Data Management at Scale, 2nd Edition

Piethein Strengholt

Published by O'Reilly Media, Inc.

Chapter 1. The Journey to Becoming Data-Driven

The pre-COVID-19 world was already fast and highly data-driven, but the pace of change has accelerated rapidly. Fierce competition, a digital-first era, ever-increasing customer expectations, and rising regulatory scrutiny require organizations to transform themselves into modern data-driven enterprises. This transformation will inevitably result in future organizations being more digital than those of today, and having a different view of data. Tomorrow's organizations will breathe data and embrace a philosophy that places it at the heart of their business. They will manage data as a product, make strategic decisions based on data analysis, and have a culture that acts on data.

Data-driven isn't just a buzzword.[1] Being data-driven provides an organization with a significant competitive advantage over other organizations. It can be proactive, and it can predict what will happen before it does. By using data correctly, organizations can quickly react to changes. Using data leads to greater confidence because decisions are based on facts, not intuition. With data, new industry trends and business opportunities can be spotted sooner. Customer retention and satisfaction are improved as well, because data tells organizations what customers think, how they behave, and what they want. With data, organizations can be more flexible, agile, and cost effective, because data provides insights into measured results, employee loyalty, dependencies, applications, and processes. So, the imperative for organizations to transform themselves into data-driven enterprises is definitively there.

Before we jump into the transformation itself, we'll explore the present-day challenges that require us to reevaluate how data must be managed. We'll establish a common definition of data management, encompassing all the disciplines and activities related to managing data as an asset. After that, we'll zoom in on several key technology developments and industry trends, and consider their impact on data management. We'll look at some best practices from the last decade of data management, providing insights into why previous-generation architectures are hard to scale. Finally, we'll consider what a next-generation data architecture might look like, and I'll present a set of action points that you will need to address while developing your data strategy.

Recent Technology Developments and Industry Trends

Transforming an organization to become data-driven isn't easy. It's a long-term process that requires patience and fortitude. With more data available, traditional architectures can no longer be scaled up because of their size, complexity, monolithic designs, and centralistic operating models. Enterprises need a new data strategy and cloud-based architecture. A

paradigm shift and change of culture are needed, too, because the centralized data and IT operating models that work today will no longer work when applying federated data ownership and self-serve consumption models. This requires organizations to redefine how people, processes, and technology are aligned with data.

Recent technology developments and industry trends force us to reevaluate how data must be managed. We need to shift away from funneling all data into a single silo toward an approach that enables domains, teams, and users to distribute, consume, and use data themselves easily and securely. Platforms, processes, and patterns should simplify the work for others. We need interfaces that are simple, well documented, fast, and easy to consume. We need an architecture that works at scale.

Although there are many positives about evolving into a truly data-driven organization, it's important to be aware of several technology developments and industry trends that are impacting data landscapes. In this chapter, I'll discuss each of these and its influence on data management. Firstly, analytics is fragmenting the data landscape because of use case diversity. Secondly, new software development methodologies are making data harder to manage. Thirdly, cloud computing and faster networks are fragmenting data landscapes. In addition, there are privacy, security, and regulatory concerns to be aware of, and the rapid growth of data and intensive data consumption are making operational systems suffer. Lastly, data monetization requires an ecosystem-to-ecosystem architecture. The impact these trends have on data management is tremendous, and they are forcing the whole industry to rethink how data management must be conducted in the future.

Fortunately, new approaches to data management have emerged over the past few years, including the ideas of a *data mesh* and *data fabric*:

- The data mesh is an exciting new methodology for managing data at large. The concept foresees an architecture in which data is highly distributed and a future in which scalability is achieved by federating responsibilities. It puts an emphasis on the human factor and addressing the challenges of managing the increasing complexity of data architectures.
- The data fabric is an approach that addresses today's data management and scalability challenges by adding intelligence and simplifying data access using self-service. In contrast to the data mesh, it focuses more on the technology layer. It's an architectural vision using unified metadata with an end-to-end integrated layer (fabric) for easily accessing, integrating, provisioning, and using data.

These emerging approaches to data management are complementary and often overlap. Despite popular belief among data practitioners and what commercial vendors say, they shouldn't be seen as rigid or standalone techniques. In fact, I expect these approaches to coexist and complement one another and any existing investments in operational data stores, data warehouses, and data lakes.

