworks, point to a gap between technical design and meeting users' needs. Inaccessibly formatted, complicated interfaces, and fragmented data presentation exclude non-technical users, while overwhelming volumes of data and a lack of intuitive handling tools further worsen the challenge. These challenges can only be confronted if the frameworks emphasize design principles such as simplicity in the interfaces, contextual explanation, unification of data presentation, and low- or no-code solutions, presented in detail in the next section. Such frameworks will only live to their promise of empowering each and every user irrespective of their technical capabilities when that gap between technical complexity and user accessibility is bridged.

Frameworks and Tools for Enhancing Usability and Understandability

The challenges of understandability and usability in data portability frameworks previously outlined require comprehensive solutions, to ensure that data is always accessible and interpretable. Users of any level of technical expertise should be able to engage with their data and practice their portability rights without undue friction. This can only be achieved by focusing on user empowerment and simplification of complicated technical processes. The interaction of users with complex data systems has changed significantly with the emergence of new tools, such as graph-based data visualizations and low-code platforms. These greatly reduce the complexities of data management for non-technical users.

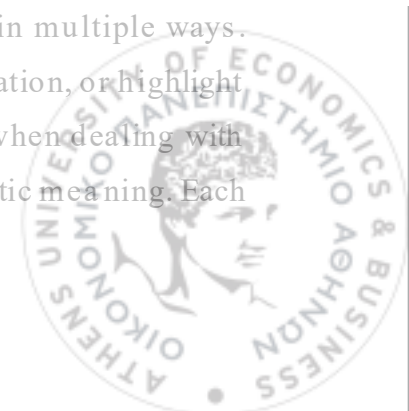
The goal is to eliminate complicated, technical interfaces and replace them with intuitive, visually oriented tools that will facilitate seamless user interaction. The process of moving data between services should be as easy as dragging a file into an upload area. Data portability interfaces can allow users to move their data between platforms with ease by utilizing drag-and-drop features. Additionally, visual dashboards are a great way of data presentation. These enable the user to explore the dataset through charts, tables, and graphs. For example, instead of the user navigating through raw banking data files, they can view their transactions on a timeline. The navigation can be made even simpler



through data filtering, cross-highlighting and drilling. Such dashboards allow the creation of views customizable to the user's requirements. For even better support, a step-by-step guide on how to export, transform, and import data should be available to the user, further reducing user confusion. [11][12][13]

To reduce complexity and improve usability, fragmented datasets should be aggregated into a cohesive, unified view. There are multiple tools like Microsoft BI, Tableau or Qlik, that consolidate data derived from various sources into a single export. For instance, a bank could provide account details, transaction history and credit information into an exportable, browsable package instead of three. Unified data are easier for users to navigate. Specifically visual representations, such as graphs, clearly display the relationships between the data. For example, how the transaction pattern relates to spending behavior or credit recommendations. The exported file should be in an interoperable format, such as JSON-LD or XML. These tools also include features such as filtering and summarization that help users export only the data that matters to them. Let us use again the example of a fitness app which is tracking multiple activities since its initial download to a device. Users might be interested in a specific time period or type of activity, like running or cycling, and therefore a complete download of all data from the app is not useful to them. Additionally, especially when dealing with large datasets, summarizing them into key insights makes data exports more manageable and meaningful for users. For instance, "Top 10 most frequently contacted people" for social media or "Most active hours" for app usage. [10][14][15][16][17]

The most common data representation is in tables, meaning columns and rows, however, graphical representations are becoming more prominent. Frameworks like Neo4j present the data entities and the relationships between them visually. Entities are represented as nodes, which can be anything from people and organizations to data points, and the connections among them are edges, which can be collaborations, dependencies, or interactions. Users have the ability to interact with the visualization in multiple ways. They can focus on specific nodes or edges, filter out unrelated information, or highlight particular relationships. This dynamic nature can prove very helpful when dealing with large or complex datasets. Additionally, they are able to convey semantic meaning. Each

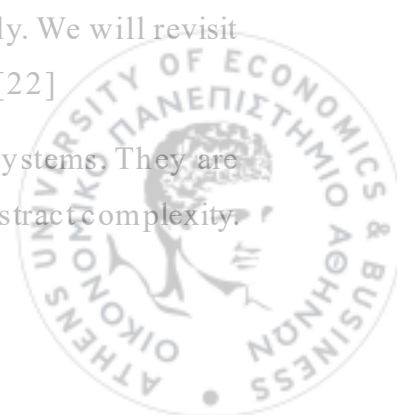


relationship between the nodes might have additional metadata that carries context. For instance, "Is employed by" is different than "Is working with", making it possible to explore various connections. The clear representation of interconnected data will further empower users to understand and manage their data portability rights. These visualizations also improve usability by letting non-technical users intuitively interact with their data. [12][18]

Low-code and No-code platforms are revolutionizing data management by allowing easier data manipulation. These platforms give the ability to perform complex data operations without needing much technical knowledge. In Low-Code platforms, users can create workflows with minimal programming. To simplify operations, they use visual tools like drag-and-drop or prebuilt templates. No-Code platforms further enhance this idea, by requiring no coding skills at all. They rely solely on visual interfaces. As they both excel at integrating with diverse systems, they allow users to work across various data sources. Specifically, when it comes to data portability, the user can import, manipulate and export data with ease. These platforms also help solve one of the major challenges in data portability: allowing users to maintain control over their data. By offering intuitive tools to handle the format, relationships, and transfers involved, low-code and no-code platforms align with the demands on usability and understandability of data portability frameworks. They illustrate that it is possible to abstract technical complexity without losing functionality, so that users can be effective and confident executors of their data rights. [19][20]

Additional context can be embedded in the data through metadata. Metadata helps users to understand the structure, origin, and meaning of their data. For example, some metadata fields are "date_created", "source", or "format_type". It carries semantic meaning. Metadata is only useful when standardized and human readable. Initiatives like Dublin Core or Schema.org support standardized metadata, for consistency and compatibility across platforms. Since data structures change dynamically. We will revisit the need for metadata enrichment in subsequent sections. [3][21][22]

These tools pave the way to accessible and effective data portability systems. They are proof that functionality and compliance don't need to be sacrificed to abstract complexity.



Data portability frameworks should finally break the twin barriers of understandability and usability, thus unleashing practical data portability for any type of user. Simple interfaces, rich metadata, unified presentation of data, low-code platforms, and dynamic adaptability ensure meaningful and effective user interaction with their data. This will meet the regulatory expectations for trust and empowerment of users and provide the means by which the digital ecosystem becomes inclusive.

Conclusion

In conclusion, user control over data cannot be achieved without eliminating barriers in usability and understandability. Current data portability frameworks indicate that technical compliance is prioritized over user experience. As a result, non-technical users are excluded from exercising their right to data portability. Proper data portability requires intuitive interfaces, simplified processes, and data presentation in a clear, contextualized way. Emerging solutions show that it is possible to bridge the gap between complex technical systems and user accessibility. A combination of unified data views, graphical representation, low-code or no-code platforms, and metadata embedding can transform data portability. A shift in focus to clarity and simplicity can practically empower all users, regardless of technical expertise.

2.1.2 Extensibility and Scalability

In the previous section, we highlighted the importance of building data portability frameworks with a focus on usability and understandability. These systems should also be technologically robust enough, to handle the growing demands due to the technological evolution. Models are required to deal with new data types, structures or functionalities as well as to effectively perform with increasing data volume and user base. In parallel, performance should be maintained even with the accommodation of the above. In this section we will explore the challenges of extensibility and scalability and outline solutions that address them.



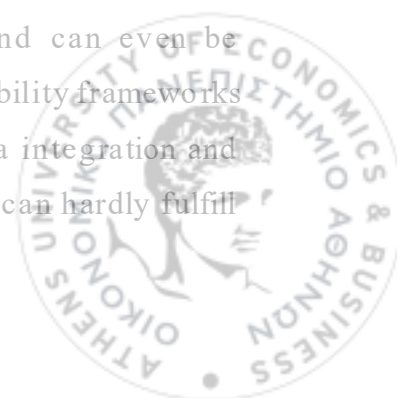
Challenges in Extensibility and Scalability

By definition, Big Data is characterized by volume, velocity, and variety, and thus is inherently challenging. Any framework designed for data portability is required to handle large volumes of data emanating from transactional systems, social media or IoT devices. The speed at which data is produced dictates that systems process information in real time, while the variety in data formats like structured, semi-structured, and unstructured requires a model that is highly flexible and adaptable. Rigid data models with fixed schemas are not suitable for such a high level of heterogeneity and dynamism. [23][24][25]

One of the main challenges is volume management. The amount of data generated and collected rises exponentially. Datasets can become too large and complicated to be handled by traditional methods of data processing. For instance, an e-commerce platform may handle millions of transactions per day. The data storage, processing and transfer capabilities of a system should be scalable, in order to avoid performance disruption. There are solutions that are able to support large datasets, like distributed systems, cloud storage and scalable databases. Their implementation, however, brings forth additional challenges. One must consider ways to ensure that data remains consistent, latency is minimized, and resources are allocated effectively.

Aside from volume, the speed at which this data is generated is another critical challenge. Real-time data generation requires systems that can ingest, process, and output data fast. Latency is the delay between a user interaction with the system and the moment it reaches the system. Correct and timely decisions are necessary in most sectors. For example, for credit card fraud detections or stock market data analysis. As a result, data integrity and accuracy need to be ensured, while latency is minimized. A trade-off arises between speed and reliability for seamless data portability.

Data are not only increasing in size at alarming rates, but they can also vary heavily. Data generation involves numerous formats, like JSON, XML, CSV, and can even be unstructured, like images, videos or free text. Due to this variety, portability frameworks should rely on flexible, dynamically changing schemas to allow data integration and transformation from various sources. In contrast, static, rigid schemas can hardly fulfill



such expectations. For example, to improve product recommendations, an e-commerce shop needs to combine purchase history (structured data), product reviews (unstructured) and social media posts (semi-structured). Schema evolution techniques are rendered necessary to facilitate real-time adjustments without tampering with existing operations.

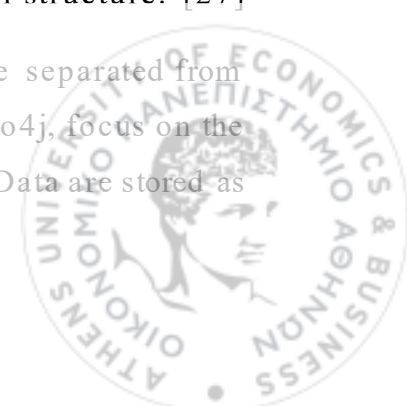
The GDPR Article 20 clause of "where technically feasible" brings forth scalability and extensibility as intrinsic features that need to be supported by systems for data portability. These systems need to grow with increasing volumes, diverse formats, and ever-evolving personal data on digital services. In the absence of these attributes, the exercise of the right to data portability will become impracticable, more so with emerging new technologies. [1]

Extensibility: Addressing Evolving Data Needs

An extensible data model supports schema evolution, addition of new data types, and changes in data structure without affecting its already existing functionalities. A truly extensible framework focuses on adapting seamlessly to new requirements, in a way that evolving data ecosystems remain operational and effective. This is critical for data portability, with the emergence of new platforms, applications, or devices that brings along new formats and structures of data that must integrate seamlessly into the already existing ones. [26]

Adaptability to emerging requirements is critical in any data portability system, given that the data landscape constantly evolves. Traditional models that rely on fixed schemas are severely limited in this regard. Most of them are not able to accommodate new data structure formats, as even minor adjustments may cause them to completely stop functioning. In contrast, flexible frameworks that allow dynamic schema management can meet such challenges, interpreting and adapting to data in real-time. For example, new IoT devices or services may generate new data types. An extensible framework is able to handle these elements without changing its core fundamental structure. [27]

To achieve integration of diverse data sets, data entities should be separated from relationships. Following this approach, Graph-based models, like Neo4j, focus on the relationships between the entities rather than the entities themselves. Data are stored as



nodes and edges. The relationships between the nodes are of primary interest in graph databases. Such models allow schema evolution by dynamically linking new data types and attributes. In contrast with rigid schemas, schema-free or schema-optional design does not require changes to the underlying structure of the database. As a result, diverse data sources do not hinder system relevance, as schema evolution is possible. Linking new data types does not require changes to be made to the underlying structure [12]

Another critical extensibility enhancer is modularity in design. Complex software is broken into smaller, self-contained components with specific functions. This approach makes both system development and management more efficient. With modularity, users can update, replace or expand only individual components of a system, without affecting its wider integrity. A new functionality, such as real-time analytics or advanced data visualization tools, can be integrated as optional extensions. [28]

Data portability keeps evolving by incorporating new technologies and data formats. This can lead to the formation of silos, which is when data is inaccessible by other systems. This is why it is important for new systems to be compatible with old ones, which is called backward compatibility. This property minimizes the risk of data loss and inaccessibility, promoting user trust. So, extensibility also covers the sustainable evolution of frameworks, enabling organizations to evolve their infrastructure without compromising on data accessibility or integrity. [29][30]

In addition, extensibility benefits from the presence of user-driven customization features. They allow users to make system adjustments in order to meet their specific needs. Such adjustments include the definition of workflows, data structures or integrations. This not only empowers them but also makes the system more flexible and manageable. End-User Development (EUD) frameworks are specifically designed to allow people without programming expertise to interact with the software. When the framework is able to adapt to diverse and changing user needs it is also dynamic. In the context of data portability, such systems assure an inclusive and user-friendly environment of personal data management, allowing users to directly engage in shaping how their data is stored, transferred, and integrated across platforms. When the users are



actively contributing to the evolution of the system, it results in a more responsive, flexible system architecture. [31]

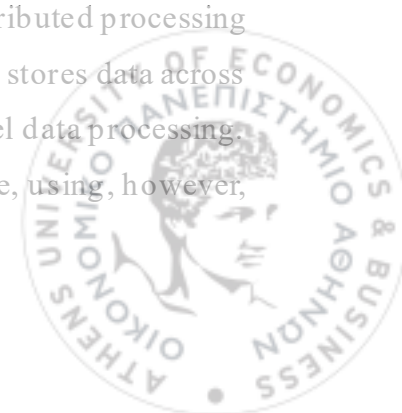
In summary, extensibility requires frameworks that are adaptable, future-proof with changes both in technology and society. This is achieved by dynamic schema management, relationship-based data modeling, modularity, backward compatibility, and user-driven customization. By combining these, data portability systems can remain robust, relevant, and effective regardless of changes.

Scalability: Managing Growing Data Demands

Data volume and user demand are constantly increasing; thus, scalability is critical for any data portability framework. Scalability ensures that the system will be able to scale through such increases efficiently without degradation in performance, reliability, or usability. While extensibility has to do with adapting to new types and structures of data, scalability is about being able to process more volume, larger concurrent user bases, and higher workloads of data with ease.

The unprecedented scale of data creation has been referred to as the “data deluge” and creates significant scalability challenges. Data portability frameworks should be capable of processing numerous data resulting from transactional systems, IoT devices, social media, and several other digital services. This requires designing systems based on distributed architectures: the slicing of larger data into portions and parallel processing in order to ensure consistent performance, given the increase in data volume. [32]

Distributed data processing is a paradigm of computing which allows large volumes of data to be processed on several nodes or machines in a distributed system. This makes it especially amenable to handling the scale and speed of modern data, which often exceeds the capabilities of a single machine. Large computation tasks are split into smaller, manageable pieces and operations can be performed concurrently as they are divided amongst multiple machines. Two examples of systems that support distributed processing are Hadoop and Apache Spark. Hadoop Distributed File System (HDFS) stores data across various nodes, and the MapReduce programming model handles parallel data processing. On the other hand, Spark builds on this well-known idea of MapReduce, using, however,



in-memory processing, therefore enabling swift computations of data by keeping the interim data in memory instead of spilling it onto a disc. Probably the most important reason behind such a system is scalability. The horizontal scaling of nodes in a distributed system allows the organizations to scale to meet the increasing volume of data without an expensive overhaul of the infrastructure. Such scalability is greatly needed in data portability contexts, as data from diverse platforms and sources needs to be efficiently processed to satisfy user or regulatory demands. These systems can present challenges however as they require advanced expertise, while presenting the risk of data inconsistency. [33]

Real-time data processing renders scalability even more complicated. In environments such as IoT, data is generated at unprecedented rates, rendering real-time processing even more complex. Building solutions that can handle high data heterogeneity and generation is a must for effective data portability. Technologies like Apache Kafka and Apache Flink provide scalable stream processing, thus enabling data portability frameworks to handle high-speed flows of data without sacrificing any accuracy or performance. Real-time scalability means that systems should maintain low latency and high throughput. This balance is critical because, for visualization, sharing, or further analysis, users require their data in real time. If this delicate balance is not maintained, it can lead to delays, hence reducing the overall usability and user satisfaction of the system. [34][35]

Despite the benefits of distributed architectures, latency and other data consistency-related challenges might occur. Latency refers to delays in user interactions when data needs to travel through many nodes or even different geographical locations. This challenge is partly addressed in many scalable systems by replicating data in a manner that frequent data can be available to users at closer distances for quicker retrieval. Another challenge is real-time consistency among distributed nodes. Techniques of eventual consistency must be implemented so that the whole system is scalable while ensuring that the update propagates in all nodes within reasonable time. For these techniques, however, it becomes very important not to allow any conflict/discrepancy in key data. [36]



Efficiency in the utilization of resources is the signature of scalable systems. There are multiple techniques that dynamically manage resources to ensure computational power serves where it is needed. Containerization, such as Docker, is the minimization of a system to only the necessary items for its execution. Orchestration platforms are then used, such as Kubernetes, for effective management and organization. Proper resource allocation ensures that systems can scale both vertically and horizontally. Auto-scaling is also important, allowing systems to adapt to real-time demand. Resources can be saved during periods of low activity or upscaled during periods with increased workload. [37]

As the digital landscape rapidly expands, systems should be able to handle larger datasets, more users, and increasingly complex workflows. Scalability ensures that data portability systems can grow with it, accommodating the increasing demands. Practicable solutions include distributed architectures, real-time processing, and resource optimization. Ensuring that scalability remains one of the strengths of data portability frameworks, while empowering users and supporting their rights in an increasingly data-driven world, will be assured by overcoming challenges with latency and consistency.

Conclusion

Scalability and extensibility are the foundation for making data portability frameworks future-proof, flexible, and efficient. As data grows exponentially in volume, with diversity in formats, and with the expanding user requirements, the static and rigid traditional models no longer suffice. Instead, modern data ecosystems require dynamic, modular, and distributed solutions. With the adoption of flexible schemas, modular architecture, and graph modeling, frameworks become extensible, allowing the addition of new data types and functionalities without disturbing existing system integrity and usability. Backward compatibility and end-user customization also improve the long-term viability of such systems, thereby rendering data portability more attainable. At the same time, advanced technologies such as distributed systems, real-time processing, and resource optimization allow systems to manage the challenges presented by Big Data, such as volume, velocity, and variety without compromising performance. Data portability solutions are then able to scale cost-effectively to accommodate heightened regulatory, business, and user needs.



2.1.3 Linkability and Interoperability

One of the pillars of the GDPR is the ability of users to transmit their personal data across service providers with no obstacles. The first functionality for effective data transfer is linkability, which is the successful integration of data across different systems and the ability to connect them meaningfully. However, systems have differences in technologies, formats or standards used. In spite these, interoperability is the ability of a system to understand, process and use the data. On the one hand, linkability ensures that data remain cohesive during transfers, without context loss, while interoperability focuses on data exchange with no compatibility issues. They are complementary requirements for building a meaningful and actionable data portability model.

The Challenges of Linkability and Interoperability

GDPR's Article 20 provision for data transfer "without hindrance from the controller" puts an emphasis on interoperability as an inherent necessity. Data portability can only succeed if models ensure that data formats are compatible across systems for smooth, lossless transitions of data. It is critical for transferred data to retain their context and remain coherent on the receiving systems. Requirement for manual interventions or custom integrations undermines the principle of "without hindrance." Although the GDPR encourages the development of interoperable formats, it does not oblige providers to adopt compatible systems. [1][38]

This challenge has been further highlighted by the fast growth in data generated from IoT devices, social media platforms, and enterprise systems. Data, these days, is generated in various formats such as structured, semi-structured, and unstructured and needs to be harmonized for cross-platform linkages. However, the existing systems usually lack flexibility for standardization and integration of diversified data efficiently. This is especially difficult to achieve since most platforms depend on proprietary or non-standard formats. While the GDPR requires structured, machine-readable formats, it does not require interoperable data outputs. This loophole allows fragmented data systems in which users may be able to technically port their data but, in practice, pose significant utilization challenges. For example, the contact information a user exports in some



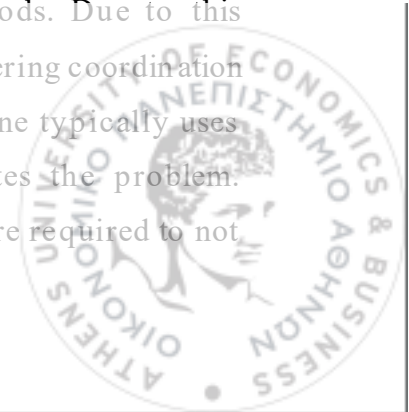
proprietary CSV format may not match the exact schema that the receiving service is expecting, thus limiting its utility. [2][26]

ISO 2382 defines portability as “The capability of a *program* to be *executed* on various types of *data processing systems* without converting the program to a different language and with little or no modification”. The application of the principle to data will, therefore, require interoperable systems to make their data output transferable across platforms with minimum reconfiguration. [39]

Frameworks are needed that can both define structured formats and can handle compatibility across diverse platforms. One of the main difficulties is maintaining relationships between the entities and therefore context during and after the transfer. The integration can become particularly complicated when dealing with data with varied granularity, like location. Semantic frameworks like RDF, graph-based databases like Neo4j or linked data solutions like Solid can support the linking of entities and attributes to ensure that after data transfer, it will remain usable and meaningful in the context provided. [12][18][40]

Currently, there are no widely adopted data standards, preventing seamless data exchanges between systems. Additionally, there can be contradicting industry and regional proprietary protocols and schemas. For example, healthcare data are based on FHIR, while financial data relies on ISO 20022. It is evident that framework heterogeneity requires advanced technical skills and resources to perform proper data mapping and transformations. [41][42]

In addition to structural difficulties, semantic misalignment is an even more pervasive problem. The same terms can mean different things in different systems, and even similar concepts may be expressed in different terms. These inconsistencies prevent proper linking of datasets. For example, the AEC industry constitutes of multiple specialized domains with individual terminology and data representation methods. Due to this fragmentation, a single asset might have multiple representations, rendering coordination and integration of data difficult. In the AEC context, each discipline typically uses distinct software tools and file formats, which further exacerbates the problem. Additionally, the size and complexity of datasets increase. Systems are required to not



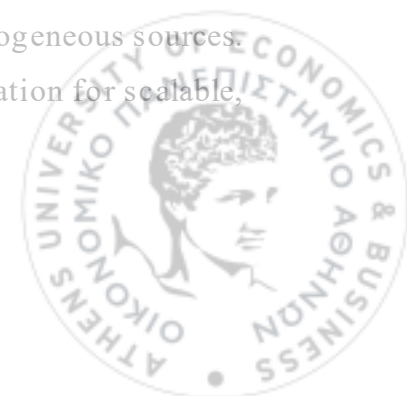
only handle the technical requirements of data integration but also the latency of complex linking processes. [43][44]

Designing Linkability and Building Interoperability

Linkability is possible only with systems and frameworks able to identify, connect, and contextualize data originating from disparate sources. Interoperability essentially means allowing systems to "speak the same language". To achieve this, data portability frameworks should be based on certain principles and tools. OECD highlights the importance of standardized data formats and APIs to achieve interoperability. However, to ensure consistent meaning and usability of the transferred data, APIs heavily depend on metadata structure alignment and universal standards. [4]

One of the ways to achieve linkability is to assign unique identifiers to different data entities. Even when data are derived from different systems, they can remain accurately mapped. The UUID method (Universally Unique Identifiers) is used to standardize identifier assignment, while ensuring global uniqueness and avoiding system conflicts. Domain specific identifiers are equally important, like unique patient IDs in the healthcare sector. With this ID, tracking medical history, prescriptions and treatments is successful across different healthcare providers. Linking data properly, without ambiguity or overlapping, is vital for maintaining meaning during transfer, which is a necessity for interoperable systems. [45]