In the transition to becoming data-driven, organizations need to make trade-offs to balance the imperatives of *centralization* and *decentralization*. Some prefer a high degree of autonomy for their business teams, while others prioritize quality and control. Some organizations have a relatively simple structure, while others are brutally large and complex. Creating the perfect governance structure and data architecture isn't easy, so while developing your strategy, I encourage you to view these approaches to data management

as frameworks. There's no right or wrong. With the data mesh approach, for example, you might like some of the best practices and principles, but not others; you don't necessarily have to apply all of them.

In this book, I'll share my view on data management—one that is based on observations from the field while working closely with many large enterprises, and that helps you to make the right decisions by learning from others. We'll go beyond the concepts of data mesh and data fabric, because I strongly believe that a data strategy should be inclusive of both the operational and analytical planes, and that the decomposition method for both data domains and data products must be altered to fit the scale of large enterprises. To help you in your journey, I'll share my observations on the strategies and architectures different enterprises have designed, why, and what trade-offs they have made.

Before we jump into details, we need to agree on what data management is, and why it's important. Next, we need to determine how to define boundaries and shape our landscape based on various trade-offs. Finally, we'll examine how current enterprise data architectures can be designed and organized for today and tomorrow.

Let me lay my cards out on the table: decentralization is not a desired state, but the inevitable future of data. I therefore have a strong belief that scalability is forcing data management to become more decentrally organized. Managing data at scale requires you to federate key responsibilities, set strong standards, and properly align central and local resources and activities. This change affects multiple areas: people, processes, and technology. It forces you to decompose your architecture, dividing and grouping responsibilities. The shift from centralization to decentralization also contradicts the established best practice from the past decade: building large data silos in which all data is collected and integrated before being served. Although data warehouse and lake architectures are excellent approaches for utilizing data, these centralized models are not suited to a decentralized distributed data architecture.

Now that we've set the scene, I ask that you take a deep breath and put your biases aside. Many of us might be deeply invested in centralized data architectures; this has been a best practice for many years. I acknowledge that the need for data harmonization and for bringing large amounts of data into a particular context remains, and that doing so brings value to organizations, but something we must consider is the *scale* at which we want to apply this discipline. In a highly distributed ecosystem with hundreds or even thousands of applications, is the best way of managing data to apply centralization on all dimensions? Is it best to integrate and harmonize all data?

Data Management

The term *data management* refers to the set of processes and procedures used to manage data. The Data Management Association's Data Management Body of Knowledge (DAMA-DMBOK) has a more extensive explanation of data management, which it defines as "the development, execution, and supervision of plans, policies, programs, and practices that deliver, control, protect, and enhance the value of data and information assets throughout their life cycles."[2] The DAMA-DMBOK identifies 11 functional areas of data management, with data governance at the heart, as shown in Figure 1-1. It's crucial to embed all of these deeply into your organization. Otherwise, you'll lack insight and become ineffective, and your data will get out of control. Becoming data-driven—getting as much value as possible

out of your data—will become a challenge. Analytics, for example, is worth nothing if you have low-quality data.

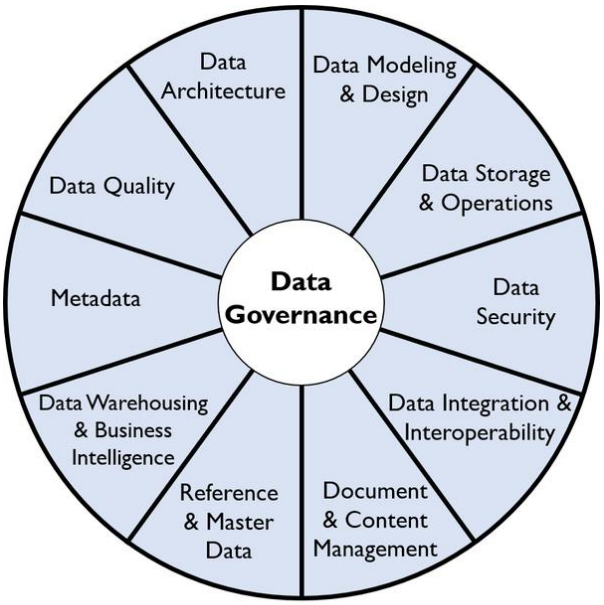

Figure 1-1. The 11 functional areas of data management

The activities and disciplines of data management are wide ranging and cover multiple areas, some closely related to software architecture.[3] In this book, I'll focus on the aspects of data management that are most relevant for managing a modern data architecture at scale. Let's take a closer look at the 11 areas identified in this figure and where they're covered in the book:

- *Data governance*, shown at the heart of Figure 1-1, involves all activities around implementing and enforcing authority and control over the management of data, including all corresponding assets. This area is described in detail in Chapter 8.
- *Data architecture* involves the definition of the master plan for your data, including the blueprints,[4] reference architectures, future state vision, and dependencies. Managing these helps organizations make decisions. The entire book revolves around data architecture generally, but the discipline and its activities will be covered fully in Chapters 2 and 3.
- *Data modeling and design* is about structuring and representing data within a specific context and specific systems. Discovering, designing, and analyzing data requirements are all part of this discipline. We'll discuss these topics in Chapters 4, 7, and 11.
- *Data storage and operations* refers to the management of the database design, correct implementation, and support in order to maximize the value of the data. Database management also includes database operations management. We'll address this in Chapter 11.
- *Data security* includes all disciplines and activities that provide secure authentication, authorization, and access to the data. These activities include

prevention, auditing, and escalation-mitigating actions. This area is described in more detail in Chapter 8.

- *Data integration and interoperability* includes all the disciplines and activities related to moving, collecting, consolidating, combining, and transforming data in order to move it efficiently from one context into another. *Data interoperability* refers to the capability to communicate, invoke functions, or transfer data among various applications in a way that requires little or no knowledge of the application characteristics. *Data integration*, on the other hand, is about consolidating data from different (multiple) sources into a unified view. This process, which I consider most important, is often supported by extra tools, such as replication and ETL (extract, transform, and load) tools. It's described extensively in Chapters 4, 5, and 6.

- *Document and content management* is the process of managing data stored in unstructured (media) and data formats. Some aspects of this will be discussed in Chapters 5 and 6.

- *Reference and master data management* is about managing critical data to make sure the data is accessible, accurate, secure, transparent, and trustworthy. This area is described in more detail in Chapter 10.**5**

- *Data warehousing* and *business intelligence* management includes all the activities that provide business insights and support decision making. This area, including advanced analytics, is described in more depth in Chapter 11.

- *Metadata* management involves managing all data that classifies and describes the data. Metadata can be used to make the data understandable, ready for integration, and secure. It can also be used to ensure the quality of data. This area is described in more detail in Chapter 9.

- *Data quality* management includes all activities related to managing the quality of data to ensure the data can be used. Some aspects of this area are described in Chapters 2 and 3.

The part of the DAMA-DMBOK that needs more work, which inspired me to write the first edition of this book, is the section on *data integration and interoperability*. I believe this section is lacking depth: the relationship to application integration and software architecture is not clear. It doesn't discuss decentralized architectures, and it lacks modern guidance on the interoperability of data, such as observability best practices and modern data pipeline management. In addition, the link to metadata management is weak. Metadata needs integration and interoperability, too, because it is scattered across many tools, applications, platforms, and environments in diverse shapes and forms. The interoperability of metadata—the ability of two or more systems or components to exchange descriptive data about data—gets insufficient treatment: building and managing a large-scale architecture is very much about metadata integration. Interoperability and metadata also aren't well connected to the area of data architecture. If metadata is utilized in the right way, you can see what data passes by, how it can be integrated, distributed, and secured, and how it connects to applications, business capabilities, and so on. There's limited guidance in the DAMA-DMBOK about managing your data as a whole by utilizing and connecting metadata.

Another concern I have is the view DAMA and many organizations have on achieving end-to-end semantic consistency. As of today, attempts to unify semantics to provide enterprise-wide consistency are still taking place. This is called a *single version of the truth*. However, *applications are always unique, and so is data*. Designing applications involves a lot of implicit thinking. The (domain) context of the business problem influences the design of the application and finds its way into the data. We pass through this context when we move from conceptual design into logical application design and physical application design.[6] It's essential to understand this because it frames any future architecture. When data is moved across applications, a data transformation step is always necessary. There's no escape from this data transformation dilemma! In the following chapters, I'll return to this idea.

Another view I see in many organizations is that data management should be central and must be connected to the strategic goals of the enterprise. Some organizations still believe that operational costs can be reduced by centralizing all data and management activities. There's also a deep assumption that a centralized platform can take away the pain of data integration for its users and consumers. Companies have invested heavily in their enterprise data platforms, which include data warehouses, data lakes, and service buses. The activities of master data management are strongly connected to these platforms because consolidating allows us to simultaneously improve the accuracy of our most critical data.

A centralized platform—and the centralized model that comes with it—will be subject to failure because it won't be able to keep up with the developments and trends that underpin decentralization, such as analytics, cloud computing, new software development methodologies, real-time decision making, and data monetization. While they may be aware of these trends, many companies fail to comprehend the impact they have on data management. Let's examine the most important trends and determine the magnitude of that impact.