Data is evolving over time, requiring real-time updates and adaptations to its schema. Dynamic mapping frameworks have the ability to link new data without manual intervention or structural changes. In such frameworks, data remains consistently connected, even when new data are introduced. In graph databases, such as Neo4j, data are represented as nodes and edges. No matter the format or source of the dataset, mappings can be created between them. Because of this treatment, identification of relationships or dependencies within large datasets becomes easier. This is particularly beneficial when dealing with evolving data which originate from heterogeneous sources. Ultimately, they are a powerful tool for linkability, providing a foundation for scalable, adaptable and consistent data interoperability. [12][46]



To preserve the relationships of data from the target system, data portability frameworks should enable schema mapping and transformation. Semantic frameworks and query reformulation techniques provide pragmatic solutions. A technique called Global-as-View treats the schema as a unified global schema, allowing data from various sources to be mapped in a single representation. Through this, even diverse datasets can become consistent and aligned. Local-as-View is another technique that treats data from one system as a subset of a global schema. The mapping process is thus simplified by focusing only on specific data portions for integration. These techniques help in maintaining the integrity and usability of the data during and after its transfer by resolving the format and schema variations. Global-as-View can be used in healthcare to unify various medical records from different systems into a single, coherent patient view. Similarly, Local-as-View can be applied when integrating sensor data from IoT devices in smart cities, where each sensor's data may be structured differently but needs to be represented under a common schema for analytics. Another example of semantic framework is RDF, where relationships are defined between entities through triples. Even as the underlying data formats or systems change, the relationships between data entities remain intelligible and usable in such systems. [10][47][48][49]

Linkability and interoperability also rely heavily on metadata because it adds, to the data itself, further context concerning origin, time, and purpose. Without this information, data might lose its meaning, rendering accurate linkages difficult. Standardized, well-structured metadata ensures a higher degree of accuracy, relevance, and usability for the linkages. Metadata can track the source of an entry, like real-time feed, historical record or user submitted. It can also include timestamps, data format specifications and even information about any undergone transformations. A fine-grained metadata schema could even be helpful for runtime query reformulation in further enhancing the efficiency of linkages. For example, in healthcare, metadata might include age or gender, which can be used for more accurate and timely data linkage. [41][50]

Systems additionally rely on various programming languages and architectures, and thus data exchange is not impossible without a common framework. Standardized data formats like JSON-LD or XML are created with a single purpose: to offer a structure that can be



read universally. JSON-LD combines the well-known JSON format with Semantic Web principles. So, data are described with entities and relationships, as triples, same as in graph frameworks. Because of its structure, it allows different systems to interpret the data in the same way. XML has also been widely used for structured data representation by diverse systems. They are linked through nested tags and attributes. It does not rely on specific platforms as it contains schemas and constraints to maintain data structure. Universal standards represent another crucial element for interoperability. Some examples are Dublin Core for metadata and FHIR in healthcare. In healthcare, access and understandability of patient records is critical to the providers, as mistakes may lead to life-threatening decisions. Therefore, proper data interpretation is critical, regardless of the software used. With standardized data formats and aligned metadata, data from different sources can be integrated without errors. [5][18][41][51][52]

Conclusion

Neither interoperability nor linkability are static objectives, as they need to be upgraded as technologies, data types and user requirements evolve. Cross-sector interoperability, for instance, has even more issues to deal with since the demands for health care data are very different from those of finance or IoT. Implementation obstacles, such as standardization costs, resistance from organizations with proprietary formats, and complexities in aligning metadata structures, further exasperate the challenge. Empowerment of the user is, therefore, at the center of these efforts. Interoperability ensures that users can move their data freely, and linkability ensures that their data remains meaningful and actionable. A combination of the two is the only way to achieve user-centric data management without hindrance. By leveraging semantic techniques, universal standards, and innovative integration methods, data portability can become not just a legal right but a practical reality.



2.2 Non-Technical Requirements for Data Portability

Besides the technical challenges outlined in the previous sections, data portability models need to consider non-technical factors, such as regulatory compliance, data governance, security, and privacy. Regulatory compliance is necessary under data protection laws such as the GDPR, which require transparent and user-controlled data portability. Data security and privacy are equally important through every stage of the data life cycle, often through encryption and access control mechanisms. Among others, anonymization is a privacy requirement to ensure that the portability of data does not affect user confidentiality. By embedding privacy-enhancing technologies and secure audit trails, a data model earns user trust. Altogether, these non-technical requirements constitute a compliant, secure, user-centered model of data portability in conformance with regulatory standards and protection of individual rights.

2.2.1 Regulatory Compliance and Data Governance

Every aspect of our lives increasingly relies on data; therefore, regulatory compliance and good data governance are not only purely legal requirements but also fundamental to user trust and operational efficiency. The implementation of GDPR highlights the need for governance systems that adequately protect user rights, while taking into consideration business and regulatory requirements. However, setting up an effective governance presents several challenges. Implementation of advanced techniques for compliance and adaptability are required in the dynamic regulatory environment. Data governance provides a framework for effective data management, with the goal of data quality, integrity, security, and usability. Regulatory compliance is the alignment between organizational practices and regulatory requirements, in order to avoid breaches, penalties or reputational damage. [53][54]

Importance of regulatory compliance

Regulatory compliance depends on robust data governance to maintain data integrity, structure, and utility during transfer. The GDPR requires that a given organization adhere to accountability, transparency, data minimization, and similar principles while handling



personal data. More specifically, Article 30 covers the requirement of maintaining records of all data processing activities by an organization. The above is called data provenance, which means logging relevant information about the origin, history, and transformations applied to the data. In this way, every step of a data life cycle would be transparent and traceable. When each access and transfer is logged by the model, an audit trail can be provided. [55][56]

Consent Management is the process of dynamically capturing user preferences and permissions, and respecting them, in compliance with Article 6 about lawfulness of processing, and Article 7 on conditions for consent. The GDPR further intersects with other rights, such as Article 17, Right to Erasure, which demands that the mechanisms of data governance avoid unauthorized retention of data after it is ported. Models must balance portability with erasure, ensuring that data that has been transferred does not violate erasure rights of the data subject. [55]

Challenges in data governance

Governance frameworks provide the necessary data management structure, but the practical implementation presents several challenges, such as proper metadata logging. Metadata is used to keep track information about the data, such as its origin, and how it is processed and moved. Extensive metadata logging can have a negative impact on storage and processing systems while insufficient logging makes audits and regulatory compliance impracticable. Organizations need to find the right balance between metadata tracing and system performance, by developing proper metadata management strategies. [57]

In recent years, new frameworks such as the Digital Governance Act (DGA) and the Digital Markets Act (DMA) have been put into effect in EU. Regulation tends to change due to technological evolution, geopolitical circumstances, or societal priority shifts. In this case, organizations find that static models of governance quickly become outdated. Flexibility in governance frameworks is critical to accommodate the new obligations without major reengineering effort and thus calls for proactive regulatory landscape monitoring and continuous investment in compliance readiness. [58][59]

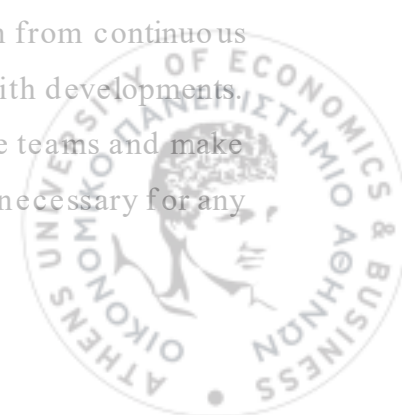


Both regional and industry specific requirement make compliance even more challenging. For example, in the EU, the GDPR prioritizes privacy and data minimization, while other regions might require extensive data storage for security or business purposes. Also, standards such as HL7 in health care and ISO 20022 in finance have been developed to meet the needs of a particular sector but are usually incompatible with one another. As a result, data governance mechanisms should be highly adaptable both to regional and global regulations, as well as sectoral standards. Forcing standards alignment brings inefficiencies and increases the risks of compliance for organizations involved across multiple regions or sectors. [41][42][60]

Techniques for regulatory compliance and governance

To achieve regulatory compliance, organizations are trying to utilize advanced techniques to address the above challenges. The most important technique is called data provenance, which is a form of metadata which documents the origin, and transformations applied to the data. Metadata is directly incorporated within the data and carries semantic meaning. This allows for easy verification of any data transformation or manipulation that is performed. Organizations can efficiently save on investigation time as problems or inconsistencies are recognized quickly and subsequently resolved. Models, such as graph databases, can be utilized for data provenance. In a graph, nodes can represent the data, and edges can represent the transformations applied. With the dynamic data interconnections of graph databases, organizations can get a complete view of data flow and transformation. With this view, the efforts required by a regulator or auditor while verifying compliance or assessing data quality can be minimized. The journey between data input and output is clear, and every step can be reviewed against the necessary standards. [12][61]

Monitoring of compliance has also transformed in recent years with the emergence of artificial intelligence. Machine learning algorithms can monitor data flows to identify anomalies, flagging associated compliance risks. Such tools also learn from continuous changes in regulations and thus can support organizations to keep up with developments. AI provides the necessary insights to lighten the work for governance teams and make their decisions more effective. An adaptable governance framework is necessary for any



organization to meet regulatory changes without changing the existing systems. It allows companies to adopt modular platforms, where on-demand schema adjustments can be made while keeping the structure relevant to existing requirements. These frameworks allow a company to adapt without any operational disruption. [62]

Automating consent management systems ensures ease of compliance with the GDPR requirements. Consent management platforms empower users through the facility for easy permission-granting, amendment, and withdrawal of permissions. This of course requires user-friendly platforms, like the ones discussed in the previous section. When there are obstacles in the process, like overcomplicated user interfaces with puzzling options, user autonomy cannot be achieved. Providing consent, and in turn modifying or withdrawing this consent should be an effortless process. This closely aligns with the GDPR's vision for full user control over their personal data and also builds trust between customers and organizations. Systems should automatically inform users on how their data is being used, so they are able to make decisions on the most up-to-date information. Additionally, organizations also save time by limiting the manual processes required to ensure regulatory compliance. [63][64]

Governance dashboards, like OneTrust or SAP GRC can help organizations manage compliance. They provide real-time monitoring and predictive analytics. Organizations are able to identify any violations in advance and resolve them before they escalate. The dashboards help keep the governance strategy dynamic and responsive toward emerging risks by providing actionable insights. [65][66]

Conclusion

A strong data governance mechanisms focuses on transparency, security, and accountability, allowing individuals to exercise their rights with confidence. Organizations need to make sure that data portability is both effective and ethical. Through its life cycle data integrity and transparency should remain unchanged, through proper data provenance and consent management. There are many challenges involved in the implementation of such frameworks, like regulatory variability, metadata overhead and automation of accountability mechanisms. However, with techniques such as real-



time monitoring, AI and dynamic adaptation can make governance more efficient and resilient.

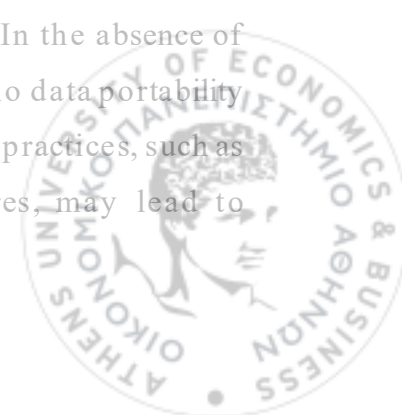
2.2.2 Data Security and Privacy

As already discussed, the main goal of data portability is to empower users, and therefore the privacy and security of their personal data should not be risked. GDPR specifically highlights that data portability should not "adversely affect the rights and freedoms of others," emphasizing the application of sound security and privacy protection in data portability models. Since data portability requires movement from one entity to another, it puts personal data at jeopardy pertaining to unauthorized access, data breaches, or even violation of privacy. In order to meet these challenges, data models need to have enhanced mechanisms of data protection so that the data that is being transferred is secure and the regulatory standards are still in compliance. [55]

Challenges in Data Security and Privacy

Data portability involves the transfer of sensitive data across multiple systems or platforms. During the process, vulnerabilities may arise, especially without the proper security measures in place. The security risk is high, as malevolent individuals may take advantage of such vulnerabilities to steal data. The more complicated the system, the highest the possibility of breaches, as more users are involved. The consequences of a successful attack can be catastrophic, including identity theft, financial fraud or exposure of personal information. A major credit bureau, Equifax, was hacked in 2017. Personal data of millions of people were exposed, such as social security numbers and credit card details. [67]

Portable data is most vulnerable if the encryption protocols are weak, or their implementation is inconsistent. Weak cryptographic algorithms, or lack of end-to-end encryption, will expose data in transit to interception and tampering. In the absence of strong encryption and secure transfer protocols such as TLS or SFTP, no data portability initiative can ensure confidentiality and integrity. Poor anonymization practices, such as ineffective pseudonymization or weak differential privacy measures, may lead to



reidentification risks, especially when datasets are cross-referenced. Not adhering to data minimization principles, in order to transfer only the required data, exposes information, risking misuse or breaches. [2][4][68]

Techniques for Security and Privacy

It is essential that data remains secure throughout its life cycle, meaning in rest, use, and during transfer. As outlined above, unauthorized data access can be catastrophic. Therefore, only approved users or systems should be able to interact with sensitive data. Through data encryption, data is converted into a format that is readable only with the proper decryption key. Thus, even if data is breached, its contents are not accessible. Some common cryptosystems are Advanced Encryption Standard (AES) and RSA, typically in combination with public-key infrastructure (PKI) for additional security. In addition to encryption, strict access controls should be implemented. With Role-Based Access Control (RBAC), authorization is provided according to a user's role, and Multi-Factor Authentication (MFA) adds additional methods, such as biometrics, to verify a user's identity. [69][70][71][72]

Transfer of data should be done, where possible, using secure transfer protocols such as TLS or SFTP. Such protocols create encrypted channels; therefore, even if intercepted, access to the data would not be possible. Secure APIs allow for token-based authentication for exchanges of data between platforms, allowing data sharing only between verified recipients. Additionally, transfers should only be performed when required. Data minimization means transferring only data strictly necessary for the purpose. The outcome is less exposure to information and reduced risk of misuse or breach. Systems should also implement mechanisms for secure deletion in order to ensure temporary data gets removed when no longer needed and not a liability. [73][74][75]

Additional techniques include pseudonymization and differential privacy. Pseudonymization replaces direct identifiers with pseudonyms, and differential privacy does so by adding noise to datasets in a manner that linking data back to the subjects becomes impossible. These techniques ensure privacy without any cost on data utility,



especially in analytics and research contexts. The ultimate goal should be however, to design platforms with privacy at their foundation to ensure that security features are implemented at all critical points in the data lifecycle. This requires interfaces to manage user consent, audit logs to track every action taken against data, and real-time breach detection through intelligent systems. [76][77][78]

Conclusion

Strong security and privacy measures in data portability are not just regulatory obligations but necessities for user trust. To align with the GDPR mandates and OECD recommendations, models should focus on encryption, access controls and privacy-preserving techniques. In order to have a good framework, these would have to balance user empowerment against data protection-data breach risk, inadequate encryption, incomplete anonymization would require strong techniques such as encryption of data, transfer protocols, and platforms of privacy-by-design. This approach will let the benefits of data portability be availed without compromising on security or privacy, thus building trust and resilience within the digital ecosystem.



3. Related Work

In a highly digital age, data portability is at the forefront, with emerging regulations like the GDPR, and increasing user demand for personal data management. A few frameworks and models have been developed to meet the technical and organizational challenges preventing easy data transfers between platforms. Examples of these efforts include frameworks for personal user control, data transfer or integration interoperability frameworks, and sector-specific initiatives. This section discusses these frameworks, with a focus on their methods, strengths and weaknesses regarding data portability. Analysis of existing work provides a better understanding of where progress has been made and what has fallen short, thus setting the stage for more complete solutions.

3.1 Personal Data Control

Data Spaces

Data spaces, as frameworks for data-sharing, are not a new concept, but their application to individuals and personal data management remains underdeveloped. Data spaces bring a new level to data management with decentralized environments for interoperable, secure, and user-centric data sharing. In traditional centralized systems, data access, management, and use are controlled by a single entity, usually the organization. On the other hand, data spaces guarantee data sovereignty and provide users with complete control over their data. In this way, users are able to define who has access to their data, for what reason and under which conditions. This is consistent with the rights enforced by regulations such as the GDPR. [79][80]

The principles on which data spaces are built on, make them stand out among other data management frameworks for data portability. Interoperability is the cornerstone of their design, which is achieved with the adoption of shared standards on data formats, schemas, and protocols. This ensures that heterogeneous systems are compatible, thus data integration is allowed without extensive preprocessing. They also fundamentally rely on federated architecture, where data remains decentralized and in full control of its owner. Secure access mechanisms enable collaboration, rather than physical data transfers,

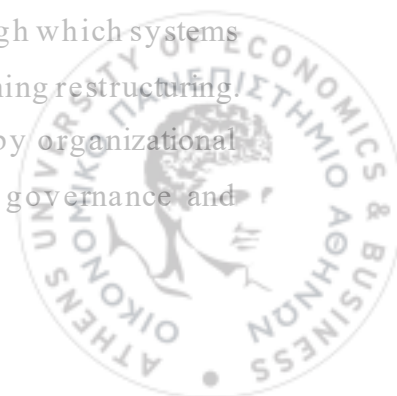


greatly reducing centralization risks such as unauthorized access or data breaches. [81][82]

As outlined above, data spaces are built on federated architecture, which allows data to remain with the original owner and accessible only through shared interfaces. Data spaces have no use for central depositories, which are associated with privacy concerns and infrastructure overhead. Moreover, semantic interoperability is enabled through utilization of ontologies and standardized vocabularies. Without it, data might lose its meaning through transfer, rendering proper integration impracticable. Data sharing is performed through secure data exchange protocols, like APIs and federated queries. Access to the data is performed through user-defined access controls and policies. Data spaces allow both virtual integration, which is access in the original location, and physical, where subsets of data are replicated before usage. [81][82]

Data spaces rely on multiple models for their operation. The Resource Description Framework (RDF), provides a standardized structure for representing data relationships as triples, forming the backbone of many semantic data environments. Based on RDF, knowledge graphs represent the entities and their relationships in a highly interoperable format, allowing semantic querying. Additionally, federated query systems allow the execution of queries across various data sources, with no requirement of physical integration into a schema. Through the above, data spaces remain decentralized while providing a unified view of distributed data. To align diverse datasets, ontologies and metadata standards provide a common language and structure for seamless data portability. [81][82]

Data spaces revolutionize data portability by addressing a number of key challenges. The first is allowing interoperability across platforms, breaking silos as data can freely flow between systems. The second is giving users control over their data, enabling them to manage, access, and share their information with complete independence from service providers. Data spaces also allow for dynamic schema handling, through which systems can adapt to new data requirements without expensive and time-consuming restructuring. They could also be a solution to the exploitation of data facilitated by organizational control. They build confidence among participants through clear governance and



transparent trust frameworks, thereby fostering collaboration with data protection. [81][82]

In addition to all the advantages, data spaces also entail challenges. Setting universal standards across diverse sectors is almost impossible, since it requires extensive coordination efforts among stakeholders. Building and maintaining data spaces is also expensive, especially for small organizations. Additionally, the regulations vary per region, requiring robust compliance mechanisms to ensure security and privacy. Widespread adoption is inherently difficult due to possible resistance from organizations. The latter are hesitant concerning data sharing, due to competition or even cultural barriers. [79][81][82]

Solid

Solid (Social Linked Data) is a decentralized framework for social Web applications based on RDF and other semantic web standards. First conceived by Tim Berners-Lee, the Solid project ultimately aimed to put the user in the center by decoupling data storage from application functionality. Its focus is creating personal data pods where users can independently store their data, without a service provider. These pods are web-accessible, meaning that users can choose where and how their data is stored, and even switch pod providers without losing access to their data. [40][83]

Decentralization is achieved by decoupling data storage from applications. Data are owned and controlled by the users. Applications may then request access to these spaces, through explicit user consent. The platform mostly relies on existing standards set by the World Wide Web Consortium (W3C). These standards are used both for authentication and communication between applications and pods or among pods. The pods are secured by Linked Data Platform (LDP) protocols. What makes Solid stand out is its interoperability, as applications can seamlessly access data stored in any compliant pod, regardless of location. The user can interact with the data with RESTful HTTP operations like GET, POST, PUT, and DELETE. SPARQL can also be used for more sophisticated queries. [40][83][84][85]



Solid provides the user with full control over their data and the ability to grant or revoke permissions to any application. Due to its flexibility, users are also able to switch between service providers easily. This follows the GDPR's guideline on user-centric data management. Widespread adoption of Solid is limited, due to technical obstacles. Although it empowers users, managing data pods and permissions might be challenging for non-technical users. When dealing with proprietary platform, in turn, full compatibility is difficult to guarantee.

Comparison with the DVM

The DVM, which will be presented in detail in section 4, shares a lot of similarities with Personal Data control frameworks. They all share the goal of enhancing data integration and accessibility. They offer dynamic schemas, enabling linking of heterogeneous data from diverse sources, and flexible data modelling. They are however different in the way they approach data portability and management. PDS and Solid are focused on how users manage data, instead of organizations. Data storage, access, and sharing permissions are entirely in the control of the individual. Solid is designed for web applications and cloud services. The users keep their own data in decentralized "pods" and third parties can access them upon request. PDS are used for sector-specific secure data exchanges such as in healthcare, finance, or government-regulated industries where compliance with sectoral standards is crucial. Both Solid and PDS require services to opt-in to their ecosystems, so companies and apps must actively support their frameworks for interoperability and data portability to function. In contrast, DVMs are tailored for enterprises with an emphasis on efficient data modeling, querying, and transformation over user-managed data governance. They are an independent data abstraction layer that facilitates organizations to gather, analyze, and operate upon large-scale sets of data in internal systems without necessarily requiring outside ecosystem interaction. While PDS and Solid are for data sovereignty for the individual and regulatory compliance, DVMs are more about optimizing structured and semi-structured data management at the organizational level.



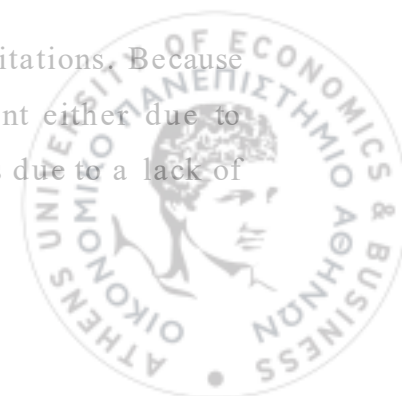
3.2 Data integration and Transfer

The Data Transfer Project is a collaborative initiative by 4 major tech companies: Google, Facebook, Microsoft and Twitter, to facilitate secure data transfers between services. In general, users can perform transfers on their own, without intermediating downloads or uploads. Interoperability is possible by utilizing open standards and protocols that bridge technical difference between systems. [86][87][88]

Data transfer in DTP is achieved by defining standardized formats for different types of data. Providers are not expected to build direct integrations between them because DTP converts data into a universal format to ensure compatibility. The heart of functionality for DTP includes two types of adapters that act as connectors between the participating platforms and the DTP ecosystem and ensure security. The first ones are data adapters and perform two types of transformations. The exporter extracts data from the source system and standardizes it. The importer takes the standardized data and converts it into a format compatible with the destination system. The second adapter type is for user authentication and authorization of data transfers. The method used is OAuth which ensures secure access and prevents data exposure. The actual execution of these transfers is handled by the Task Management System. [86]

DTP is used to directly transfer data between platforms with ease, without technical knowledge required, which is a requirement for data portability. It is highly extensible due to its modular architecture, allowing new platforms to be added into the ecosystem by just implementing the necessary adapters. By utilizing open standards and shared data models, data transfer is possible even among disparate platforms. In accordance with privacy regulations, DTP used Authentication via OAuth 2.0, along with encryption protocols to ensure secure data transfers. The initial focus of DTP is consumer-facing platforms. Because of its adaptability, it presents potential for use across sectors, like banking or healthcare. [86][89]

Despite all of the benefits, DTP also presents challenges and limitations. Because participation by platforms is voluntary, platforms will be reluctant either due to competitive reasons or resource constraints, limiting its effectiveness due to a lack of



universal adoption. DTP defines shared data models, however doing so for each and every kind of data is quite difficult. Potential variations in the way data will be represented across platforms might complicate transfers of that data seamlessly. Moreover, the resources required for the development and maintenance of adapters might prove complicated for smaller platforms, excluding them from the ecosystem. [86]

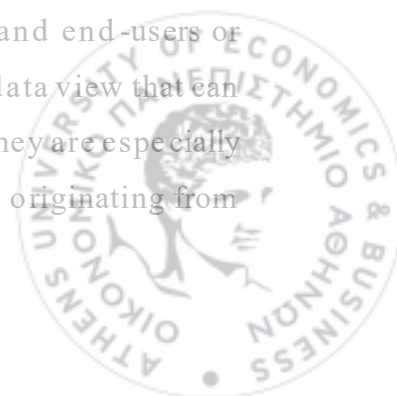
The Data Transfer Project is a milestone towards seamless data portability. Due to its modular architecture, use of standardized formats and focus on user control, it can facilitate data transfers. However, to show its full potential, it requires broad adoption, refinement of standards and compliance measures. By addressing these challenges would make DTP a paradigm of interoperable and user-friendly data portability systems in the digital economy.

Comparison with the DVM

The Data Transfer Project (DTP) and Data Virtual Machines (DVMs), which will be presented in detail in Section 4, both aim to improve data portability by enabling interoperability between services. They both abstract differences in data formats, facilitating compatibility across diverse systems. However, DTP enables direct data transfers between services, allowing users to move their data seamlessly without manual downloads, provided that participating platforms support the framework. DVMs, in contrast, provide a conceptual data layer that enables users to query, analyze, and transform data before exporting it. Unlike DTP, which depends on service cooperation, DVMs function independently, allowing organizations and users to integrate and explore heterogeneous data without requiring predefined data transfer agreements or platform participation.

3.3 Data Integration and Virtualization

Mediator-based systems act as intermediaries between data sources and end-users or applications. Even without a centralized database, they offer a unified data view that can be used for querying and retrieving data from heterogeneous sources. They are especially useful for data portability applications. They are able to integrate data originating from



disparate systems, including relational databases, NoSQL stores, and flat files with different formats and schemas. The source systems always remain in control of the data, following the GDPR requirement for decentralized data control. Additionally, mediators adjust to changes in data sources or schemas, which is crucial for extensibility and scalability. [90]

Federated systems integrate data from multiple sources into a virtual database. Unlike mediator systems, they may be immediately equipped with advanced optimization techniques for distributed query processing and provide near-real-time data integration. Users can query data from multiple systems as if they were a single entity, which enables the facilitation of seamless portability. Federated systems integrate data without modifying the source systems. They are capable of accommodating new data sources in a dynamic manner, increasing extensibility [91]

Denodo is a data virtualization platform mainly used for integrating and handling heterogeneous data. It creates a unified data view without requiring the actual physical consolidation of the data. It uses mediator architecture, providing a layer of abstraction between the actual data sources and the application using the data. It can handle both structured and semi-structured data from relational databases to XML files. It follows an approach similar to relational models, where each data source is represented as a combination of relationships. A global schema is created for users to access the data. In the Aggregation Engine, queries are broken into subqueries and then executed on the original data sources in an optimized way. Even when some sources are not available, queries can still be executed and provide a partial result. It additionally supports caching for improved performance. It is particularly useful for fast and cost-effective data integration for environments where data are scattered across various systems. It is widely used in corporate environments and internet-based application which require real-time access. [92][93]

Unlike traditional federated systems, the BigDAWG polystore system can handle data not only coming from multiple storages but also with different data models. A set of data engines share a query language and data model, forming "islands of information". Users interact with them through shims, which translate queries into a language compatible for



the storage engine. Cross-island queries are also possible through CAST operations, so that data are transformed between models. Through optimization mechanisms, the system learns from past performance and adjusts data placement and query execution accordingly. This system is ideal for complex environments with different data models, such as healthcare or enterprise applications. [94]

Comparison with the DVM

Virtualization platforms and polystores are similar to DVMs, presented in detail in section 4. They all unify heterogeneous data sources into a single access layer, where data can then be utilized without actually being transferred. These systems rely on abstraction layers to present a unified view of data stored in different platforms, just as DVMs create a conceptual schema to unite heterogeneous data. Their role in data portability is not the same, however. Denodo prioritizes enterprise data integration through virtual views and SQL interfaces to provide access to multiple databases without reconfiguration of the underlying model. BigDAWG is a polystore system which handles queries across different databases with interoperability at the execution level. While they foster data interoperability and data access, they do not inherently allow data restructuring, reformatting, or transformation. On the other hand, as will be demonstrated in section 4, DVMs focus on data virtualization with an emphasis on schema flexibility. Along with querying across data sources, users can also restructure and transform their data dynamically for a more flexible integration and portability solution.

3.4 Sector-specific frameworks

Open Banking Framework

Open Banking is a regulatory and technical framework that enables financial institutions to share data with third-party providers in a secure way through standardized APIs. The original regulatory motivation has come from the EU's Second Payment Services Directive (PSD2), a similar framework in the UK, with the overall goal of increasing competition, improving innovation, and putting the user in greater control of their financial data. [95][96]



Open Banking requires standardized APIs, ensuring interoperability amongst banks and TPPs. Through these, TPPs can either receive financial data or initiate payments upon explicit consent from the user. Consent of the user to share data must be explicitly obtained. Consent management systems are generally integrated into Open Banking platforms for compliance with GDPR. SCA is essential to check the identity of users in data access or whenever the user initiates any payment. OAuth 2.0 and OpenID Connect are commonly used for the purpose of securely sharing data and authentication. Users can additionally view and manage financial information from multiple banks through a single interface. TPPs will directly initiate payments from user accounts, thereby bypassing traditional payment methods such as credit cards. [95][96]

However, interpretation and implementation of Open Banking can vary regionally. For instance, PSD2 has variations in interpretation and implementation in each EU member state. Additionally, the increased number of TPPs dealing with sensitive information introduces vulnerabilities. Most small financial institutions face challenges due to the cost and intricacy involved with the implementation of Open Banking APIs. Both the benefits and risks of open-banking might not be as well known, which could result in lower-than-expected levels of usage. Open Banking is an excellent example of large-scale application of structured, machine-readable data sharing. It leverages standardized APIs and mainly focuses on user consent, showing close alignment with the GDPR's goals for data portability. However, the sector-specific nature of the latter limits the wider applicability to broader data portability contexts. [95][96]

FHIR

FHIR (Fast Healthcare Interoperability Resources) is a standards framework from HL7 International for exchanging electronic health information. Interoperability means the ability of healthcare systems to verify that patient information is accessible, shareable, and usable across different platforms or providers. Healthcare information, in FHIR, comes in "resources" that are modular, reusable units representing things like Patient, Observation, Medication, and Appointment. Each resource is formatted with one of several standard formats, including JSON, XML, or RDF. With FHIR APIs, healthcare providers and systems are able to share data in real time. Resources are designed to be

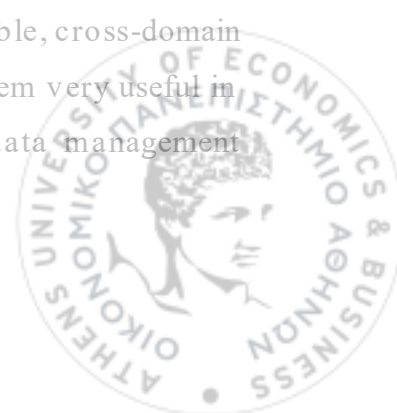


extensible, accommodating evolving healthcare needs. FHIR is currently used for exchanging patient records between hospitals, mobile health applications view and to visualize patient data. Research studies aggregate anonymized patient data. [41][97][98]

Healthcare data is very sensitive, and FHIR implementations have to strictly follow severe privacy laws like HIPAA in the US or GDPR in the EU. Data transmission and storage need to be totally secure. Different healthcare providers interpret FHIR resources differently, and thus, different implementations are done. Lots of healthcare systems still run on old standards or proprietary formats, which makes their interaction with FHIR quite difficult. While FHIR simplifies healthcare data exchange, the extensive resource definitions and optional elements can make implementation difficult for smaller providers. FHIR is one of the widely used sector-specific methods for data portability in healthcare. Its structure with resources, standardization, and interoperability mechanisms are fully aligned with the GDPR portability requirements. However, this is strongly dependent on healthcare-specific resources. [41][97][98]

Comparison with the DVM

Industry-specific frameworks such as Open Banking and FHIR enable data portability across regulated markets with standardized, structured data transfers. Like DVMs, presented in section 4, these designs prioritize interoperability so that different systems can exchange information through common data models and APIs. But their data portability approach is pre-defined and reliant on specific sectors, with little applicability outside their domain. Open Banking is used to transfer financial data between banks and third parties within pre-defined regulatory environments and data types. FHIR has a standard data exchange format for healthcare data among insurers, hospitals, and applications with semantic interoperability between medical record systems. DVMs offer a general solution, applicable for connecting heterogeneous data, allowing query formulation and dynamic evolution of schemas. While these frameworks are superb at enabling secure and compliant data sharing, they do not have the flexible, cross-domain data portability and integration capabilities of DVMs, which makes them very useful in their own domains but less effective for broader, multi-domain data management purposes.



4. The Data Virtual Machine (DVM)

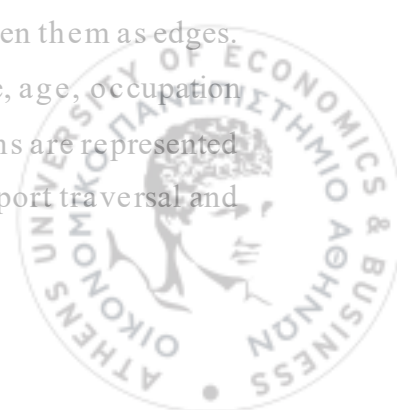
Modern analytics environments are characterized by extreme heterogeneity in data systems, formats and analytical needs, and are thus inherently challenging. Data are stored across various systems, making it difficult to extract any meaningful insight. Traditional data management and exploration approaches are no longer sufficient as they rely on fixed schemas and rigid hierarchies. Emerging data modeling techniques should be able to adapt to the dynamic multi-faceted nature of data environments. The Data Virtual Machine (DVM) presents a novel approach to data management and analytics. It bridges the gap between complex data systems and user accessibility by providing a unified, visual interface. Data interactions are simplified and can therefore be performed by users of all technical levels.

The main goal of DVMs is data virtualization. Access and manipulation of data from various sources can be performed as if it were a single entity. Data management can be particularly complex, with extensive data integration processes and advanced technical expertise as a requirement. DVMs, however, allow users to explore data relationships, select attributes and perform analysis quickly and efficiently, improving data accessibility and usability. This section is a presentation of the work conducted by D. Chatziantoniou and V. Kantere. In their publications [99]–[102], they introduce and develop the concept of Data Virtual Machines (DVM) and its applications in big data environments.

4.1 Theoretical foundation

The DVM, is a novel conceptual model established on multiple disciplines to achieve data management and analysis.

Graph models utilize graphs to represent relationships among entities. The entities (e.g. customer, product) are represented as nodes and the relationships between them as edges. The both can have additional properties, called attributes. For example, age, occupation can be attributes of the entity customer. In this way, complex connections are represented intuitively through a visualization. DVM relies on graph theory to support traversal and



querying of data, allowing dynamic data exploration. This structure is also easier for users to understand. As a result, users can derive insights from interconnected datasets. [103]

The Entity-Relationship model (ER) lays the foundation for database design. It is a conceptual model which focuses on the representation of entities and relationships. It creates a unified view from different data models. ER diagrams are used for the representation of entities, attributes and relationships. DVMs are an adaptation of this model, focusing on a more flexible, bottom-up approach, through which users can define and manipulate relationships as needed. This adaptability is very important in highly heterogeneous data environments, lifting the constraints of rigid schemas. [14]

Data Virtualization is related to mediators and virtual databases. It is a methodology that allows data access and manipulation without requiring technical details about the data, such as how it is formatted at source, or where it is physically located. It provides a single view of any entity of the data. Data virtualization may also be considered as an alternative to ETL and data warehousing. The goal is to produce quick insights from multiple sources without requiring a major data project. DVM provides a virtual layer where the data engineer can easily map data and processes related to an entity. In this respect, it can be considered as a data virtualization platform. [104]

Declarative Querying allows users to specify what information they need, without requiring the steps to retrieve it, like in procedural querying. The user focuses on the result, while the system is responsible to find the most efficient way to achieve it. This abstraction renders querying simpler, and thus accessible to users with varying technical expertise. DVMs follow this principle, focusing on the outcome rather than the process behind it. [105]

The Algebraic Framework, when used for data management, provides a formal mathematical foundation for the structure, query and transformation of data. It defines operations like selection, projection, join, aggregation, mapping and filtering, which can be then combined in queries to make transformations. DVMs relies on the these for query evaluation and optimization. Users are able to summarize and manipulate data to derive meaningful insights. [105]



When built using User-Centric Design principles, interfaces are intuitive and accessible. This is critical for widespread adoption of an interface, enabling users to utilize their data effectively. DVM is prioritizing the user experience, making access and data manipulation accessible to users of various technical backgrounds. [106]

In conclusion, the DVM's theoretical foundation is a composite of graph theory, entity-relationship modeling, data virtualization, declarative querying, algebraic structure, and user-centric design. Together, these elements create a robust framework for data accessibility, collaboration, and user empowerment to derive insights from complex data environments.

4.2 Structure

The DVM is defined, at its essence, as a graph, similar to the Entity-Relationship Model (ER). Traditional models, like the ER model, have a top-down schema design. In ER, the data are mapped onto a conceptual schema defined in the beginning. Attributes either contain one value, like age, or multiple values, like comments, and they can also be derived, containing values from computations. In contrast, the DVM follows a bottom-up approach, deriving the schema from the processing of the data, as shown in Figure 1. The inversed design allows the DVM to therefore stay agile and reflective of the real data infrastructure as it evolves. Same as the ER, attributes can be single or multi-valued. However, attributes can only be derived and not pre-defined, since mappings are created between existing data to entities. The goal is to be able to easily add nodes, which represent entities or attributes to entities from a variety of data sources.

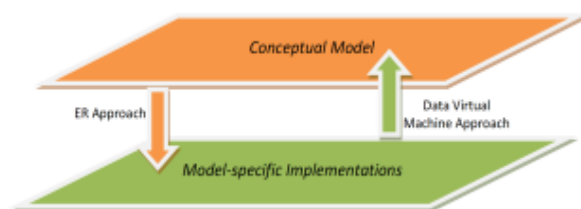


Figure 1: DVM-modeling vs traditional ER-modeling. Reproduced from [101]



Under traditional modeling, the first step is schema design, and then the creation of mappings to accommodate it. In contrast, in the DVM, we first define data processing tasks (DPT) and then the schema is produced. In the example below, the Customer is the entity and has multiple attributes such as age or gender. DPTs are computational processes that define the attributes and semantically accompany the mappings (edges) between them. This means that DPTs are not just retrieving data from the sources but rather defining the dynamic relationships between entities and attributes. Any computation that maps at least one value to an entity can be represented in the DVM. As a result, the output of a DPT should be at least two columns, for example an ID and a value (ID,value). They can also be intra or inter-organization. Some examples of DPTs are:

- An SQL query, such as “SELECT custID, age FROM Customers”. This query maps the customer entity, through the primary key custID to the attribute ‘age’, as shown in Figure 2.
- A MongoDB query
- A Cypher query
- Any program that reads from a flat or excel file
- A program assigning probability of churn to a customer

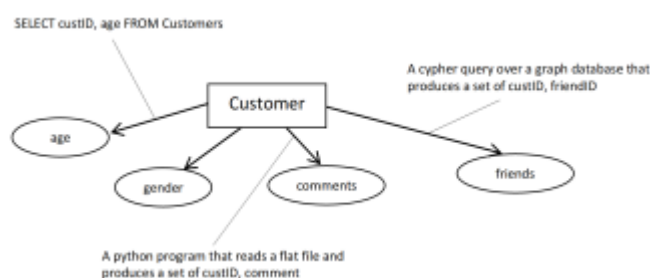
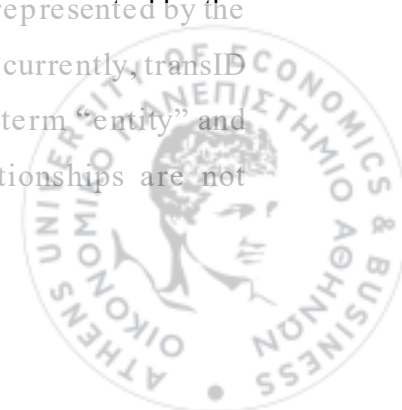


Figure 2: A customer entity with several attributes. Reproduced from [101]

All entities in the DVM can be represented by their primary key, like the custID attribute for the customer entity. Another feature of the DVM is that all entities are also attributes and vice versa. It is evident through the Figure 3, where transactions, represented by the primary key transID, are a multi-valued attribute of the customer. Concurrently, transID is an entity with its own attributes, such as amount and date. So, the term “entity” and “attribute” can interchangeably be used in the DVM, where relationships are not predefined, like in the ER.



This can be demonstrated through the SQL query “SELECT custID, age FROM Customers”, which maps the age attribute to a custID attribute but also one or more custIDs to an age value. Let us consider a DPT with a two-column output (u,v), where $u \in U$ and $v \in V$. It provides the mappings from U to V but also from V to U. The graph can therefore be characterized as bidirectional, meaning the all nodes in the graph are considered equal, having no hierarchy and all connections between them are symmetrical. A different color is used, as shown in Figure 3 below, to highlight the importance of nodes with multiple connections, degree higher than 1.

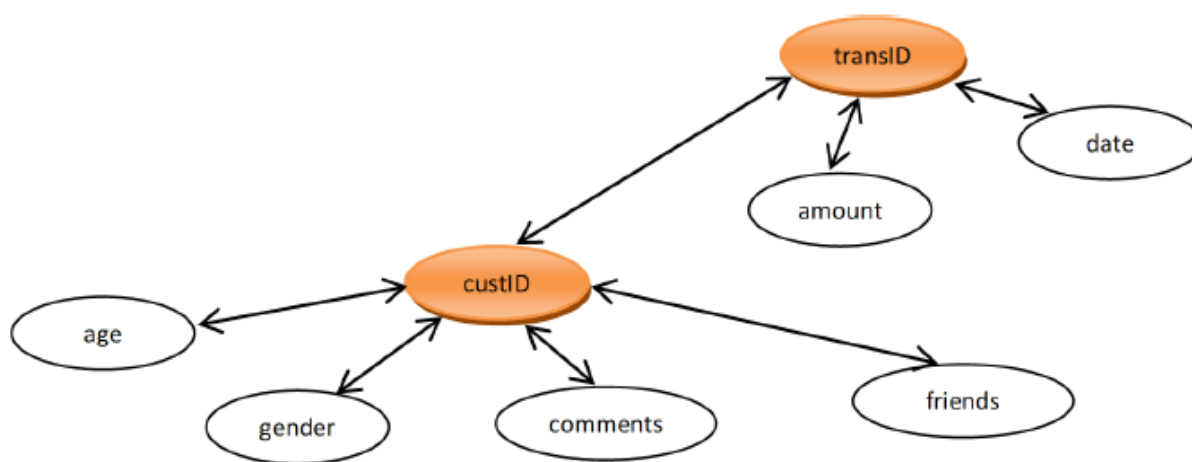


Figure 3: A simple DVM example. Reproduced from [101]

4.3 Key Concepts

The DVM's foundational elements are its nodes, edges, and key-list structures (KL-structures).

1. Nodes represent attribute domains, such as custID, transID, or Age in the DVM. The nodes are not restricted to one role; hence, they can be used either as an entity or attribute based on the analysis performed. This ensures maximum flexibility in schema representation.
2. Edges (Mappings) connect nodes and represent relationships. These come from the outputs of DMTs. For instance, a SQL query such as “SELECT custID, transID



FROM Transactions” would induce a mapping between the nodes *custID* and *transID*. This would carry edges of actual processes that link data attributes rather than pre-defined relationships.

- Key-List Structures (KL-Structure) are essentially a multi-map data structure, as shown in Figure 4. Every edge is associated with a KL-Structure (K), which is a set of key, list pairs $\{(k, L_k)\}$. The list of keys k of a KL-Structure K is symbolized as (k, K) . L_k is a list of elements, or the empty list. For every pair (k_1, L_{k_1}) , (k_2, L_{k_2}) which belong to K , k_1 is different than k_2 . All keys and elements are strings. For instance, the edge from *custID* to *transID* can be given the KL-structure representation of $\{(custID1: [transID1, transID2]), (custID2: [transID3])\}$. These KL-structures represent the basic blocks in the mappings of the DVM, without needing to reconfigure the general schema for dynamic extension.

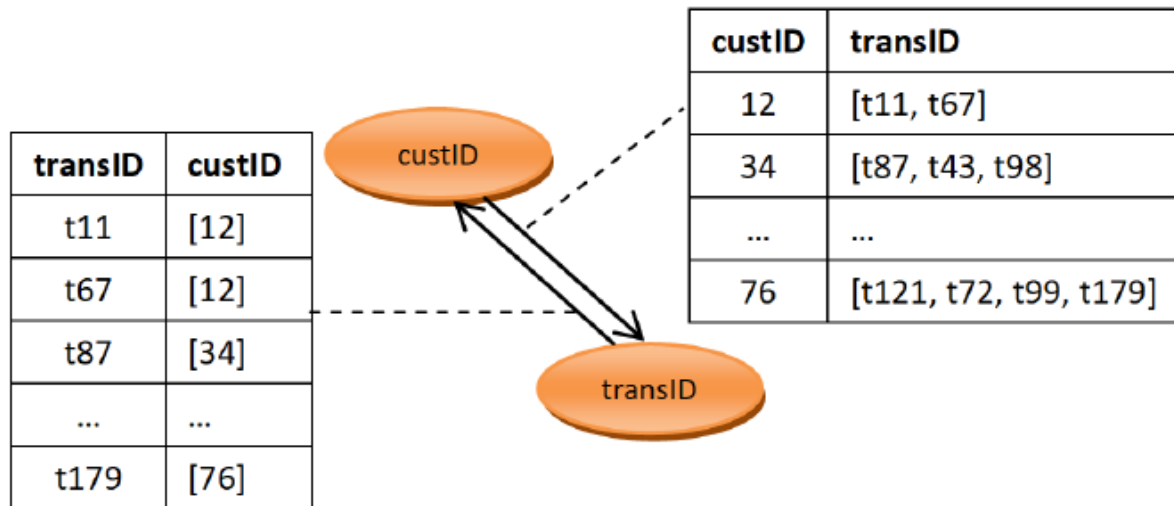


Figure 4: Key-list structures to represent edges of DVMs. Reproduced from [101]

The DVM is a multigraph of mappings (edges) between attributes (nodes). The mappings are created between the output pairs of values from DPTs.

- Let us assume n attributes A_1, A_2, \dots, A_n , each representing a node in the graph. The attributes are drawn from domains D_1, D_2, \dots, D_n respectively.
- We also have multiple DPTs (P), each one with an output multiset $S = \{(u, v) : u \in D_i, v \in D_j, i, j \in \{1, 2, \dots, n\}\}$. Two key-list structures can be derived from



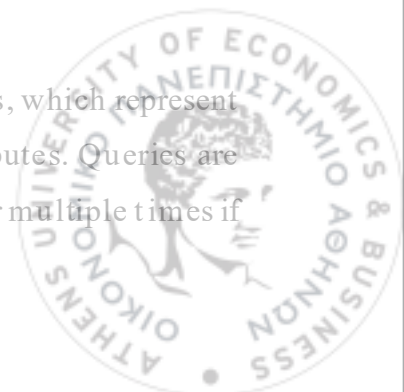
this output: $KL_{ij}(S)$ and $KL_{ji}(S)$. K is a set, and it is equal to $\{k : (k, v) \in S\}$, $\forall k \in K$, while L_k is a list $[v : (k, v) \in S]$.