Analytics Is Fragmenting the Data Landscape

The most trailblazing trend is *advanced analytics*, which exploits data to make companies more responsive, competitive, and innovative. Why does advanced analytics disrupt the existing data landscape? With more data available, the number of options and opportunities skyrockets. Advanced analytics is about making what-if analyses, projecting future trends and outcomes or events, detecting hidden relations and behaviors, and automating decision making. Because of the recognized value and strategic benefits of advanced analytics, many methodologies, frameworks, and tools have been developed to use it in divergent ways. We've only scratched the surface of what artificial intelligence (AI), machine learning (ML), and natural language processing (NLP) will be capable of in the future.

Note

OpenAI's new ChatGPT is a mind-blowing example of what AI is capable of. The models behind OpenAI, which include the Generative Pre-trained Transformer (GPT) series, can work on complex tasks such as analyzing math problems, writing code snippets, producing book essays, creating recipes using a list of ingredients, and much more.

These analytical trends force data to be distributed across many analytical applications because every individual use case requires different data. Unique business problems require unique thinking, unique data, and optimized technology to provide the best solution. Take, for example, a marketing business unit whose goal is to identify new sales opportunities for older and younger customers. Targeting these two audiences requires different features—measurable properties for analyzing—in datasets. For example, prospects that are younger will be segmented and clustered differently than prospects that are older. Asking the marketing department to use a single dataset for both target audiences requires many compromises, and you'll probably end up with a feature store that doesn't add any value to each use case.[7] The optimal solution for generating the most value is to give either use case its own unique set of features optimized for each learning algorithm. The increasing popularity of advanced analytics and resulting use case diversity issues lead to two problems: data proliferation and data intensiveness.

With data proliferation, data is distributed and scattered across a myriad of locations, applications, and databases. This is because consuming domains need to process the data to fit it into their unique solutions. This data distribution introduces other problems. For one, when data is repeatedly copied and scattered throughout the organization, it becomes more difficult to find its origin and judge its quality. This requires you to develop a single logical view of the same data that is managed in different locations. Additionally, extensive data distribution makes controlling the data much more difficult because data can be spread even further as soon as it leaves any given application. This requires you to develop a framework for efficiently reusing data, while applying governance and staying compliant with external regulations.

The proliferation of analytical techniques is also accelerating the growth of data intensiveness: the *read-versus-write ratio* is changing significantly. Analytical models that are constantly retrained, for example, constantly read large volumes of data. This impacts application and database designs because we need to optimize for data readability. It might also mean that we need to duplicate data to relieve systems from the pressure of constantly serving it, or to preprocess the data because of the large number of diverse use case variations and their associated read patterns. Additionally, we might need to provide different representations of the same data for many different consumers. Facilitating a high variety of read patterns while duplicating data and staying in control isn't easy. A solution for this problem will be provided in Chapter 4.

The Speed of Software Delivery Is Changing

In today's world, software-based services are at the core of most businesses, which means that new features and functionality must be delivered quickly. In response to the demands for greater agility, new ideologies have emerged at companies like Amazon, Netflix, Meta, Google, and Uber. These companies have advanced their software development practices based on two beliefs.

The first belief is that software development (Dev) and information technology operations (Ops) must be combined to shorten the systems development life cycle and provide continuous delivery with high software quality. This methodology, called *DevOps*, requires a new culture that embraces more autonomy, open communication, trust, transparency, and cross-discipline teamwork.

The second belief is about the size at which applications must be developed. Flexibility and speed of development are expected to increase when applications are transformed into smaller decomposed services. This development approach incorporates several buzzwords: *microservices*, *containers*, *Kubernetes*, *domain-driven design*, *serverless computing*, etc. I won't go into detail on all of these concepts yet, but it's important to recognize that this evolution in software development involves increased complexity and a greater demand to better control data.

The transformation of monolithic applications into distributed applications—for example, microservices—creates many difficulties for data management. When breaking up applications into smaller pieces, the data is spread across different smaller components. Development teams must also transition their (single) unique data stores, where they fully understand the data model and have all the data objects together, to a design where data objects are spread all over the place. This introduces several challenges, including increased network communication, data read replicas that need to be synchronized, difficulties when combining many datasets, data consistency problems, referential integrity issues, and so on.

The recent shift in software development trends requires an architecture that allows more fine-grained applications to distribute their data. It also requires a new *DataOps* culture and a different design philosophy with more emphasis on data interoperability, the capture of immutable events, and reproducible and loose coupling. We'll discuss this in more detail in Chapter 2.