- Let us take the first KL-Structure $KL_{ij}(S) = \{(k, L_k) : k \in K\}$. Then, for every u belonging to the set $\{u : (u, v) \in S\}$ we can define the list $Lu = \{v : (u, v) \in S\}$ and appended (u, Lu) to KS_{ij} . $KL_{ji}(S)$ can be defined in the same way, by using second constituent of the value pairs of S as key.
- A DVM is thus a multigraph $G = \{A, S\}$, where G are nodes corresponding to each attribute. Each DPT defines two edges $A_i \rightarrow A_j$ and $A_j \rightarrow A_i$, with the respective labels $KL_{ij}(S)$ and $KL_{ji}(S)$.
- Each edge $e : A_i \rightarrow A_j$ corresponds to a KL-Structure $KL(e)$, with schema (A_i, A_j) .

4.4 Query Language

The most commonly used query languages for dataframe formation are Python, R or Spark. Dataframes are tables built around an entity, which is usually the first column and contains the key (custID), while the rest are its attributes. There are multiple processes that can be performed on the attributes, such as aggregation, filtering or any other transformation defined by the user. Although the query language used in DVMs is not yet set in stone, it is critical to perform the above processes intuitively. The goal is to simplify attribute selection, condition expression and definition of transformations. The first way to achieve that is of course through methods already built-in the DVM. The second one, is through plug-in functions using various programming languages, which is called polyglotism. KL-Structure are the foundation of DVMs and can be stored and managed in a single key-value system, like Redis. However, the operators used for the manipulation of these structures are not limited to one programming language. This redundancy means that the same operation, for example summing of transaction amounts, can be executed in either Python or R, depending on user preference. The DVM query engine then uses the selected programming language to complete the operation.

Dataframe queries in DVMs are structured as trees, comprising of nodes, which represent attributes, and edges, which represent the mappings between the attributes. Queries are not necessarily subtrees of the DVM, the same node or edge can appear multiple times if



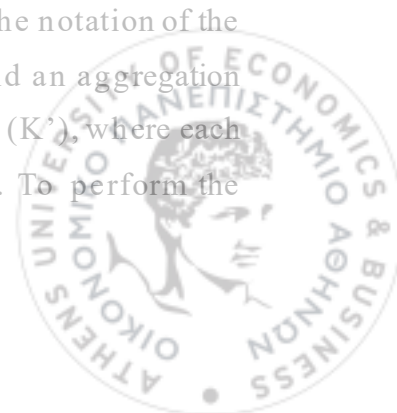
needed. For example, if a user wants to extract customer data, retrieving both total spending and average spending from the transaction amount attribute will make the amount node appear twice. Query evaluation requires the application of transformations on edges and also the combination of edges, either along a path across nodes or by merging attributes belong to the same root. The primary entity of interest is the root node of the tree, like the custID in an analysis about customers. The attributes related to the entity of interest are the child nodes. Through the nodes, the user can visually define and manipulate relationships. In the DVM, there are three main operators used for query evaluation and optimization, based on an algebraic framework. Their input is one or more edges (KL structures) and their output is another edge.

4.4.1 Operators

Transformation operators

Transformation Operators apply functions or conditions to existing KL-Structures and produce a new KL-Structure. This new KL-Structure can then be used for additional computations. There are three primary transformation operators in the DVM: Aggregation, which is the combination of multiple values of a list into a single value (e.g. sum of transactions), Filtering, the selection of elements that match a given criterion (e.g. filtering by age), and Mapping, which is used to transform each element in a list (e.g. text processing). A combination of the above can be used to perform more complex queries. For example, a retail company is interested in customer purchasing behavior analysis. They can use filtering to limit the result to specific date ranges, mapping to convert reviews into sentiment analysis values and aggregation to calculate the average amount spent by each customer.

- Aggregation: The aggregation operator in a DVM is used to summarize or reduce a list of values associated with a given key in a KL-Structure. The notation of the operator is $Aggr(K, f)$. It takes as inputs a KL-Structure (K) and an aggregation function (f) (e.g. sum, count). The output is a new KL-Structure (K'), where each key is mapped to a single aggregated value instead of a list. To perform the



aggregation, for each $k \in K$, $f(L_k)$ is computed on the list of values L_k and the result is stored as a single element list $L_k' = [f(L_k)]$ in K' .

The aggregation function f can be a built-in function:

- Min = smallest value
- Max = largest value
- Average = mean of all values
- Sum = total of all values
- Count = number of elements

Otherwise, custom functions can be written in different programming languages, like Python or R, such as $f(x) = \text{median}(x)$, supporting polyglotism.

The example in Figure 5 showcases the aggregation of total Transaction amounts per Customer. In this case, the input KL-Structure K contains “custID” and “Amount”, as shown below. After applying the aggregation functions sum and count we produce new KL-Structures K' and K'' accordingly.

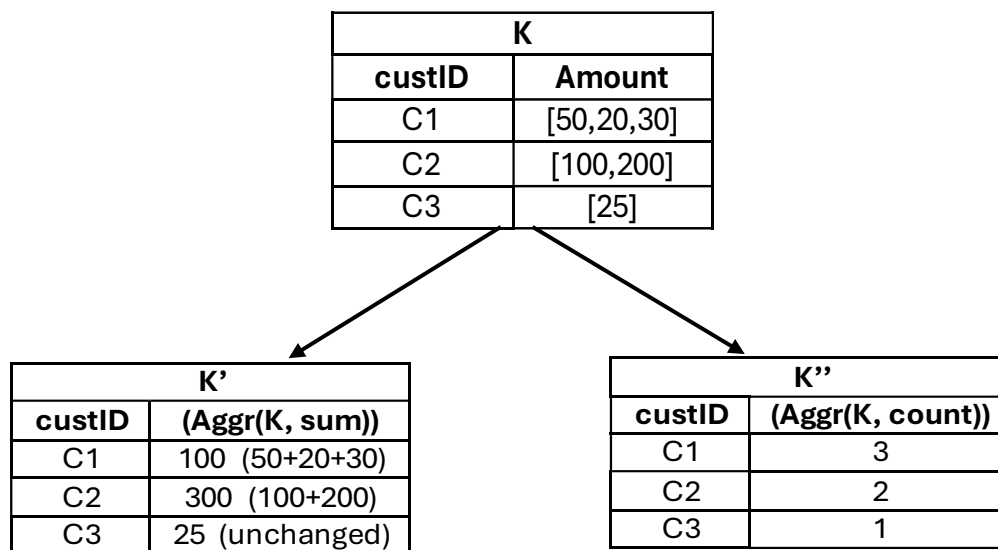


Figure 5: Aggregation Function example

- **Filtering:** The filtering operator in a DVM is used to select only relevant elements from a list based on a specific condition. The notation of the operator is $\text{Filter}(K,$



ϕ). It takes as inputs a KL-Structure (K) and a condition (ϕ), which is a Boolean function applied to each element of the list (e.g. rating > 5). It does not remove entire keys but rather returns a new KL-Structure (K') containing only elements from the list associated with each key that satisfy the condition. To perform filtering, for each $k \in K$, the pair (k, Lk') is added to K', where $Lk' = \{x \in Lk \mid \phi(x) \text{ is true}\}$. This means that K' is a filtered version of the original list K.

Let us use as an example the filtering of customer transactions by year, shown in Figure 6. The input KL-Structure K contains “custID” and the “Year” of each transaction. We are interested in all transactions performed in the year 2024.

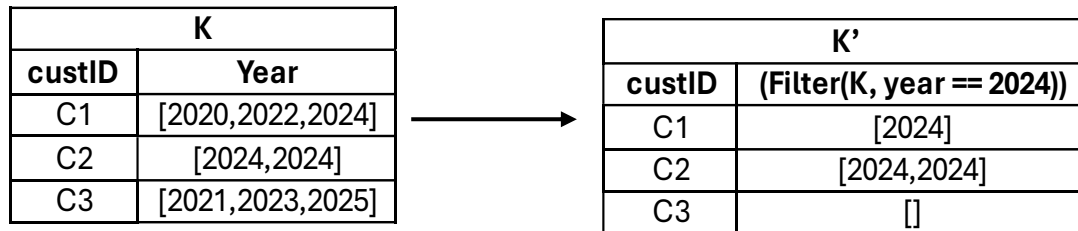


Figure 6: Filtering Function example

- **Mapping:** The mapping operator in a DVM is used to transform elements in a list by applying a function f to each element. It replaces each element with a modified version of itself. The notation of the operator is $\text{Map}(K, f)$. It takes as inputs a KL-Structure (K) and a function (f) that takes an element and transforms it. The output is a new KL-Structure (K'), where each element x in Lk is replaced by $f(x)$. To perform mapping, for each $k \in K$, the pair (k, Lk') is added to K', where $Lk' = \{f(x) \mid x \in Lk\}$. This means that each value in the list is transformed based on function f . Same as the aggregate function, mapping support polyglotism as it can be written in a variety of programming languages.

An example would be mapping reviews to sentiment scores, as shown in Figure 7. The input KL-Structure K contains “custID” and the product reviews “Review”.



Using a sentiment analysis function, the text can be converted to sentiment scores, +1 for positive, -1 for negative and 0 for neutral.

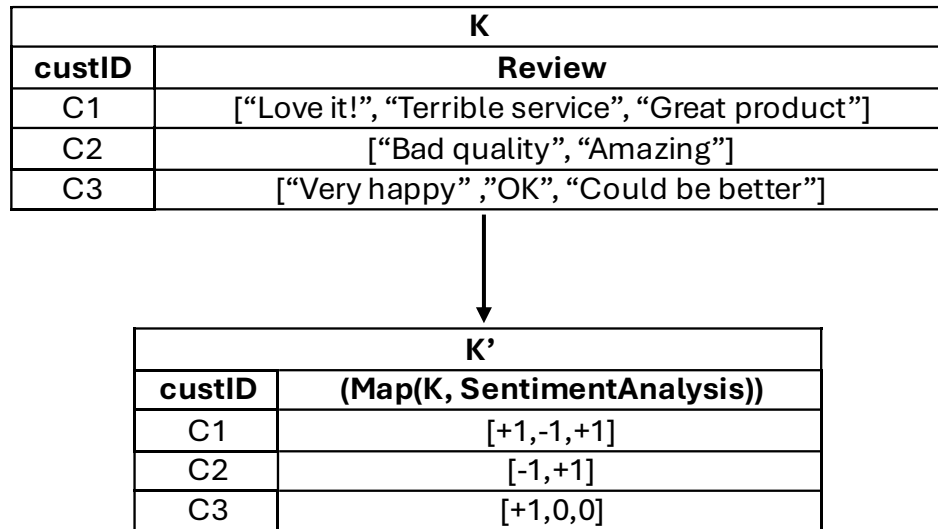


Figure 7: Mapping Function example

All transformation operators in DVM are unary, meaning they operate on a single edge. However, they can be combined sequentially to perform complex queries. An example use case is the identification of high-value customers. The question posed is "What is the average transaction amount in \$ for customers who have made at least one purchase over 500€?". To answer the question, we need to first perform filtering, then mapping and then aggregation:

1. Filter customer transactions to keep only the ones above 500€
2. Map the remaining amounts to a new currency (\$)
3. Aggregate the amounts to compute the average

Sequential transformations are performed, each one modifying the original dataset. Additional transformation can be added according to the analysis.



RollupJoin operator

The RollupJoin operator in the DVM is used for queries involving a path between attributes. Unlike transformation operators that modify lists within a single KL-Structure, RollupJoin connects consecutive edges in a path, as shown in Figure 8. It effectively combines data across multiple KL-Structures that share a common intermediate key. The references (IDs) are replaced with their associated values, thus allowing additional computations to be performed. The inputs are two KL-Structures (K1, K2) sharing a common key relationship. The first one contains a mapping from a primary key to an intermediate key, while the second maps that intermediate key to the final attribute. The output is a new KL-Structure (K), having the intermediate key replaced with the corresponding attribute. The notation of the operator is rollUpJoin(K1,K2), To perform RollupJoin, for each $k \in K$, the pair (k, L_k) is added to K, where $L_k = \bigoplus_{x \in \text{list}(k, K1)} \text{list}(x, K2)$, where \bigoplus represents list concatenation.

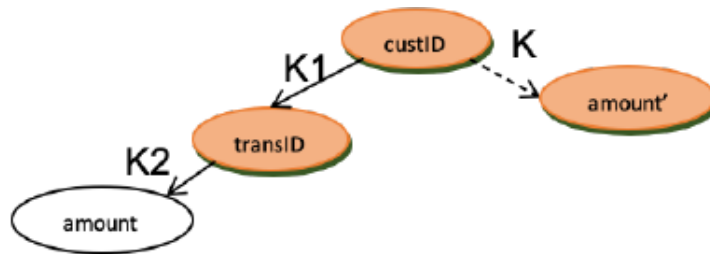


Figure 8: Joining two consecutive edges to one. Reproduced from [102]

A common example would be to associate transaction amounts with customers, as shown in Figure 9. The inputs KL-Structures contain custID to transID and transID to Amount respectively. The RollupJoin operator is used to create a KL-Structure containing custID and Amount, by replacing the intermediate key transID.

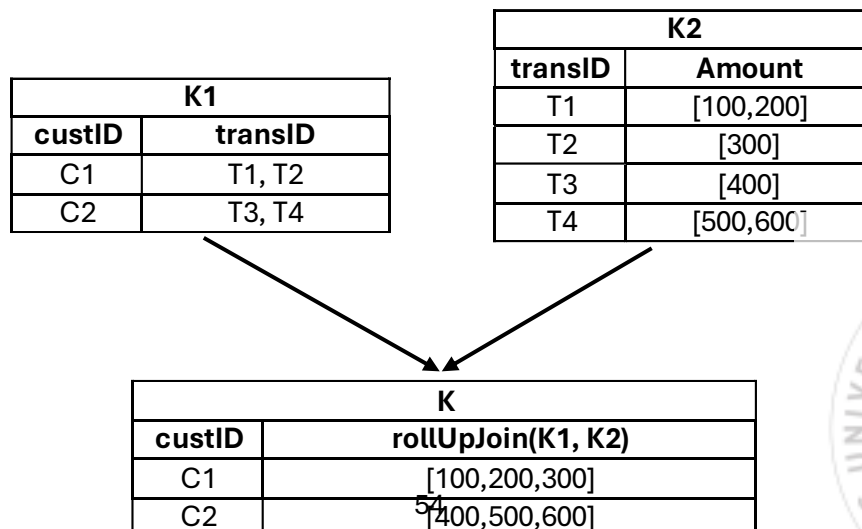


Figure 9: RollupJoin example



ThetaCombine operator

The ThetaCombine operator in DVM is used for the combination of multiple edges rooted at the same node into a new edge, while applying selection conditions on the keys. Projection concatenates multiple KL-Structures that share the same key domain into a single KL-Structure. Selection filters specific keys based on conditions involving multiple attributes. ThetaCombine allows the combination of child nodes in the DVM, while ensuring that KL-structures are maintained when applying conditions. The notation is $\text{thetaCombine}(O, S, \phi)$. The inputs are an output list O , containing the KL-Structures (K_1, K_2, \dots, K_n) that need to be combined, the selection list S , containing the KL-Structures $(K'_1, K'_2, \dots, K'_m)$ that restrict the keys in the result based on the selection condition which is a Boolean expression $\phi(k, l_1, l_2, \dots, l_m)$. The lists may overlap and either can be empty, but not both. The result is a new KL-Structure $K = \text{KL}(A \rightarrow B)$, where A is the root attribute and B is a new attribute created. To perform ThetaCombine, for each k in the intersection of all involved keys we evaluate the condition $\phi(k, \text{list}(k, K'_1), \dots, \text{list}(k, K'_m))$. If it is true and O is empty then L_k is the empty list, if it is true and L_k is not empty, all lists in O are concatenated: $L_k = \bigoplus_{i=1, 2, \dots, n} \text{list}(k, K_i)$. The resulting KL-Structure contains only the keys that passed the condition, with their lists merged. This operator takes the intersection of all involved key-list structures to determine the final key domain, similar to an inner join in traditional relational models.

A simple example is the retrieval of the age and gender of a customer, while filtering for females above 25 years old, as shown in Figure 10. Both age and gender are attributes of the custID. We first apply projection, by combining custID and age, gender into a single KL-Structure and then selection by filtering $\text{age} \Rightarrow 25$ and $\text{gender} = F$. In this case K_1 contains custID and Age and K_2 contains custID, Gender. The output list $O = \{K_1, K_2\}$ is the Age and Gender. The selection list S is also $\{K_1, K_2\}$ since we are applying conditions on both Age and Gender. The condition is $\phi(k, \text{age}, \text{gender}) = (\text{age} \Rightarrow 25 \text{ AND } \text{gender} = F)$.



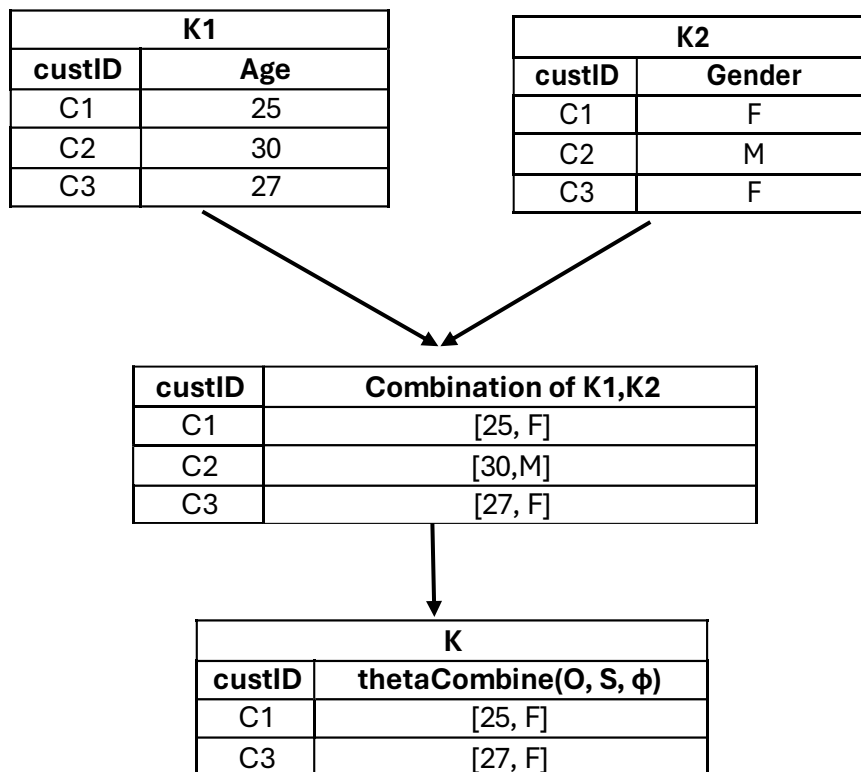
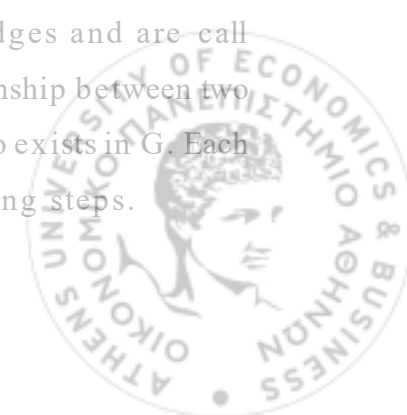


Figure 10: ThetaCombine example

4.4.2 Query tree

As already discussed, a DVM is a multigraph notated as $G=\{A,S\}$ where A represents the attributes and S is the schema, meaning the edges connecting the attributes. A dataframe query in DVM is structured as a tree Q , where each node represents an attribute of the DVM, and each edge defines relationships and transformations between these attributes. The following can be found in Q :

- (N: name) is a unique name of each node N within the query tree Q
- (N: label) is a label corresponding to an attribute in G . For example, a node $N1$ with $N1:\text{name} = \text{"customer_info"}$ and $N1:\text{label} = \text{"custID"}$ is uniquely named "customer_info" in Q and its label corresponds to the "custID" attribute of the DVM.
- (e:transformations) are lists of transformations applied to edges and are called transformation strings. Each edge $e:N \rightarrow N'$ represents a relationship between two attributes in Q . A corresponding edge label $\text{label}(N) \rightarrow \text{label}(N')$ also exists in G . Each edge can be transformed using a sequence of data processing steps.



- (N:selection) is the selection condition of each node N and defines whether a record should be included in the result. It can either be the value “TRUE” or a Python-like logical expression referring to the node N itself or any of its child nodes.
- (N:output) is a label that all nodes except the root have, which determines whether the node’s data will appear in the result. If “true” then the node and all its ancestors should also have “true” output labels.

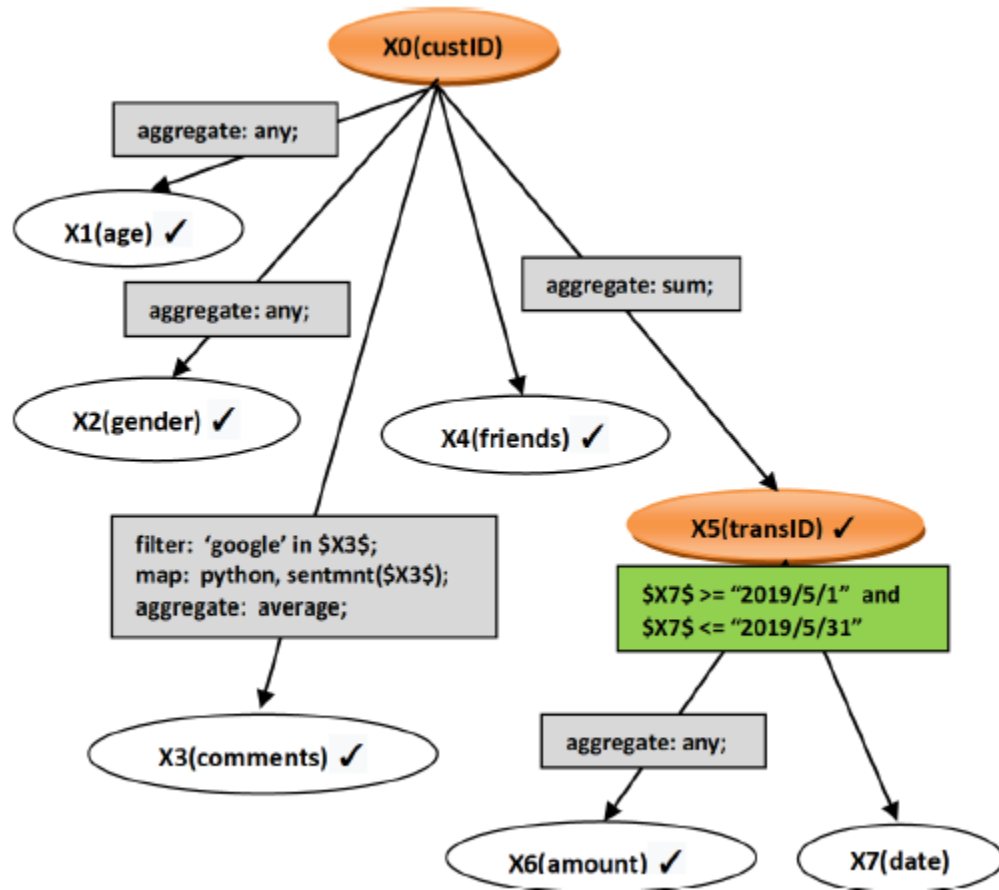
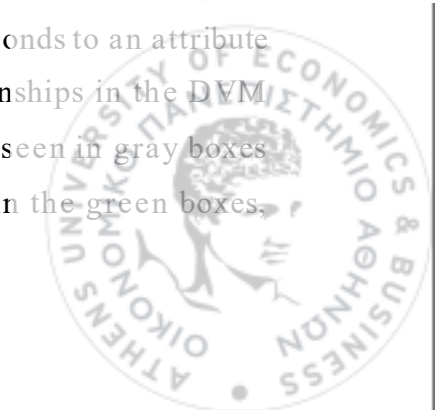


Figure 11: An example of a dataframe query. Reproduced from [102]

The tree represents a DVM query, as shown in Figure 11. Each node (ellipses) X0,X1,...,X7 represent attributes. Each node in the query tree corresponds to an attribute in the DVM. The edges (arrows) between the nodes are the relationships in the DVM schema. Transformation operators (filtering, mapping, aggregation) seen in gray boxes modify the data as it flows through the tree. Selection conditions, in the green boxes,



restrict which data is included. The output labels, signified by the checkmarks, indicate which nodes are included in the final output. The root node of the tree is the custID and every attribute is related to it. Below, in Figure 12, is a breakdown of the tree:

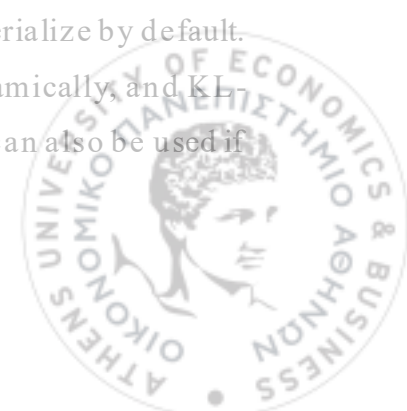
Node (N)	Label	Transformation / Selection Condition	Included in output
X0	custID	-	Yes
X1	age	Aggregate: any()	Yes
X2	gender	Aggregate: any()	Yes
X3	comments	Filter: 'google' in X3 Map: python, sentiment(X3) Aggregate: average()	Yes
X4	friends	Aggregate: any()	Yes
X5	transID	Selection Condition: date between 2019/5/1 and 2019/5/31	Yes
X6	amount	Aggregate: any()	Yes
X7	date	-	No

Figure 12: Breakdown of dataframe query

The query retrieves the age(X1), gender(X2), friends(X4) of each customer(X0). Aggregate any() means that if multiple values exist, only one of them is returned. Customer comments(X3) are filtered so that only the one mentioning “Google” are included, converted into sentiment scores using Python and then the average score per customer is calculated. Transactions(X5) are limited to May 2019 and the amount(X6) associated with them is included. The date(X7) is used only for filtering but not included in the output.

4.4.3 Query evaluation and optimization

The DVM is a virtual schema, meaning that KL-Structures do not materialize by default. When a dataframe query Q is executed, the edges are processed dynamically, and KL-Structures are populated accordingly. Pre-existing materialized edges can also be used if available to improve performance.

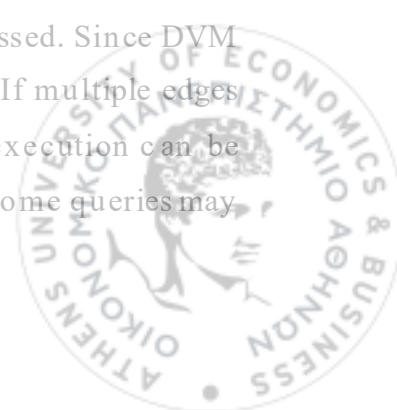


A bottom-up evaluation approach is followed for dataframe queries in a DVM to ensure efficient execution. Outgoing edges, except for the root node, are combined into a KL-Structure using `thetaCombine`. It essentially merges attributes related to the same entity. Additionally, adjacent edges are joined using `rollupJoin`. If an edge maps a key to an intermediate key, it is replaced with a direct mapping. This step eliminated intermediate lookups, reducing query complexity. The transformation defines in the transformation strings are also applied to the edges (filtering, mapping, aggregation). On the other hand, the root node is processed differently. The outgoing edges are either combined in a dataframe format (like Python Pandas or R DataFrame) where each row represents a key, or with a KL-Structure using `thetaCombine`, which can be added as a new attribute.

To summarize, the key functions used in query evaluation are:

- `keyCombine(A, B1, B2, ..., Bn)` is used to combines multiples edges ($A \rightarrow B_1, A \rightarrow B_2, \dots$) into a single key structure
- `applyTransformations(A, B)` executes filtering, mapping, and aggregation on the edge $A \rightarrow B$
- `rollupJoin(A, B, C)` replaces an intermediate key (B) with direct mapping ($A \rightarrow C$)
- `thetaCombine(B, C1, ..., Cm)` merges multiple child attributes of B into a new key-list structure. The output list only includes attributes where $C.Output = true$. The selection condition is inherited from the parent node (B)

After a dataframe query is evaluated, optimizations ensure efficient execution. A dataframe query is first converted into a key-list algebraic expression, which is then optimized. When a query requires an edge to be materialized there are several actions to ensure minimal computational overhead. To avoid repetitive calculations for edges that are frequently used, computations occur once and are stored. Additionally, instead of materializing all data and then applying transformations (like mapping and aggregation), filtering is executed first to reduce the number of records being processed. Since DVM queries are graph-based, optimization can be performed on the graph. If multiple edges originate from the same node but are independent between them, execution can be performed simultaneously, significantly reducing the execution time. Some queries may



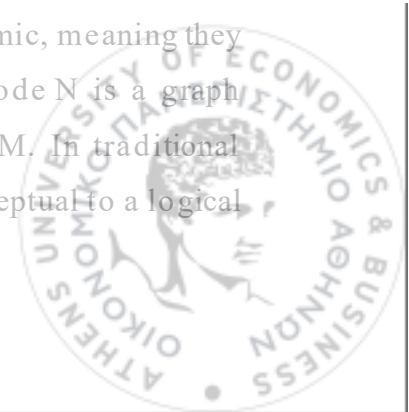
also contain repetitive computations. Instead of performing them in different parts of the query tree, common subexpressions can be evaluated once and reused. Rewriting a query modifies the structure of the tree without affecting the result. For instance, unnecessary steps can be eliminated, redundant functions can be replaced with direct mappings, selection queries can also be replaced with aggregation. All these minimize data movement, improving efficiency. Similar to SQL query optimization, DVM can follow relational algebra principles. Queries can be rewritten using alternative algebraic expressions that achieve the same result but are more efficient.

DVMs are ideal for non-technical users, as they support a visual formulation of dataframe queries. Both query evaluation and optimization techniques are critical for creating a flexible framework that can handle complex dataframe queries, in an intuitive way.

4.5 Any-Entity View and Model Polymorphism

In section 4, the DVM was introduced, highlighting its differences to traditional data management systems. Models, like the ER, are characterized by rigid schemas that are defined in the beginning. In contrast, the DVM follows a bottom-up approach, deriving the schema from the processing of the data. Thus, they are a more flexible adaptation of the ER model, allowing the schema to evolve along with the data. This means that the materialized schema is not concrete. Users can choose any node as the root, effectively reorienting the schema around it, which is called "Any-Entity-View". This is a great strength compared to traditional models, which restrict the analysis to a specific view. Data exploration, through DVM, can be performed from the perspective of customers, transactions, dates or even computed attributes like probability of churn.

In a DVM, all conceptual objects and relations are simplified into graph nodes and edges respectively. Nodes in the DVM can either be entities or attributes, depending entirely on the perspective of the query. The relationships between them are dynamic, meaning they are defined by the DTP applied by the user. The entity view of a node N is a graph originating from N and includes paths that end at a different node M . In traditional databases, the schema mappings are required to transition from a conceptual to a logical



schema. A key strength of the DVM is its ability to materialize schemas in logical formats, such as relational tables (work in progress) or JSON, which is known as model polymorphism. Multiple schema representations can be derived from the same DVM model, and a single dataset can be represented in different formats. This flexibility ensures compatibility with various downstream applications and data formats.

Below, in Figure 13, is an example of JSON documents derived from a DVM where custID, ChurnCateg, and Date is selected as an entity by the user.

```

{
  "custID": "920",
  "ChurnCateg": "3",
  "Comment": ["This is comment #1 of customer 920", "This is comment #2 of customer 920"],
  "transID_s": [
    {
      "transID": "t100326",
      "ATMCode": null,
      "Date": "2018-02-22",
      "Amount": "21"
    },
    {
      "transID": "t102556",
      "ATMCode": null,
      "Date": "2018-04-05",
      "Amount": "7"
    },
    {
      "transID": "t103175",
      "ATMCode": null,
      "Date": "2018-03-22",
      "Amount": "48"
    },
    {
      "transID": "t103560",
      "ATMCode": "129",
      "Date": "2018-08-24",
      "Amount": "53"
    },
    {
      "transID": "t105027",
      "ATMCode": null,
      "Date": "2018-06-01",
      "Amount": "45"
    },
    {
      "transID": "t105079",
      "ATMCode": null,
      "Date": "2018-10-19",
      "Amount": "29"
    },
    {
      "transID": "t106846",
      "ATMCode": null,
      "Date": "2018-08-27",
      "Amount": "10"
    }
  ]
}

{
  "ChurnCateg": "0",
  "custID_s": [
    {
      "custID": "2",
      "Comment": ["This is comment #1 of customer 2", "This is comment #2 of customer 2"],
      "transID_s": [
        {
          "transID": "t100387",
          "ATMCode": null,
          "Date": "2018-07-19",
          "Amount": "9"
        },
        {
          "transID": "t101147",
          "ATMCode": null,
          "Date": "2018-11-02",
          "Amount": "40"
        },
        {
          "transID": "t101321",
          "ATMCode": null,
          "Date": "2018-03-17",
          "Amount": "21"
        },
        {
          "transID": "t102753",
          "ATMCode": null,
          "Date": "2018-08-25",
          "Amount": "52"
        },
        {
          "transID": "t105187",
          "ATMCode": null,
          "Date": "2018-05-12",
          "Amount": "20"
        },
        {
          "transID": "t106093",
          "ATMCode": null,
          "Date": "2018-09-03",
          "Amount": "6"
        },
        {
          "transID": "t109120",
          "ATMCode": "174",
          "Date": "2018-08-27",
          "Amount": "10"
        }
      ]
    }
  ]
}

{
  "Date": "2018-01-01",
  "transID_s": [
    {
      "transID": "t100712",
      "ATMCode": null,
      "Amount": "60",
      "custID_s": [
        {
          "custID": "584",
          "ChurnCateg": "4",
          "Comment": ["This is comment #1 of customer 584", "This is comment #2 of customer 584"],
          "City": "Seattle",
          "Gender": "F",
          "Age": "20"
        }
      ]
    },
    {
      "transID": "t100818",
      "ATMCode": null,
      "Amount": "54",
      "custID_s": [
        {
          "custID": "550",
          "ChurnCateg": "1",
          "Comment": ["This is comment #1 of customer 550", "This is comment #2 of customer 550"],
          "City": "Boston",
          "Gender": "F",
          "Age": "39"
        }
      ]
    },
    {
      "transID": "t100863",
      "ATMCode": null,
      "Amount": "57",
      "custID_s": [
        {
          "custID": "639",
          "ChurnCateg": "3",
          "Comment": ["This is comment #1 of customer 639", "This is comment #2 of customer 639"],
          "City": "Miami",
          "Gender": "M",
          "Age": "39"
        }
      ]
    }
  ]
}

```

Figure 13: Creating JSON documents for different DVM's nodes, conceived as entities. Reproduced from [100]

4.6 Practical Implementation: DataMingler Tool

DataMingler is a prototype tool designed for end-to-end processing and analytics. It provides a graphical interface that users can interact with and visually manage DVMs (Data Virtual Machines). Through DataMingler, users can integrate data, create graphical schemas, construct queries intuitively and even export data in a JSON-like file. It is not only suitable for experienced users such as data engineers or data scientists but also for people lacking technical knowledge. Figure 14 showcases DataMingler's architecture.



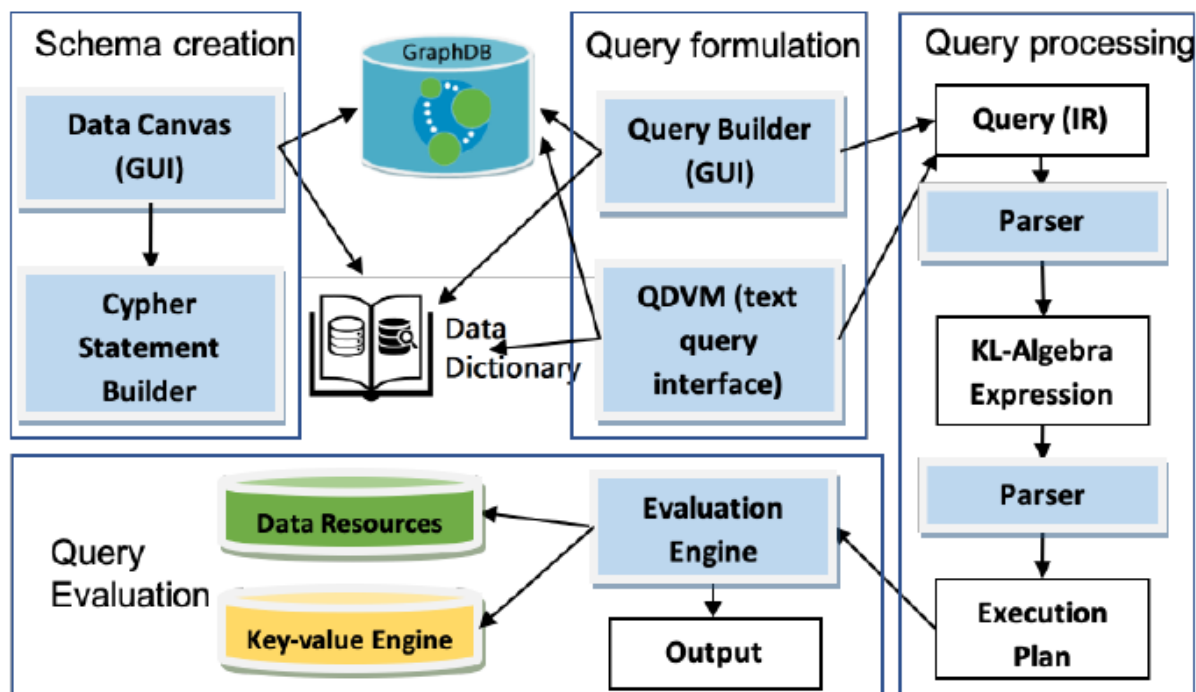
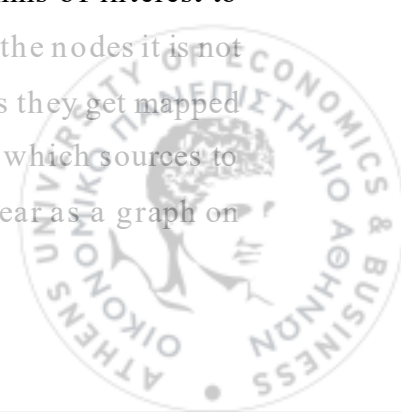


Figure 14: DataMingler's Architecture. Reproduced from [99]

A module called Data Canvas allows users to create and manage DVM graphs by defining the nodes and edges between them, as shown in Figure 15. The data sources supported range from relational databases to flat files and standalone programs like Java or Python. The data source dictionary is available in XML and includes descriptions of the resources. Users can perform multiple actions in this environment in a simple and intuitive way. New data sources can be added with ease by both technical and non-technical users. The first can directly add through an XML file, while the latter can opt for manual configuration directly on the left pane. The configuration process is quite simple. The first step is to decide on the type of data source, either database, excel, csv or program. Then, some additional basic information is required, such as database credentials, file path, delimiter or headings. After, the user can pick the specific columns of interest to create the according nodes in the graph. To create connections between the nodes it is not necessary for columns to have the same names across sources, as long as they get mapped to the same node. Upon successful configuration, the user can choose which sources to utilize according to their requirements. As sources are used, they appear as a graph on



the right panel, with edges formed according to the configuration. The graph can be easily manipulated, by moving nodes to a shape that is visually suitable. By clicking on nodes and edges, the user can view additional information of the element. The DVM graph is stored in a Neo4j database. All nodes have properties such as Name and Description and edges have Data source, query string if applicable and the position in the output result.

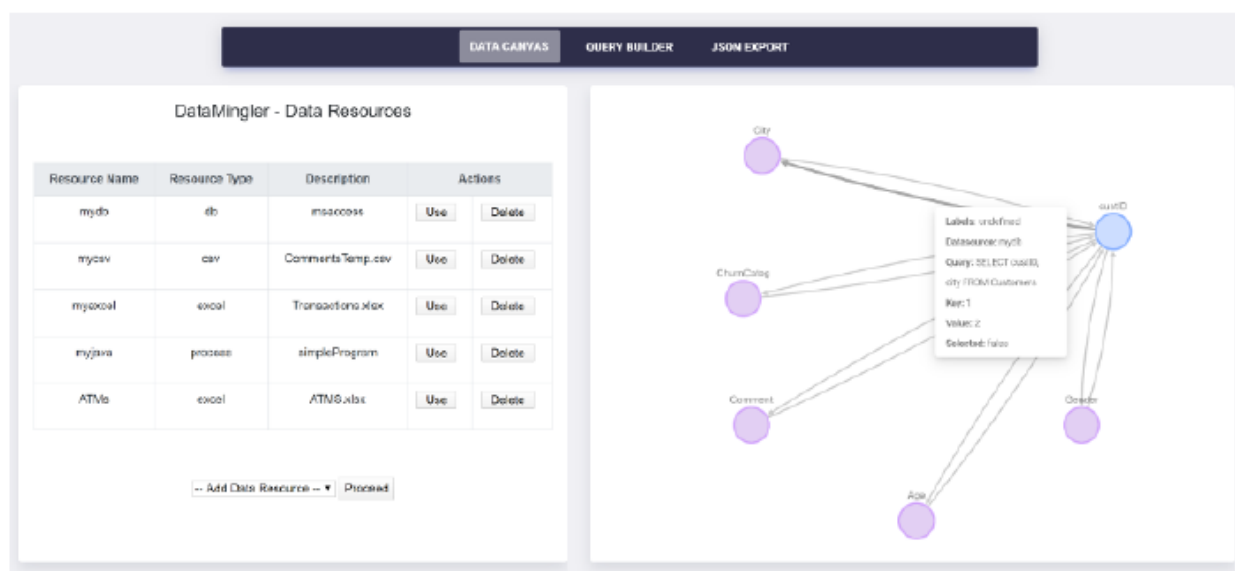
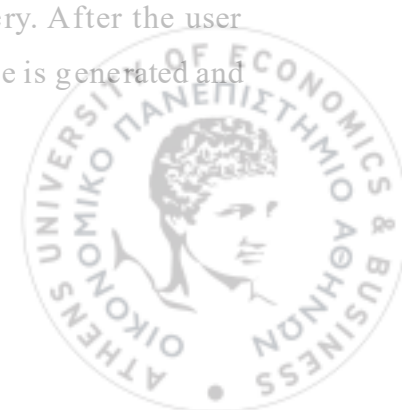


Figure 15: Data Canvas: Managing DVMs. Reproduced from [99]

After integration, the users can navigate to the Query Builder module, as shown in Figure 16. On the right, the query name is required and is subsequently stored for easy retrieval. The left pane includes all nodes and edges previously present in Data Canvas. Here, the user is able to select all or a subset of the nodes to include in the query. Queries can be constructed either by writing or by using a drag-and-drop interface. Multiple programming languages can be used, depending on user preference. All queries are internally represented in an XML-based intermediate format and then reformatted to KL-algebraic expressions for execution. After this, the query optimizer handles the proper execution. The results are then returned as dataframes. Below is an example of the module, where the user can see the available data source and the query. After the user selects a node in the DVM, a breadth-first search tree rooted at that node is generated and the user can define transformations and mark output nodes.



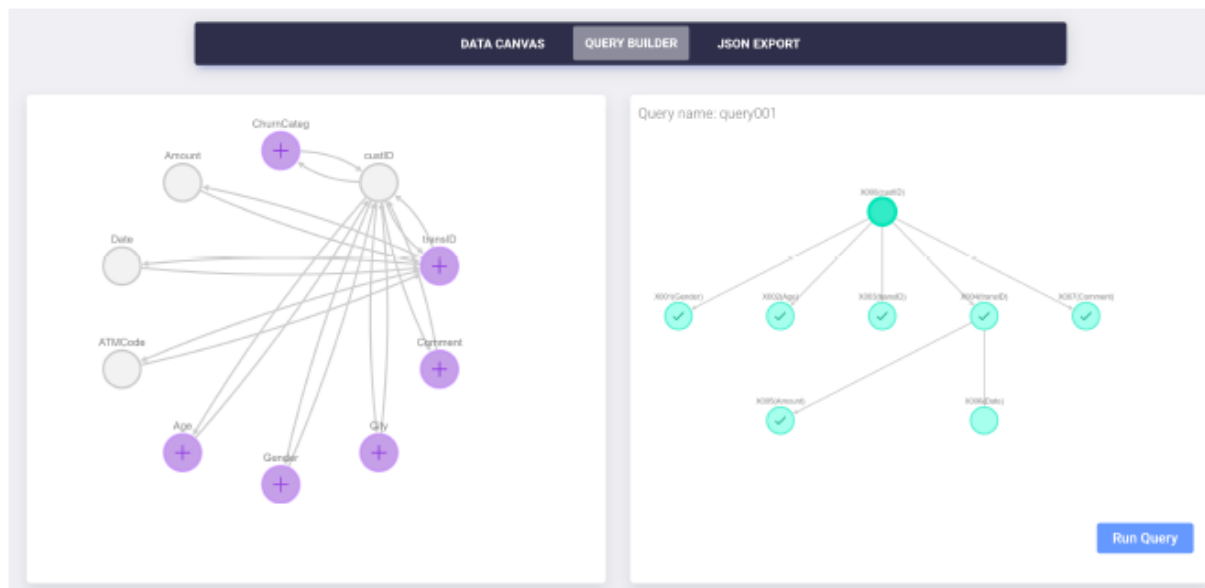
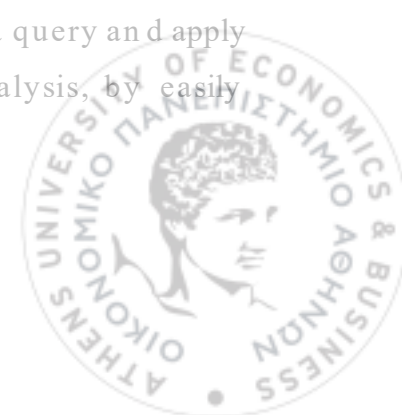


Figure 16: QueryBuilder: Dataframe queries. Reproduced from [99]

Finally, users can export the schema views in JSON format, making it easy to generate model-specific outputs. Documents are created based on the structure defined by the user.

Data engineers can use DataMingler similarly to an ETL tool. In Data Canvas, they can integrate data from various sources and visually map them into the DVM graph. They can greatly benefit from the dynamic schema, as reconfiguration is not required even when adding a new source. The mapping can be directly defined on the edges of the graph. For example, a unified schema can be created by mapping customer transactions from a relational table to customer sentiment using a Python model. The data engineer can additionally incorporate extraction rules and transformations to the schema. To improve performance, frequently used datasets can be predefined.

Data scientists wish to derive insights from their data, even without database expertise. With Query Builder even complex queries can be visually created. So, instead of writing multiple SQL joins, they can just drag and drop relevant nodes into a query and apply transformations. They are also able to experiment with their analysis, by easily reorienting the model when choosing a new root entity.



In compliance with the GDPR, data subjects can utilize DataMingler as a self-service data portability and integration tool. Data subjects are able to export and transfer their data between platforms using DVMs. For example, an individual may download data from various service providers such as banks, telecoms and social media in a formatted document. In DataMingler, they have the option to reorient their view, from a customer-centric to a service-centric. They can also connect the data from multiple companies and create custom reports or personal analytics, such as a combination of health records with fitness app data. The data exported are also compatible with different platforms.

Data protection officers can also use DVMs to assist in data regulation compliance. They provide a clear, unified view of how data is connected and stored so that the flow of the data is traceable. By incorporating metadata, they will be able to trace data provenance and consent as well.

4.7 Conclusion

The DVM is a leap forward for data virtualization because it adds dynamism and flexibility to schema modeling and query design. It brings structured and semi-structured data together by abstracting relationships in a single graph structure, accessible to users with varying technical expertise. With advancing analytics environments, DVM gives a robust and flexible foundation for data exploration and manipulation in a cohesive and intuitive manner.

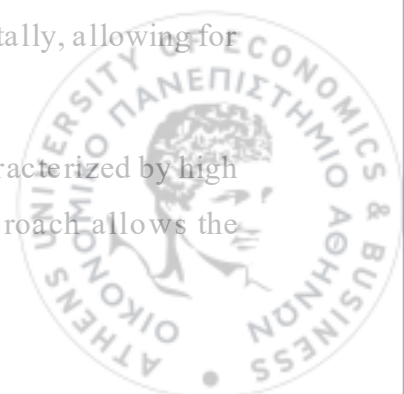


5. DVM: Data Portability Applications

The DVM framework addresses the challenges in modeling for data portability, discussed in previous sections, by offering ease of use and understandability with a graph-based model; scalability and extensibility by dynamic schema generation and support of heterogeneous data sources; interoperability through unified data representation and export capabilities; governance and compliance through traceability, and user empowerment; security facilitation by seamless integration with existing systems, while segmenting and customizing views on the data. The DVM facilitates these capabilities through tools such as DataMingler, an algebraic query evaluation framework, using its flexible architecture centered around graphs.

The DVM achieved both understandability and usability through several of its key features that differentiate it from traditional models and integration techniques. Complex data infrastructures are simplified via its graph-based conceptual model, allowing the user to visually navigate through its intuitive interface. It follows an entity-relationship approach which is pretty intuitive for technical and non-technical users, just like the Entity-Relationship model. Entities and relationships are represented as nodes and edges in the graph respectively, rendering the model visually accessible. DVM is also capable of "reorienting" around any node, such as age or churn category. The user can study and manipulate data from any perspective, such as single-customer view or transaction-centric. It therefore eliminates the complexities of rigid relational tables and predefined hierarchies. Through Data Mingler, users can also visually create, manipulate and query DVMs with ease. Schema building is done by drag-and-drop, and any modification performed does not affect existing relationships. Deep technical knowledge is not a requirement for interaction with the model, as the query formulation can be done using built-in functions or with the user's preferred programming language, in a declarative way. Interactive attribute selection, filtering and transformation eliminated the need for complex and manual query construction. The queries are built incrementally, allowing for additional refinement based on immediate insights.

DVM's design makes it suitable for modern analytics environments, characterized by high heterogeneity and evolving requirements. Its bottom-up schema approach allows the



DVM to automatically adapt to new data sources and processes. Attributes are derived directly from DPTs and not predefined structures. Addition of new DPTs automatically incorporates new nodes and edges within the model, so it evolves dynamically when new data are introduced, or existing ones become unavailable. As already outlined, with each query the user is able to reorient the schema around any node, using the same DVM. The graph is also bidirectional, meaning that all entities can be attributes and vice versa and all edges are symmetrical. The need for reconfiguration every time requirements change is therefore eliminated. DVM is also fit for scaling across various systems and data formats as it handles structured data (SQL), semi-structured data (JSON), unstructured data (text files), transient data (streams) and processes that produce output. The queries also rely on the algebraic framework and can be executed concurrently on multiple nodes. With operators like rollupJoin and thetaCombine, performance can be optimized even when data is growing. Also, it has the ability to materialize to various logical models, such as relational or semi-structured JSON, enabling scalability across different analytical requirements.

DVM inherently facilitates linkability and interoperability through its design. Attributes and entities are represented as nodes in a graph, connected by edges. The graph like structure, by nature, supports linking data across several domains. It also allows feature selection and attribute discovery via relationships across connected nodes, enabling data linking and navigation in analytics workflows. Because of its bottom-up approach, new data can be added without requiring reconfiguration. The relationships between entities are preserved through the DPTs, ensuring that data remains linked even if the structure changes. It also provides an any-entity-view, allowing the user to reorient the schema according to the requirements of the analysis. So, the existing relationships can be reviewed in a different context. In the DVM, metadata are also directly embedded within the graph. Users can access not only data, but also their context, origin and transformations. This is particularly useful for data lineage tracking. The model additionally supports SQL, NoSQL, flat files, excel spreadsheets and program outputs, allowing interoperability with diverse systems. Users can build their own data models with data from multiple domains. Through DataMingler, the export of nodes and their connected paths in formats such as JSON documents is enabled, for interoperability with



external systems and models. Data sharing is therefore achieved easily and fast, in a standardized way.

DVM provides features in line with regulatory compliance and data governance. As outlined above, metadata are directly embedded into the graph of a DVM, including information on origin, context and transformations. This data lineage tracking capability can fulfill the requirements of many regulations. Through Data Mingler, the user can additionally create different views from the same DVM. It is evident that regulations and standards are not aligned between regions or sectors. Different views can be used to accommodate contradicting requirements within the same model. The same regulator or auditor can also query the data from different compliance perspectives by reorienting the schema. New requirements in the regulatory landscape can also be easily incorporated into a DVM due to its highly flexible design. Although consent management has not been built in DVMs, it can be integrated into its conceptual framework with the addition of data-sharing restrictions or user-specific access controls. Users are also able to choose which information to export and share, such as a subset of the original schema. The data can also be exported in various formats, fulfilling GDPR's data portability requirements.

DVMs do not currently have explicit controls for security or privacy, which are of course an important part of regulatory compliance and efficient data portability in general. An organization would have to implement additional layers, such as encryption or access management, to secure data represented in a DVM that include sensitive personal or organizational data. DVM can achieve data segmentation by defining explicit mappings between entities and attributes for limited access to relevant subsets of data. DVM's graph design supports different views for different roles, which makes sure that sensitive data is only available to the privileged users. It also does not alter the underlying storage mechanisms, allowing organizations to rely on existing security and encryption protocols for data storage and transmission.



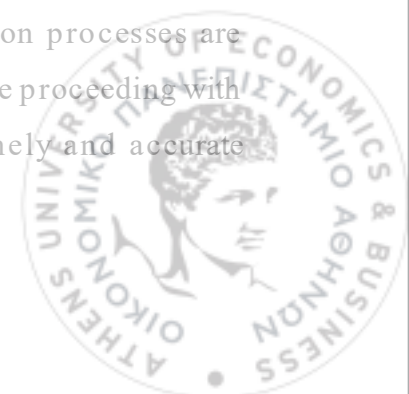
6. DVM: Use Case

6.1 Introduction to data management for regulatory reporting

The asset management industry is both heavily regulated and intensively data driven. Seamless data management is therefore critical to ensure compliance, optimize portfolio performance, and prevent financial or operational risk. In my line of work as a regulatory reporting analyst in a FinTech company, the importance of frictionless data integration and transformation cannot be overemphasized. Our customers, asset managers, find themselves in a regulatory ecosystem where they are constantly under pressure to produce accurate and timely reports, which require consolidations of massive amounts of financial information. Failure to achieve these expectations has the potential to incur regulatory fines, reputational damage, and even loss of money, therefore making efficient management of data extremely critical.

To address such challenges, asset managers rely on fintech technologies to achieve rapid, real-time, and dynamic data processing. Their data are characterized by volume and variety, such as portfolio information, market feeds, transactional records or regulatory reports. To handle these, a platform is needed that offers accuracy and adaptability. Without robust data integration capabilities, asset managers are vulnerable to inefficiencies, inaccuracies, and delays risking regulatory compliance as well as business decision-making. This highlights the key role that fintech vendors must play to supply scalable, next-generation solutions to support data workflows and enhance the accuracy of reporting.

Regulatory reporting analysts must deal with the complexities associated with handling large, heterogeneous data sets originating from diverse sources. The data sets have differences in structure, format, and granularity, rendering their harmonization and integration challenging. Extensive data transformation and validation processes are required to deal with inconsistencies, redundancies or data gaps, before proceeding with reporting and decision-making. Investment management requires timely and accurate



reporting and therefore organizations must adopt advanced data management methods that facilitate transparency, efficiency, and responsiveness.

Approaches to Data Management

One approach used by data analysts for data processing and aggregation is to extract them from various sources and manually combine them in spreadsheets. Manual processing is accessible to all because no programming skills or advanced technical knowledge are required. Nevertheless, while entering or modifying data, users may make mistakes, leading to inconsistencies, redundancy, and calculation errors. The complexity also grows exponentially as volumes grow, making this solution non-scalable. The above can have a negative impact on regulatory reports and decision-making. In addition, ensuring data accuracy and consistency across reports is always a challenge since even small inconsistencies can lead to compliance problems, audit failure, and reputation loss. With the regulations becoming stricter and data becoming more complex, manual processing is becoming obsolete, and thus automated and scalable solutions are necessary.

Another approach that can be employed by more advanced organizations is ETL (Extract, Transform, Load) technology and relational databases. Through these, data can be loaded automatically, providing structured data flows for business intelligence and regulatory reporting. These solutions greatly improve accuracy, consistency, and efficiency compared to manual processing, but they also have their disadvantages. They require a major upfront investment in infrastructure, have recurring maintenance costs, and require an IT support team for system implementation and refinement. Additionally, as outlined in previous sections, traditional relational databases enforce rigid schemas that predefine the structure and relationships of the data. As a result, the ability to introduce additional data sources or respond to changing regulatory requirements is limited. This lack of flexibility prevents asset managers from being able to analyze data ad hoc, requiring the intervention of IT for schema alterations, new mappings of data, and query optimizations. Therefore, these kinds of systems, while powerful, are often not as responsive as the ever-changing landscape of the financial world, where adaptive regulatory reporting and instantaneous analytics become increasingly critical.



Challenges in Regulatory Reporting

Regulatory reporting analysts are supplied with data from various sources that vary in structure, storage format, or level of complexity. The input data sets can be structured formats, such as SQL databases and Excel spreadsheets, semi-structured such as CSV, XML, or JSON files, and unstructured data such as PDFs, emails, and text files. The challenge is how to consolidate the above into a single and consistent format suitable for further analysis. Inconsistencies in naming conventions, missing values, various data schemas, and varying update frequencies also make integration even more difficult. Without an efficient method for data standardization and consolidation, organizations may not be able to produce the report in time, incur increased operational costs, and increased risk of inaccuracy, leading to compliance issues and bad decision-making.

Another critical element of regulatory compliance and operational efficiency is the assurance of accurate, complete, and reliable datasets. Even slight inconsistencies or errors in data can have profound consequences, leading to inaccurate financial reporting, regulatory penalties, and loss of client confidence. Errors occurring from manual processing or rigid ETL pipelines can cause misreporting, which requires significant time and resources to repair. Further, disparities between different sources of data, such as unmatched identifiers, missing data, or outdated records, can be significant hurdles in data reconciliation for audit and regulatory reporting. Only through deployment of dynamic and automated solutions can data accuracy and consistency in all reporting functions be guaranteed.

Traditional systems rely on predefined schemas and report formats, not suitable for dynamic, on-the-fly data analysis. However, the data landscape is not static as inclusion of new data is often required or an analysis to be conducted from a different angle. Modifying ETL pipelines, adjusting database schemas, or building SQL queries are all highly technical processes that not everyone can execute. Today's data management systems therefore prevent analysts from generating new insights or responding to new regulation requirements. This of course further contributes to operating expense and reliance on resources, like IT personnel. Under demanding financial circumstances, the



speed of decision-making is imperative. The restrictions outlined above can negatively impact the responsiveness and agility of fintech vendors.

New compliance, reporting requirements, and jurisdictional regulations emerge on a regular basis causing changes in the regulatory landscape of the financial industry. Companies are required to comply to updates from regulatory bodies such as the SEC, ESMA, or local financial authorities with typically short implementation deadlines. Data integration platforms built upon rigid schemas and rigid workflow templates are not equipped to keep up with these changes, as they amount to labor-intensive and costly reconfigurations. Failure to adopt a more agile and responsive data management solution will have organizations falling behind in compliance, facing fines and penalties, and ruining their reputation. Dynamic adaptation of data pipelines, introducing new reporting fields, and automated compliance updates are essential to regulatory responsiveness and operational resilience.

Traditional data management systems are further strained by the expansion of businesses, with the amount and complexity of their data exponentially increasing. Older systems have limitations in storage, processing power, and database architecture, making scaling inefficiently. These restrictions lead to slowing query performance, delayed regulatory reporting, and increased operational costs. Data expansion, new data sources, and real-time processing of high-frequency data should be anticipated by businesses. Scalable solutions, with modern architectures, able to adjust to shifting data needs offer effective workflows and sufficient analytical capabilities.

Conclusion

As financial data environments grow more complex, asset managers are expecting dynamic, intuitive, and agile data integration platforms from their financial vendors. These systems should be capable of receiving disparate data sets and connecting them effectively, supporting different types of analysis, both structured and ad-hoc, while at the same time ensuring data integrity. The goal is not just to achieve compliance, but also



improving operational efficiency, enabling faster decision-making, and generally enhancing data governance.

DataMingler, a graphical interface for Data Virtual Machines (DVMs), is a revolutionary new alternative to current data management processes. In this section, we will demonstrate how DVMs address real-world data integration problems by introducing a schema-flexible, scalable solution that supports seamless joining of heterogeneous data sources without requiring extensive ETL configurations. We will show how fintech vendors can successfully structure the datasets of their clients, execute dynamic queries without programming expertise, and extract valuable insights with minimal technical overhead.

6.2 Integrating Data in Data Mingler

For the use case, we utilized DataMingler to integrate and link data sets from four diverse sources in a single view that simplifies complicated data joins and renders analytical potential more accessible. Each source plays a critical role in the processing of asset management data:

- A relational table, from a database, containing portfolio information. There are multiple columns included such as portfolio ID, issuer, currency, and country code. This is the foundation for structuring investment portfolios in a way that each portfolio is properly classified and linked to relevant financial data.
- A relational table, from the same database, containing information on countries. It includes columns such as country codes and their corresponding country names. In combination with other data sets, this information can provide geographic classification. The analyst can then derive useful insights, such as country-level risk or regional exposure.
- A CSV of holdings with columns such as portfolio ID, security ID, market value, price, and quantity. This information can be used for asset allocation monitoring, fund performance measuring, and accurate reporting of investment positions.



- An Excel sheet including transaction information by security ID with trade dates. Transactional data provides insight into the history of trade executions, useful for liquidity tracking and compliance monitoring for audits.

Step 1: Data resource set-up

With the Data Canvas module, we configured the above sources of data into an easy-to-use interface. Users simply need to select the appropriate resource type and fill in the required connection information, as shown in Figures 17-19. By contrast to typical database systems imposing rigid schemas and requiring pre-processing of data before ingestion, DataMingler allows users to work with raw, heterogeneous data in a more flexible way. This ability is particularly valuable to fintech providers since asset managers are likely to give data in fragmented forms from multiple third-party sources, and they require a solution that can aggregate, clean, and standardize the data with minimal pre-configuration.

The screenshot shows the 'DataMingler - Add Data Resource' window. At the top, 'DB System' is set to 'SQL Server'. Below, there are several input fields: 'Source Name' (empty), 'Database' (empty), 'Connection Str.' (containing 'Server= X.Database=Use Case;Integrated Security=True;'), 'Username' (containing 'username'), 'Password' (empty), and 'Database' (containing 'Use Case'). A 'Create Data Resource' button is at the bottom.

Figure 17: Addition of Database resource

The screenshot shows the 'DataMingler - Add Data Resource' window. 'Source Name' is 'Excel'. 'File Path' is 'C:\Users\User\Documents\use case'. 'Filename' is 'Transactions'. 'Sheet Name' is 'Sheet1'. 'Headings' is set to 'Yes'. A 'Create Data Resource' button is at the bottom.

Figure 18: Addition of Excel resource

The screenshot shows the 'DataMingler - Add Data Resource' window. 'Source Name' is 'CSV'. 'File Path' is 'C:\Users\User\Documents\use case'. 'Filename' is 'Holdings'. 'Delimiter' is '|'. 'Headings' is set to 'Yes'. A 'Create Data Resource' button is at the bottom.

Figure 19: Addition of CSV resource



Step 2: Creating the DVM

Once the resources are defined, they are displayed on the left-hand panel of the module, where they could optionally be enabled or disabled as needed, as shown in Figure 20.

Database	db	sqlserver	Use	Delete
Excel	excel	Transactions	Use	Delete
CSV	csv	Holdings	Use	Delete

Figure 20: Available resources in Data Canvas

After the set-up, all resources are stored and can be used for the creation of DVMs. The same resource can also be used to construct several different DVMs depending on what columns the user selects when setting up. This feature provides a high level of flexibility, allowing users to personalize their DVMs to meet various analysis demands, as shown in Figure 21 with an example of a CSV file set-up. Same as before, the information required is minimal and can be provided by users of all technical levels.

Data Resource: CSV
Resource's Type: csv

Select the Root Node

Name of the Root Node:
 Column Position for the Root Node:

Add/Link other Nodes to the Root

Node Name:
 Column Position for the Node:

Node Name:
 Column Position for the Node:

Node Name:
 Column Position for the Node:

Node Name:
 Column Position for the Node:

Figure 21: Configuration of CSV



A major advantage of DataMingler over traditional relational database systems is its ability to allow users to manually define relationships between datasets by assigning the same node name to common attributes across different sources, rather than relying on predefined joins. For instance, Portfolio ID is one of the key features available in multiple datasets (portfolio table, holdings CSV, and transactions Excel file). The column name and position in each dataset can be different, therefore the user assigns the same node name manually during set-up of each resource. This ensures that common data points remain connected within the DVM architecture for easy integration. Having labeled the node common names, DataMingler then joins data points sharing similar values and creates unified data without going through advanced SQL joins. The process allows users complete freedom when it comes to linking datasets while reducing the extent of technical knowledge required to join data. This auto linking capability greatly reduces the time and technical expertise required to join data sources so that business users and analysts may allocate their time to data discovery and reporting, rather than struggling with database management.

Once a resource is utilized, its columns are automatically translated into entities in a graph schema, graphically represented on the right-hand pane of the module, as shown in Figure 22 for our use case. Graphical representation allows interactive navigation of data relationships eliminating the requirement for traditional tables. Through the graph, the users are able to better understand the dataset structure, revealing linkages that are perhaps not necessarily obvious in relational structures.

In addition, the DVM model offers schema flexibility to allow analysts to add or change data mappings as needed. Traditional systems require major reconfiguration every time a new data set is introduced, while DataMingler dynamically facilitates it. It is therefore a future-proof solution, able to adapt to evolving regulatory and analytical requirements. Users can also version dataset configurations so that data mappings from the past can be retrieved or modified as business and regulatory requirements change over time.



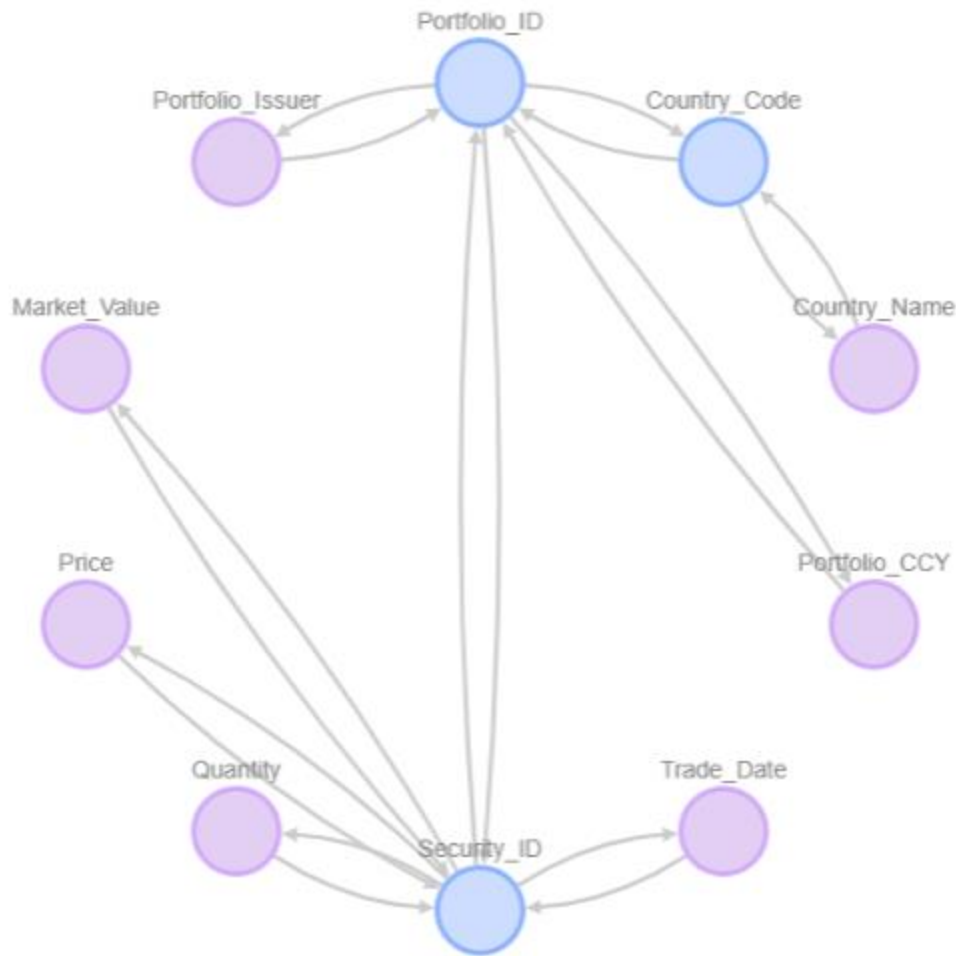
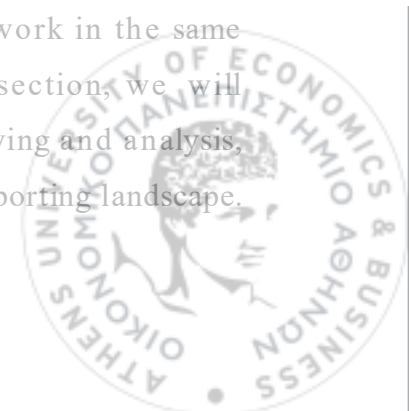


Figure 22: Use Case DVM

Through DataMingler, we were able to automate and simplify data integration for a real-life use case. The amount of manual effort otherwise required for converting disparate sources of data into a unified view was greatly reduced. Data consistency and accuracy are enhanced, compared to manual processing, while also avoiding the complexity involved with conventional ETL pipelines. Through its visual and interactive nature, the Data Canvas module allows collaboration between technical and non-technical users, with data engineers, analysts, and compliance groups being able to work in the same toolset without having special database knowledge. In the next section, we will demonstrate how this integrated data structure enables efficient querying and analysis, further showcasing the advantages of DataMingler in the regulatory reporting landscape.



6.3 Executing Queries in Data Mingler

Once the data are integrated successfully into DataMingler, the next step is utilizing the Query Builder module for further processing and analysis, as shown in Figure 23. Here, complex queries can be created and executed without programming skills required. The goal of this module is for people of varying degrees of technical skills to be able to generate insights and discover connections in datasets.

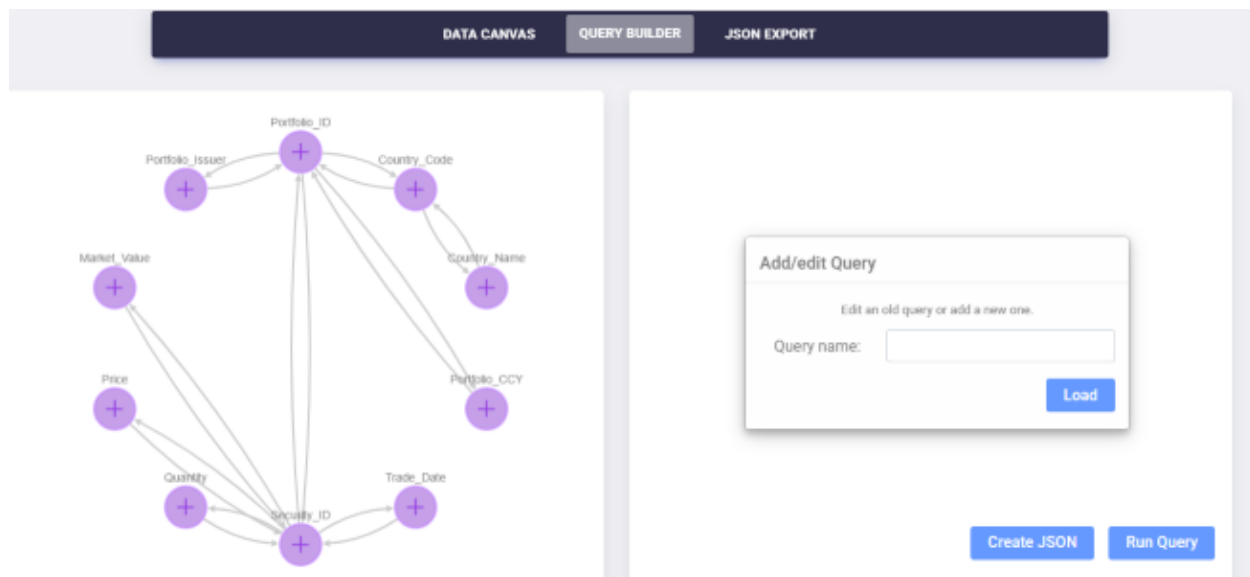


Figure 23: Use Case Query Builder

In the Query Builder module, with the utilization of the drag-and-drop feature, users can define queries by selecting entities and their attributes from the DVM graph. There is no need for construction of intricate SQL joins, as users can browse their data visually, choosing the relevant nodes. Subsequently, the users can apply conditions or transformations on each edge based on the algebraic framework, depending on their investigation. The system then translates these choices into a formal query so that users can obtain insights in an effective way. Multiple queries can also be based on the same DVM, by simply choosing a different root node, child nodes and transformations.



This graphical approach to query building renders data exploration and manipulation accessible to even non-technical users. Regulatory analysts and financial professionals can run complex queries on holdings, transactions, and portfolio structures without relying on database administrators. With operators such as aggregation, filtering, and mapping, the queries can be further personalized to achieve business objectives. The Query Builder module also supports users in saving query templates, which facilitates reusability as well as consistency in various analyses, even if the original data change.

The process for executing the same queries on a traditional relational database would require deep SQL knowledge, identification of the proper keys for different levels of joins and step-by-step debugging in order to obtain the proper results. Furthermore, the analysts would have to ensure that different data sources map properly, something which typically involves creating temporary tables, data cleaning, and reformatting datasets for compatibility. The trial-and-error method of query iteration is a process that substantially increases the potential for inefficiencies or mistakes. Additionally, revisions of the query also involve revisiting and rewriting huge parts of SQL code, and hence flexibility and responsiveness are difficult to attain in dynamic regulatory environments.

Below we will use three example queries to demonstrate how DataMingler enables users to execute queries.

Query 1: Portfolio-centric view

Our goal is to calculate the number of holdings each fund contains and the total market value of each fund. The analysis is centered around funds, therefore requiring Portfolio_ID to be the child node. In this way we ensure that all data aggregates around the individual portfolios. Once chosen, it is added to the right, and the left pane is affected accordingly, as shown in Figure 24. The user has now the ability to add any other node directly connected to the child node.



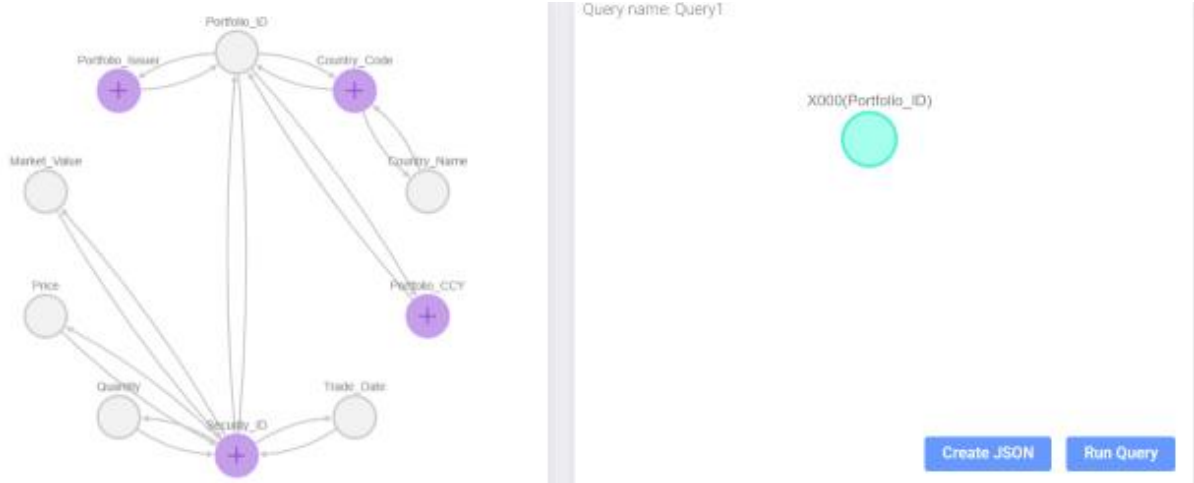


Figure 24: Use Case Query 1

- For the first goal, we need to add both Security_ID and Market_Value as child nodes to bring in holdings and valuation data, as shown in Figure 25. Portfolio_ID and Market_Value are not directly connected and therefore cannot be queried without Security_ID, we therefore need to include the path between them. We then apply a sum aggregation transformation on Market_Value.
- The second goal is simpler to achieve, as Portfolio_ID and Security_ID are directly connected. We add both nodes, as shown in Figure 25, and perform a count aggregation function on the Security_ID.

Query name: Query1

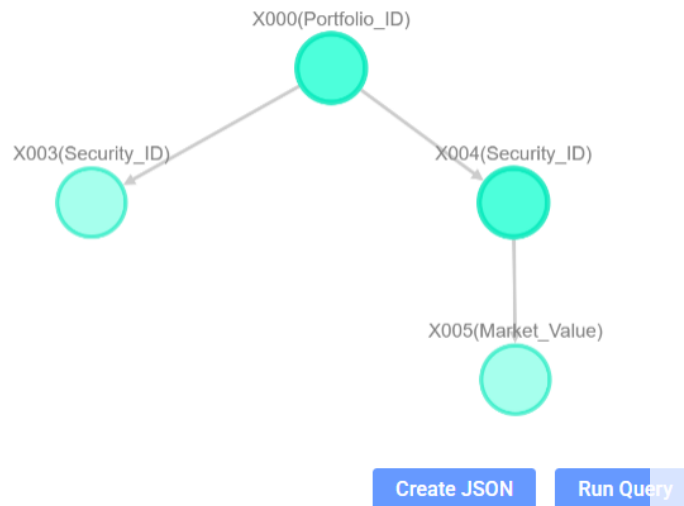


Figure 25: Use Case Query 1 Tree



This query enables asset managers to quickly assess the financial standing of their portfolios without requiring manual data consolidation. Additionally, the output can be dynamically linked to portfolio risk models, ensuring real-time insights into portfolio exposure and diversification.

Multiple transformations can also be applied to the same edge, by simply double clicking on it. For the purposes of the use case, we only used the aggregation functions ‘sum’ and ‘count’ for the first and second goals respectively, as shown in Figure 26.

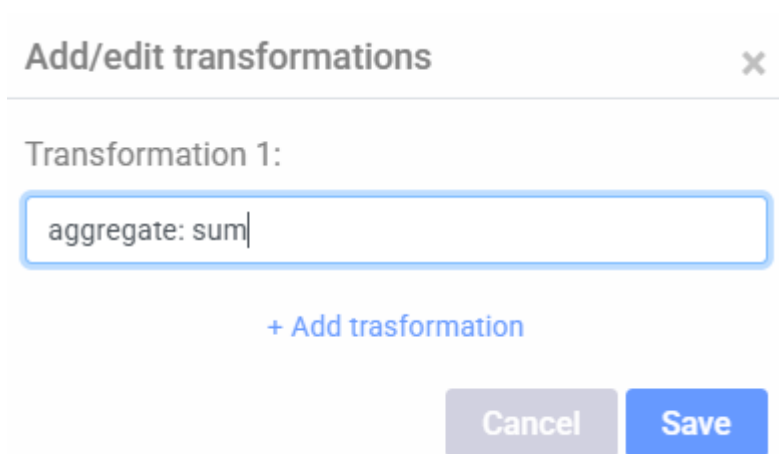


Figure 26: Use Case Query 1 Transformation

Query 2: Country-centric view

Our goal is to calculate the average investment value per country and the total number of funds. We therefore choose the Country Name as our root node, and we add as many additional levels required to reach the nodes we need. Meaning, the path from Country_Name to Market_Value for the first goal and from Country_Name to Portfolio_ID for the second, as shown in Figure 27. In turn here, we will use the average and the count aggregation functions.



Query name: Query2

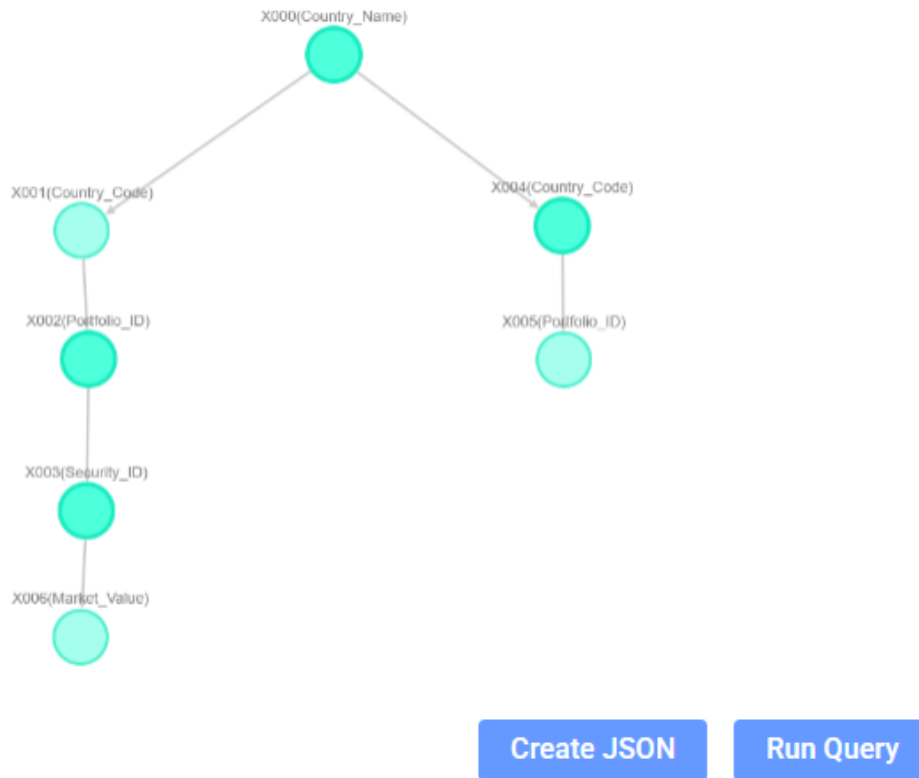


Figure 27: Use Case Query 2 Tree

This view provides insights into geographical investment distribution, helping analysts assess country-based exposure and risk. The ability to overlay these insights with macroeconomic indicators can enhance decision-making for strategic asset allocation.

Query 3: Transaction-centric view

Our goal is to find the most recent transaction per security and the number of funds that contain that security. Our root node here is the Security_ID. Again, we create two paths so that we achieve the desired outcome, as shown in Figure 28. The corresponding functions used are max and count.



Query name: Query3

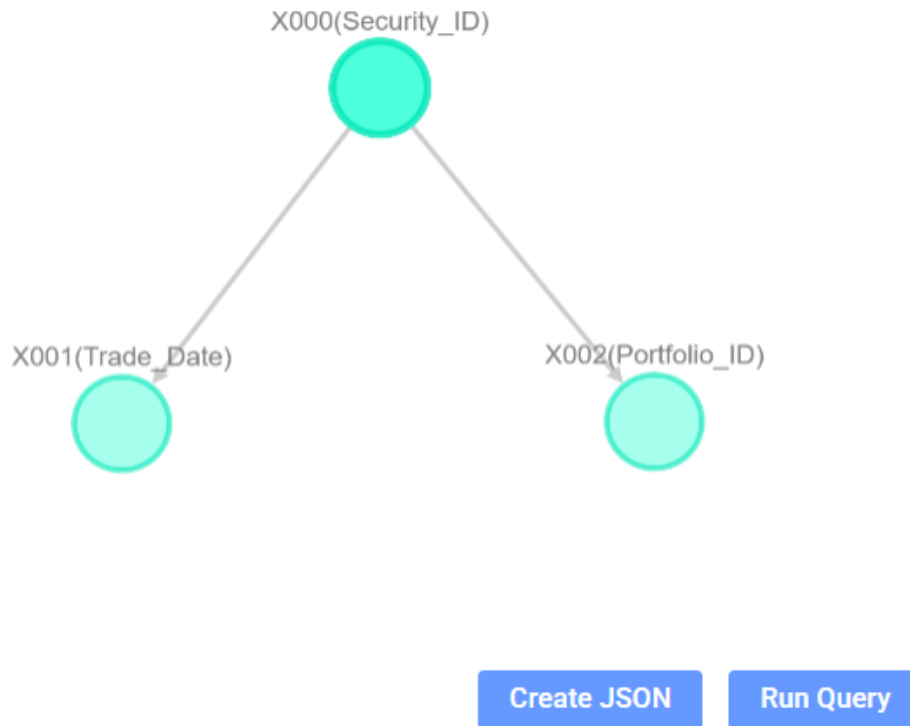


Figure 28: Use Case Query 3 Tree

This query helps asset managers track liquidity, monitor trading activity, and evaluate security distribution across portfolios. The results can be integrated into risk management frameworks, allowing firms to identify potential liquidity bottlenecks and compliance concerns.

DataMingler is based on a graph model, eliminating the need for writing queries by visualizing the connections between datasets. Users can navigate through data interactively, dynamically changing parameters without altering the underlying database schemas. DataMingler additionally allows the user to easily change selections and aggregations without resubmitting SQL code. This significantly reduces the reliance on database or IT professionals, as analysts themselves can derive insights without compromising accuracy and consistency across reporting tasks. These examples illustrate how flexible DataMingler's Query Builder module is, in the sense that users are able to



extract useful insights without knowing SQL while retaining full control of data relationships and transformations.

The process of building queries was fast and simple, without requiring extensive technical knowledge or tools. We managed to create three distinct views out of the same DVM. The first one focuses on the portfolios, through an estimation of their value and review of their structure. The second was a geographic analysis, allowing us to explore differences in investments per country and derive insights. The third is based on holdings and their transactions, to keep track of movements. This capability to swap between views with ease, without any reconfiguration required, allows us to explore the same datasets differently. Analysis can therefore be performed flexibly, according to the project's needs. The same analysis can also be performed with different data, by reusing the same DVM and queries.

6.4 Use Case results

Using DataMingler, as regulatory reporting analysts, we were able to consolidate heterogeneous, fragmented data into a single view, greatly reducing the efforts otherwise required for data integration and analysis. Asset managers have to operate within a heavily regulated environment where compliance reporting is mandatory. There are strict requirements on accuracy, transparency, and timely completion of reports. Traditional data processing techniques, including manual processing, rigorous ETL processes and SQL queries, are neither efficient nor adaptable in handling diverse financial data.

Through this application, we have illustrated how DataMingler facilitates the integration of disparate datasets using a dynamic and schema-flexible approach that allows users to rapidly connect, learn, and analyze data without programming knowledge. Compared with conventional data systems with rigid schemas, DataMingler's graph data model makes it possible to create and update connections among data points in rapid succession. It is therefore a suitable solution to accommodate the evolving regulatory requirements, diverse analysis needs, increasing data sources and data evolution.



In addition, the Query Builder module offers many facilitating features for data analysis. Analyst can visually construct queries, create transformations, and query information without resorting to SQL queries or lookup functions within spreadsheets. The ability to toggle between portfolio-oriented, country-oriented, and transaction-oriented views in a single DVM speaks volumes for the flexibility of this solution. Queries usually involve code development or manual reconciliations. With DataMingler we managed to build and execute them in a series of steps, significantly reducing query execution time and administrative effort.

Moreover, DataMingler facilitates collaboration and usability, bridging the gap among data engineers, analysts, and compliance personnel. IT experts are no longer a necessity for building queries and data mappings, allowing the fintech providers to save on costs, increase the level of efficiency in operations, and speed up decision-making. The resources and queries can also be reused in such a way that analysis can be performed consistently, while real-time updates allow companies to quickly react to ongoing changes.

DataMingler frees fintech professionals and regulatory specialists from the battle to cope with data complexity and they can instead divert their attention to derive value from it. With its ability to consolidate different sets of financial data, facilitate compliance reporting and exploratory data analysis, it is a critical tool for data management. The use of DVMs can make the fintech providers more responsive, precise, and likely to achieve regulatory compliance for their clients without losing their competitiveness in the dynamically changing financial environment.

6.5 Future Direction

Throughout section 6, we established the value of Data Virtual Machines (DVMs) and DataMingler in financial data management for fintech vendors. The natural next step would be an expansion to other business sectors. Companies handling large volumes of heterogeneous data would especially benefit from the flexibility, scalability, and efficiency that this approach has to provide.



A key sector for DataMingler expansion would be the banking sector. Banks deal with diverse data sets derived from multiple systems. The data ranges from information on customer transaction, details on loans or compliance reports. The typical approaches followed are either manual data extraction or utilization of ETL tools, which offer little flexibility while requiring extensive efforts and technical knowledge. With DataMingler, banks can consolidate fragmented data sets into one view, accessible by all of their employees, such as customer service or risk analysts, improving collaboration. For instance, processing a loan request requires customer information on credit score, income statements or interaction history, probably stored into multiple systems. The process can be performed faster and more accurately if all the required details can be obtained instantly.

Another sector that could benefit is telecommunications, where companies have to manage data on customer subscriptions, billings, and usage history information. Similar to banking, telecom operators work with disconnected databases, unable to view all customer's information in real time. With DataMingler, a telecom operator could automatically merge user information so that customer service personnel could observe a full account history, past issues, and payments, leading to faster resolution and higher customer satisfaction.

Expansion into these practical applications will make DVMs more widely adopted. By focusing on everyday data problems, DataMingler can become a building block tool in banking, telecommunications, and more sectors, making data more accessible and usable.



7. Conclusion

The growing demand for data portability in a highly heterogeneous digital environment entails significant technical as well as non-technical challenges. This thesis has considered the essential requirements, concerns, and solutions to enable data exchange across platforms in a seamless, secure, and efficient way. Through the analysis of technical requirements, such as usability, extensibility, scalability, linkability, and interoperability, and non-technical issues, such as regulatory compliance, governance, security, and privacy, the research has provided a comprehensive approach to address the requirements of contemporary data portability.

Another key contribution of the thesis is proposing and evaluating the Data Virtual Machine (DVM) as a new paradigm in data integration and portability. In contrast to traditional systems which are based on rigid data models, DVMs offer a schema-flexible, highly adaptive model. Users are able to connect, query, and manipulate data dynamically regardless of their format or source. The any-entity view, and model polymorphism capabilities further attest to the agility of DVMs to adapt with shifting data ecosystems, making them a credible substitute for traditional data integration and virtualization techniques.

In this thesis, we also demonstrated the application of the DVM with a practical scenario. We analyzed DataMingler, which is an implementation tool for DVMs. We showed how DataMingler can support financial institutions' data management requirements to merge heterogeneous data sets, ease compliance processes, and accelerate decision-making. DataMingler enables processing of complex data structures in an intuitive and effective way without the need for traditional ETL pipelines, manual processing, and rigid schemas. Besides data management for regulatory reporting, the likely applications of DVM solutions may involve banking, telecommunications, and others. DVMs can potentially be a practicable solution for addressing today's data challenges.

Last but not least, this thesis has established the DVMs potential as a disruptive data portability solution. Data Virtual Machines and DataMingler address significant technical and regulatory issues and provide an efficient, flexible, and scalable alternative to



traditional data management techniques. With further development and adoption, DVMS have the potential to revolutionize how data is stored, accessed, and exchanged by organizations and make the digital world more open, interoperable, and connected.



References

- [1] European Union, *General Data Protection Regulation (GDPR) – Article 20: Right to data portability*. Available: <https://gdpr-info.eu/art-20-gdpr/>.
- [2] J. Wong and T. Henderson, “How Portable is Portable? Exercising the GDPR's Right to Data Portability,” in *Proc. 2018 ACM Int. Joint Conf. and 2018 Int. Symp. on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp '18)*, New York, NY, USA: ACM, 2018, pp. 911–920. doi: [10.1145/3267305.3274152](https://doi.org/10.1145/3267305.3274152).
- [3] P. De Hert, V. Papakonstantinou, G. Malgieri, L. Beslay, and I. Sanchez, “The Right to Data Portability in the GDPR: Towards User-Centric Interoperability of Digital Services,” *Comput. Law & Secur. Rev.*, vol. 34, pp. 193–203, 2018. Available: <https://ssrn.com/abstract=3447060>.
- [4] OECD, *Mapping Data Portability Initiatives: Opportunities and Challenges*, 2021. Available: https://www.oecd.org/content/dam/oecd/en/publications/reports/2021/12/mapping-data-portability-initiatives-opportunities-and-challenges_97cd728b/a6edfab2-en.pdf.
- [5] S. Zunke and V. D'Souza, “JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats,” *Int. J. Comput. Sci. Netw. (IJCSN)*, vol. 3, no. 4, Aug. 2014. Available: <https://ijcsn.org/IJCSN-2014/3-4/JSON-vs-XML-A-Comparative-Performance-Analysis-of-Data-Exchange-Formats.pdf>.
- [6] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu, “Making database systems usable,” in *Proc. 2007 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD '07)*, New York, NY, USA: ACM, 2007, pp. 13–24. doi: [10.1145/1247480.1247483](https://doi.org/10.1145/1247480.1247483).
- [7] A. Hassan et al., “User experience: challenges and opportunities,” 2013. Available: https://seminar.utmspace.edu.my/jisri/download/F1_FinalPublished/Pub4_UserExperienceChallenges.pdf.
- [8] F. Adebessin, P. Kotzé, and H. Gelderblom, “The complementary role of two evaluation methods in the usability and accessibility evaluation of a non-standard



system,” in *Proc. 2010 Annual Research Conf. of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '10)*, New York, NY, USA: ACM, 2010, pp. 1–11. doi: [10.1145/1899503.1899504](https://doi.org/10.1145/1899503.1899504).

[9] J. Kranz, S. Kuebler-Wachendorff, E. Syrmoudis et al., “Data Portability,” *Bus. Inf. Syst. Eng.*, vol. 65, pp. 597–607, 2023. doi: [10.1007/s12599-023-00815-w](https://doi.org/10.1007/s12599-023-00815-w).

[10] M. Masmoudi, S. B. A. B. Lamine, M. H. Karray, B. Archimede, and H. B. Zghal, “Semantic Data Integration and Querying: A Survey and Challenges,” *ACM Comput. Surv.*, vol. 56, no. 8, Apr. 2024, Art. 209, pp. 1–35. doi: [10.1145/3653317](https://doi.org/10.1145/3653317).

[11] A. Achanta, “Data Democratization: Empowering Non-Technical Users with Self-Service BI Tools and Techniques to Access and Analyze Data Without Heavy Reliance on IT Teams,” *Int. J. Comput. Trends Technol. (IJCTT)*, vol. 71, no. 8, pp. 39-46, 2023. doi: [10.14445/22312803/IJCTT-V71I8P106](https://doi.org/10.14445/22312803/IJCTT-V71I8P106).

[12] J. Guia, V. Soares, and J. Bernardino, “Graph Databases: Neo4j Analysis,” in *Proc. 19th Int. Conf. on Enterprise Information Systems (ICEIS 2017) – Vol. 1*, 2017, pp. 351–356. Available: <https://www.scitepress.org/Papers/2017/63560/63560.pdf>.

[13] A. Sarikaya, M. Correll, L. Bartram, M. Tory, and D. Fisher, “What Do We Talk About When We Talk About Dashboards?” *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 682–692, Jan. 2019. doi: [10.1109/TVCG.2018.2864903](https://doi.org/10.1109/TVCG.2018.2864903).

[14] P. P.-S. Chen, “The Entity-Relationship Model: Toward a Unified View of Data,” *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, Mar. 1976. doi: [10.1145/320434.320440](https://doi.org/10.1145/320434.320440).

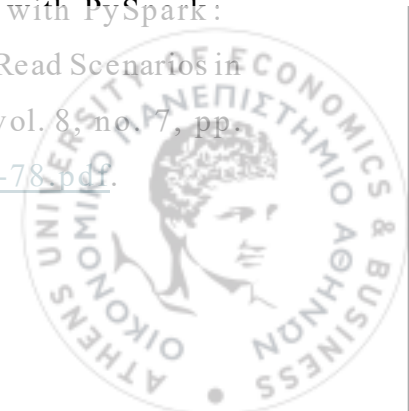
[15] Microsoft, *Power BI*. Available: <https://www.microsoft.com/en-us/power-platform/products/power-bi>.

[16] Tableau, *Tableau Software*. Available: <https://www.tableau.com/>.

[17] Qlik, *Qlik Data Analytics Platform*. Available: <https://www.qlik.com/us>.



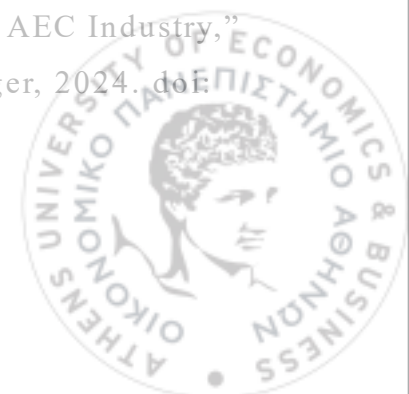
- [18] S. Decker et al., “The Semantic Web: The Roles of XML and RDF,” *IEEE Internet Comput.*, vol. 4, no. 5, pp. 63–73, Sept.–Oct. 2000. Available: <https://jmvidal.cse.sc.edu/library/w5063.pdf>.
- [19] A. C. Bock and U. Frank, “Low-Code Platform,” *Bus. Inf. Syst. Eng.*, vol. 63, pp. 733–740, 2021. doi: [10.1007/s12599-021-00726-8](https://doi.org/10.1007/s12599-021-00726-8).
- [20] OutSystems, *Low-code development: The ultimate low-code guide*. Available: <https://www.outsystems.com/low-code/>.
- [21] Dublin Core, *Metadata Basics*. Available: <https://www.dublincore.org/resources/metadata-basics/>.
- [22] R. V. Guha, D. Brickley, and S. Macbeth, “Schema.org: Evolution of Structured Data on the Web,” *Commun. ACM*, vol. 59, no. 2, pp. 44–51, Feb. 2016. doi: [10.1145/2844544](https://doi.org/10.1145/2844544).
- [23] S. Sagioglu and D. Sinanc, “Big data: A review,” in *Proc. 2013 Int. Conf. on Collaboration Technologies and Systems (CTS)*, San Diego, CA, USA, 2013, pp. 42–47. Available: https://academics.uccs.edu/~ooluwada/courses/datamining/ExtraReading/Big_data_A_review.pdf.
- [24] J. Fan, F. Han, and H. Liu, “Challenges of Big Data analysis,” *Natl. Sci. Rev.*, vol. 1, no. 2, pp. 293–314, Jun. 2014. doi: [10.1093/nsr/nwt032](https://doi.org/10.1093/nsr/nwt032).
- [25] C. K. Emani, N. Cullot, and C. Nicolle, “Understandable Big Data: A survey,” *Comput. Sci. Rev.*, vol. 17, pp. 70–81, 2015. doi: [10.1016/j.cosrev.2015.05.002](https://doi.org/10.1016/j.cosrev.2015.05.002).
- [26] I. M. Putrama and P. Martinek, “Heterogeneous data integration: Challenges and opportunities,” *Data Brief*, vol. 56, 2024. doi: [10.1016/j.dib.2024.110853](https://doi.org/10.1016/j.dib.2024.110853).
- [27] S. S. Kona, “Dynamic Schema Evolution and Data Ingestion with PySpark: Techniques for Handling Dynamic Schema Evolution and Schema-on-Read Scenarios in Data Ingestion Processes Using PySpark,” *Eur. J. Adv. Eng. Technol.*, vol. 8, no. 7, pp. 72–78, 2021. Available: <https://ejaet.com/PDF/8-7/EJAET-8-7-72-78.pdf>.



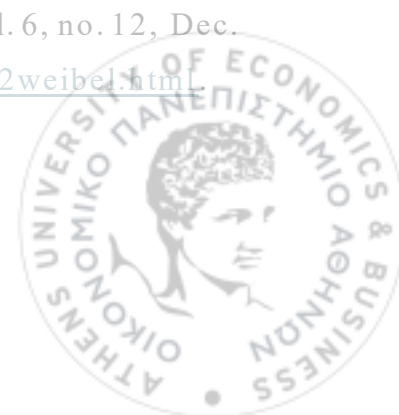
- [28] M. S. Thaiya et al., “On Software Modular Architecture: Concepts, Metrics and Trends,” *Int. J. Comput. Org. Trends*, vol. 12, no. 1, pp. 3–10, Jan.–Apr. 2022. doi: [10.14445/22492593/IJCOT-V12I1P302](https://doi.org/10.14445/22492593/IJCOT-V12I1P302).
- [29] H. Knoche and W. Hasselbring, “Continuous API Evolution in Heterogeneous Enterprise Software Systems,” in *Proc. 2021 IEEE 18th Int. Conf. on Software Architecture (ICSA)*, Stuttgart, Germany, 2021, pp. 58–68. Available: <https://www.researchgate.net/publication/351500899> Continuous API Evolution in Heterogeneous Enterprise Software Systems.
- [30] T. Kretschmer and J. Claussen, “Generational Transitions in Platform Markets—The Role of Backward Compatibility,” *Strategy Sci.*, vol. 1, no. 2, pp. 90–104, 2016. doi: [10.1287/stsc.2015.0009](https://doi.org/10.1287/stsc.2015.0009).
- [31] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, “End-User Development: An Emerging Paradigm,” in *End User Development*, vol. 9, H. Lieberman, F. Paternò, and V. Wulf, Eds. Dordrecht, Netherlands: Springer, 2006. Available: <https://www.researchgate.net/publication/225951884> End-User Development An Emerging Paradigm.
- [32] G. Bell et al., “Beyond the Data Deluge,” *Science*, vol. 323, pp. 1297–1298, 2009. Available: <https://www.researchgate.net/publication/24180847> Beyond the Data Deluge Computer Science.
- [33] F. Ullah, S. Dhingra, X. Xia, and M. A. Babar, “Evaluation of distributed data processing frameworks in hybrid clouds,” *J. Netw. Comput. Appl.*, vol. 224, 2024. doi: [10.1016/j.jnca.2024.103837](https://doi.org/10.1016/j.jnca.2024.103837).
- [34] S. Turner and L. M. Tanczer, “In principle vs in practice: User, expert and policymaker attitudes towards the right to data portability in the internet of things,” *Comput. Law Secur. Rev.*, vol. 52, 2024. doi: [10.1016/j.clsr.2023.105912](https://doi.org/10.1016/j.clsr.2023.105912).



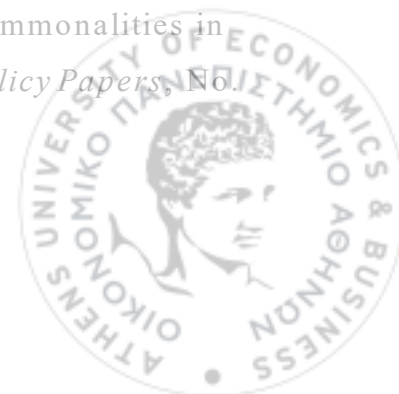
- [35] S. Ounacer, M. A. Talhaoui, S. Ardchir, A. Daif, and M. Azouazi, “A New Architecture for Real Time Data Stream Processing,” *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 8, no. 11, 2017. doi: [10.14569/IJACSA.2017.081106](https://doi.org/10.14569/IJACSA.2017.081106).
- [36] S. Ramanathan, G. Gautam, V. Srinivasan, and P. Parag, “Latency-Redundancy Tradeoff in Distributed Read-Write Systems,” in *Proc. 2022 14th Int. Conf. on COMMunication Systems & NETWORKS (COMSNETS)*, Bangalore, India, 2022, pp. 172–180. Available: https://www.researchgate.net/publication/354268563_Latency-Redundancy_Tradeoff_in_Distributed_Read-Write_Systems.
- [37] M. Rodriguez and R. Buyya, “Containers Orchestration with Cost-Efficient Autoscaling in Cloud Computing Environments,” 2018. Available: <https://arxiv.org/pdf/1812.00300>.
- [38] L. Somaini, “The Right to Data Portability and User Control: Ambitions and Limitations,” 2019. Available: <https://www.medialaws.eu/wp-content/uploads/2019/05/8.-Somaini.pdf>.
- [39] International Organization for Standardization (ISO), *ISO/IEC 2382-1:1993 – Information Technology – Vocabulary – Part 1: Fundamental Terms*, 3rd ed., 1993. Available: <https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:-1:ed-3:v1:en>.
- [40] A. V. Sambra et al., “Solid: A Platform for Decentralized Social Applications Based on Linked Data,” 2016. Available: http://emansour.com/research/lusail/solid_protocols.pdf.
- [41] HL7, *FHIR Overview*. Available: <https://www.hl7.org/fhir/overview.html>.
- [42] ISO 20022, *ISO 20022 Universal Financial Industry Message Scheme*. Available: <https://www.iso20022.org/>.
- [43] A. Lombardi, “Interoperability Challenges: Exploring Trends, Patterns, Practices and Possible Futures for Enhanced Collaboration and Efficiency in the AEC Industry,” in *Coding Architecture*, P. Ruttico, Ed. Cham, Switzerland: Springer, 2024. doi: [10.1007/978-3-031-47913-7_3](https://doi.org/10.1007/978-3-031-47913-7_3).



- [44] M. Cheatham and C. Pesquita, “Semantic Data Integration,” in *Handbook of Big Data Technologies*, A. Zomaya and S. Sakr, Eds. Cham, Switzerland: Springer, 2017. doi: [10.1007/978-3-319-49340-4_8](https://doi.org/10.1007/978-3-319-49340-4_8).
- [45] K. Davis, B. Peabody, and P. Leach, *RFC 9562: Universally Unique Identifiers (UUIDs)*, RFC Editor, USA, 2024. Available: <https://www.rfc-editor.org/rfc/rfc9562.pdf>.
- [46] A. Bonifati, P. Furniss, A. Green, R. Harmer, E. Oshurko, and H. Voigt, “Schema Validation and Evolution for Graph Databases,” in *Conceptual Modeling: 38th Int. Conf., ER 2019, Salvador, Brazil, Nov. 4–7, 2019, Proc.*, Berlin, Germany: Springer, 2019, pp. 448–456. doi: [10.1007/978-3-030-33223-5_37](https://doi.org/10.1007/978-3-030-33223-5_37).
- [47] L. Xu and D. W. Embley, “Combining the Best of Global-as-View and Local-as-View for Data Integration,” *Inf. Syst. Technol. Appl.*, 2004. Available: https://www.researchgate.net/publication/2844372_Combining_the_Best_of_Global-as-View_and_Local-as-View_for_Data_Integration.
- [48] C. Bizer, T. Heath, and T. Berners-Lee, “Linked Data - The Story So Far,” in *Linking the World’s Information: Essays on Tim Berners-Lee’s Invention of the World Wide Web*, 1st ed., New York, NY, USA: ACM, 2023, pp. 115–143. doi: [10.1145/3591366.3591378](https://doi.org/10.1145/3591366.3591378).
- [49] S. Sakr, M. Wylot, R. Mutharaju, D. Le Phuoc, and I. Fundulaki, “Distributed RDF Query Processing,” in *Linked Data*, Cham, Switzerland: Springer, 2018. doi: [10.1007/978-3-319-73515-3_4](https://doi.org/10.1007/978-3-319-73515-3_4).
- [50] J. Riley, *Understanding Metadata: What is Metadata, and What is it For?*, Baltimore, MD, 2017. Available: <https://digital.library.unt.edu/ark:/67531/metadc990983/>.
- [51] S. Weibel, “The Dublin Core Metadata Initiative” *D-Lib Mag.*, vol. 6, no. 12, Dec. 2000. Available: <https://mirror.dlib.org/dlib/december00/weibel/12weibel.html>.



- [52] G. Kellogg, P. Champin, and D. Longley, *JSON-LD 1.1 – A JSON-based Serialization for Linked Data*, 2019. Available: <https://hal.science/hal-02141614v1/file/mozilla.pdf>.
- [53] IBM, *What is data governance?*. Available: <https://www.ibm.com/think/topics/data-governance>.
- [54] D. Dittmann, “Compliance – Where does it fit in a data strategy?” *SAS Data Management Blog*, Feb. 27, 2017. Available: <https://blogs.sas.com/content/datamanagement/2017/02/27/compliance-fit-in-data-strategy/>.
- [55] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation – GDPR)*. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679>.
- [56] IBM, *What is data provenance?*. Available: <https://www.ibm.com/think/topics/data-provenance#:~:text=Data%20provenance%20is%20the%20record,Data%20provenance%20tools>.
- [57] V. Khatri and C. V. Brown, “Designing data governance,” *Commun. ACM*, vol. 53, no. 1, pp. 148–152, Jan. 2010. doi: [10.1145/1629175.1629210](https://doi.org/10.1145/1629175.1629210).
- [58] European Commission, *Data Governance Act (DGA)*. Available: <https://digital-strategy.ec.europa.eu/en/policies/data-governance-act>.
- [59] European Commission, *Digital Markets Act (DMA)*. Available: https://digital-markets-act.ec.europa.eu/index_en.
- [60] F. Casalini, J. López González, and T. Nemoto, “Mapping commonalities in regulatory approaches to cross-border data transfers,” *OECD Trade Policy Papers*, No. 248, OECD Publishing, Paris, 2021. doi: [10.1787/ca9f974e-en](https://doi.org/10.1787/ca9f974e-en).



[61] P. Buneman, S. Khanna, and W.-C. Tan, “Why and Where: A Characterization of Data Provenance,” in *Database Theory—ICDT 2001*, vol. 1973, J. Van den Bussche and V. Vianu, Eds. Berlin, Germany: Springer, 2001. doi: [10.1007/3-540-44503-X_20](https://doi.org/10.1007/3-540-44503-X_20).

[62] F. Lorè, P. Basile, A. Appice, et al., “An AI framework to support decisions on GDPR compliance,” *J. Intell. Inf. Syst.*, vol. 61, pp. 541–568, 2023. doi: [10.1007/s10844-023-00782-4](https://doi.org/10.1007/s10844-023-00782-4).

[63] T. R. Chhetri, A. Fensel, and R. J. DeLong, “GDPR consent management and automated compliance verification tool,” *SoftwareX*, vol. 27, 2024. Available: <https://www.sciencedirect.com/science/article/pii/S2352711024001924>.

[64] P. V. Kakarlapudi and Q. H. Mahmoud, “A Systematic Review of Blockchain for Consent Management,” *Healthcare*, vol. 9, no. 2, p. 137, 2021. doi: [10.3390/healthcare9020137](https://doi.org/10.3390/healthcare9020137).

[65] P. Silveira et al., “On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management,” in *Service-Oriented Computing. ICSSOC/ServiceWave 2009 Workshops*, vol. 6275, A. Dan, F. Gittler, and F. Toumani, Eds. Berlin, Germany: Springer, 2010. doi: [10.1007/978-3-642-16132-2_20](https://doi.org/10.1007/978-3-642-16132-2_20).

[66] J. Bartlett and N. Tkacz, *Governance by Dashboard*, Demos, Mar. 2017. Available: <https://demos.co.uk/wp-content/uploads/2017/04/Demos-Governance-by-Dashboard.pdf>.

[67] Financial Conduct Authority (FCA), *Final Notice: Equifax Limited*, 2023. Available: <https://www.fca.org.uk/publication/final-notice/equifax-limited-2023.pdf>.

[68] P. Jain, M. Gyanchandani, and N. Khare, “Big data privacy: a technological perspective and review,” *J. Big Data*, vol. 3, no. 25, 2016. doi: [10.1186/s40537-016-0059-y](https://doi.org/10.1186/s40537-016-0059-y).

[69] Google Cloud, *What is Encryption?*. Available: <https://cloud.google.com/learn/what-is-encryption>.

[70] Cloudean, *Data Encryption: The Ultimate Guide*. Available: <https://cloudian.com/guides/data-protection/data-encryption-the-ultimate-guide/>.



[71] IBM, *What is role-based access control (RBAC)?*. Available:

<https://www.ibm.com/think/topics/rbac>.

[72] Microsoft, *What is Multifactor Authentication?*. Available:

<https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661>.

[73] K. A. McKay and D. A. Cooper, *Special Publication 800-52 Revision 2: Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*, National Institute of Standards and Technology (NIST), 2019.

Available: <https://csrc.nist.gov/pubs/sp/800/52/r2/final>.

[74] IBM, *SFTP Server Overview*. Available: <https://www.ibm.com/docs/de/b2badv-communication/1.0.1?topic=concepts-sftp-server-overview>.

[75] Information Commissioner's Office (ICO), *Age Appropriate Design: A Code of Practice for Online Services – Data Minimisation*. Available: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/childrens-information/childrens-code-guidance-and-resources/age-appropriate-design-a-code-of-practice-for-online-services/8-data-minimisation/>.

[76] Information Commissioner's Office (ICO), *Anonymisation, Pseudonymisation and Privacy Enhancing Technologies Guidance – Chapter 3: Anonymisation*, 2022.

Available: <https://ico.org.uk/media/about-the-ico/consultations/4019579/chapter-3-anonymisation-guidance.pdf>.

[77] IEEE, *What is Differential Privacy?*. Available:

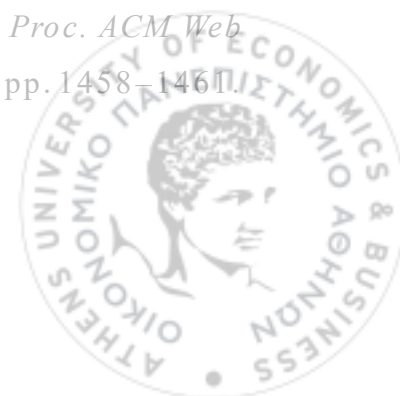
<https://digitalprivacy.ieee.org/publications/topics/what-is-differential-privacy>.

[78] P. Schaar, “Privacy by Design,” *IDIS*, vol. 3, pp. 267–274, 2010. doi:

[10.1007/s12394-010-0055-x](https://doi.org/10.1007/s12394-010-0055-x).

[79] V. Lähteenoja, “What are ‘personal data spaces’?” in *Companion Proc. ACM Web Conf. 2023 (WWW '23 Companion)*, New York, NY, USA: ACM, 2023, pp. 1458–1461.

doi: [10.1145/3543873.3587656](https://doi.org/10.1145/3543873.3587656).



- [80] IBM, *What is data sovereignty?*. Available:
<https://www.ibm.com/think/topics/data-sovereignty>.
- [81] H. Schüritz, S. Riasanow, and L. M. Kolbe, *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, Cham, Switzerland: Springer, 2022. doi:
[10.1007/978-3-030-93975-5](https://doi.org/10.1007/978-3-030-93975-5).
- [82] E. Curry, S. Scerri, and T. Tuikka, *Data Spaces: Design, Deployment, and Future Directions*, 2022. Available:
<https://library.oapen.org/viewer/web/viewer.html?file=/bitstream/handle/20.500.12657/58395/978-3-030-98636-0.pdf?sequence=1&isAllowed=y>.
- [83] E. Mansour et al., “A Demonstration of the Solid Platform for Social Web Applications,” in *Proc. 25th Int. Conf. Companion on World Wide Web (WWW '16 Companion)*, Geneva, Switzerland: Int. World Wide Web Conf. Steering Committee, 2016, pp. 223–226. doi: [10.1145/2872518.2890529](https://doi.org/10.1145/2872518.2890529).
- [84] World Wide Web Consortium (W3C), *OWL Web Ontology Language Overview*. Available: <https://www.w3.org/OWL/>.
- [85] World Wide Web Consortium (W3C), *Linked Data Platform (LDP) 1.0*. Available: <https://www.w3.org/TR/ldp/>.
- [86] Data Transfer Project (DTP), *Overview and Fundamentals*, 2018. Available: <https://dtinit.org/assets/dtp-overview.pdf>.
- [87] Data Transfer Project (DTP), *Data Transfer Project – GitHub Repository*. Available: <https://github.com/dtinit/data-transfer-project>.
- [88] Data Transfer Initiative, *Data Transfer Project Website*. Available: <https://dtinit.org/>.
- [89] Auth0, *OAuth 2.0 Authentication Protocol Overview*. Available: <https://auth0.com/docs/authenticate/protocols/oauth>.
- [90] J. Axelsson, “Mediators in Systems-of-Systems and Ecosystems: A Systematic Literature Review and Conceptualization,” in *Proc. 12th ACM/IEEE Int. Workshop on*



Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS '24), New York, NY, USA: ACM, 2024, pp. 21–28. doi: [10.1145/3643655.3643880](https://doi.org/10.1145/3643655.3643880).

[91] S. Busse, R.-D. Kutsche, U. Leser, and H. Weber, “Federated Information Systems: Concepts, Terminology and Architectures,” 1999. Available: https://www.researchgate.net/publication/2277599_Federated_Information_Systems_Concepts_Terminology_and_Architectures.

[92] A. Pan, J. Raposo, M. Álvarez, P. Montoto, V. Orjales, J. Hidalgo, L. Ardao, A. Molano, and Á. Viña, “The Denodo data integration platform,” in *Proc. 28th Int. Conf. on Very Large Data Bases (VLDB '02)*, 2002, pp. 986–989. Available: <https://dl.acm.org/doi/pdf/10.5555/1287369.1287456>.

[93] Denodo, *Data Management Overview*. Available: <https://www.denodo.com/en/data-management/data-management-overview>.

[94] J. Duggan et al., “The BigDAWG Polystore System,” *SIGMOD Rec.*, vol. 44, no. 2, pp. 11–16, Jun. 2015. doi: [10.1145/2814710.2814713](https://doi.org/10.1145/2814710.2814713).

[95] Open Banking, *Official Open Banking Website*. Available: <https://www.openbanking.org.uk/>.

[96] Software AG, *What is Open Banking?*. Available: https://www.softwareag.com/en_corporate/resources/api/article/open-banking.html.

[97] Office of the National Coordinator for Health Information Technology (ONC), *What is FHIR?*, 2019. Available: <https://www.healthit.gov/sites/default/files/2019-08/ONCFHIRFSWhatIsFHIR.pdf>.

[98] Google Cloud, *FHIR in the Cloud Healthcare API*. Available: <https://cloud.google.com/healthcare-api/docs/concepts/fhir>.

[99] D. Chatziantoniou and V. Kantere, “DataMingler: A Novel Approach to Data Virtualization,” in *Proc. 2021 Int. Conf. on Management of Data (SIGMOD '21)*, New York, NY, USA: ACM, 2021, pp. 2681–2685. doi: [10.1145/3448016.3452752](https://doi.org/10.1145/3448016.3452752).



- [100] D. Chatziantoniou and V. Kantere, “Data Virtual Machines: Enabling Data Virtualization,” in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB Workshops, Poly 2021 and DMAH 2021*, Berlin, Germany: Springer, 2021, pp. 3–13. doi: [10.1007/978-3-030-93663-1_1](https://doi.org/10.1007/978-3-030-93663-1_1).
- [101] D. Chatziantoniou and V. Kantere, “Data Virtual Machines: Data-Driven Conceptual Modeling of Big Data Infrastructures,” in *Workshops of EDBT 2020*, 2020. Available: <https://ceur-ws.org/Vol-2578/SEADData4.pdf>.
- [102] D. Chatziantoniou, V. Kantere, N. Antoniou, and A. Gantzia, “Data Virtual Machines: Simplifying Data Sharing, Exploration & Querying in Big Data Environments,” in *Proc. 2022 IEEE Int. Conf. on Big Data (Big Data)*, Osaka, Japan, 2022, pp. 373–380. doi: [10.1109/BigData55660.2022.10020508](https://doi.org/10.1109/BigData55660.2022.10020508).
- [103] M. I. Jordan, “Graphical Models,” *Statist. Sci.*, vol. 19, no. 1, pp. 140–155, Feb. 2004. doi: [10.1214/088342304000000026](https://doi.org/10.1214/088342304000000026).
- [104] A. Bogdanov et al., “Big Data Virtualization: Why and How?” in *Proc. CEUR-WS*, vol. 2679, 2020. Available: <https://ceur-ws.org/Vol-2679/paper2.pdf>.
- [105] J. D. Ullman, *Principles of Database and Knowledge-Base Systems, Vol. I*. Rockville, MD, USA: Computer Science Press, 1988. Available: <https://www.db.bme.hu/databases/principles-of-database-and-knowledge-base-systems-volume-1-1.pdf>.
- [106] C. Abras, D. Maloney-Krichmar, and J. Preece, “User-Centered Design,” 2004. Available: <https://aim.johnkeston.com/wp-content/uploads/2012/01/User-centered-design-encyclopedia-chapter.pdf>.