The Cloud's Impact on Data Management Is Immeasurable

Networks are becoming faster, and bandwidth increases year after year. Large cloud vendors have proven that it's possible to move terabytes of data in the cloud in minutes, which allows for an interesting approach: instead of bringing the computational power to the data—which has been the common best practice because of network limitations—we can turn it around and bring the data to the computational power by distributing it. The network is no longer the bottleneck, so we can move data quickly between environments to allow applications to consume and use it. This model becomes especially interesting as software as a service (SaaS) and machine learning as a service (MLaaS) markets become more popular. Instead of doing all the complex stuff in-house, we can use networks to provide large quantities of data to other parties.

This distribution pattern of copying (duplicating) data and bringing it to the computational power in a different facility, such as a cloud data center, will fragment the data landscape even more, making a clear data management strategy more important than ever. It requires you to provide guidelines, because fragmentation of data can negatively impact performance due to data access lag. It also requires you to organize and model your data differently, because cloud service providers architected separate compute and storage to make them independently scalable.

Privacy and Security Concerns Are a Top Priority

Enterprises need to rethink data security in response to the proliferation of data sources and growing volume of data. Data is inarguably key for organizations looking to optimize, innovate, or differentiate themselves, but there is also a darker side with unfriendly undertones that include data thefts, discrimination, and political harm undermining

democratic values. Let's take a look at a few examples to get an idea of the impact of bad data privacy and security.

UpGuard maintains a long list of the biggest data breaches to date, many of which have capitalized on the same mistakes. The Cambridge Analytica breach and 500 million hacked accounts at Marriott are impressive examples of damaging events. Governments are increasingly getting involved in security and privacy because all aspects of our personal and professional lives are now connected to the internet. The COVID-19 pandemic, which forced so many of us to work and socialize from home, accelerated this trend. Enterprises cannot afford to ignore the threats of intellectual property infringements and data privacy scandals.

The trends of massive data, more powerful advanced analytics, and faster distribution of data have triggered a debate around the dangers of data, raising ethical questions and discussions. Let me share an example from my own country. In the Netherlands, the Dutch Tax Administration practiced unlawful and discriminatory activities. They tracked dual nationals in their systems and used racial and ethnic classifications to train models for entitlements to childcare benefits. The result: thousands of families were incorrectly classified as criminals and had their benefits stopped. They were ordered to repay what they already had received, sometimes because of technical transgressions such as failing to correctly sign a form. Some people were forced to sell their homes and possessions after they were denied access to debt restructuring.

This is just one example of improper use of data. As organizations inevitably make mistakes and cross ethical lines, I expect governments to sharpen regulation by demanding more security, control, and insight. We've only scratched the surface of true data privacy and ethical problems. Regulations, such as the new European laws on data governance and artificial intelligence will force large companies to be transparent about what data is collected and purchased, what data is combined, how data is used within analytical models, and what data is distributed (sold). Big companies need to start thinking about transparency and privacy-first approaches and how to deal with large regulatory issues now, if they haven't already.

Regulation is a complex subject. Imagine a situation in which several cloud regions and different SaaS services are used and data is scattered. Satisfying regulations, such as the GDPR, CCPA, BCBS 239, and the new Trans-Atlantic Data Privacy Framework is difficult because companies are required to have insight and control over all personal data, regardless of where it is stored. Data governance and correctly handling personal data is at the top of the agenda for many large companies.[8]

These stronger regulatory requirements and data ethics concerns will result in further restrictions, additional processes, and enhanced control. Insights about where data originated, how models are trained, and how data is distributed are crucial. Stronger internal governance is required, but this trend of increased control runs contrary to the methodologies for fast software development, which involve less documentation and fewer internal controls. It requires a different, more defensive viewpoint on how data management is handled, with more integrated processes and better tools. Several of these concerns will be addressed in Chapter 8.

Operational and Analytical Systems Need to Be Integrated

The need to react faster to business events introduces new challenges. Traditionally, there has been a great divide between transactional (operational) applications and analytical applications because transactional systems are generally not sufficient for delivering large amounts of data or constantly pushing out data. The accepted best practice has been to split the data strategy into two parts: operational transactional processing and analytical data warehousing and big data processing.

However, this divide is subject to disruption. *Operational analytics*, which focuses on predicting and improving the existing operational processes, is expected to work closely with both the transactional and analytical systems. The analytical results need to be integrated back into the operational system's core so that insights become relevant in the operational context. I could make the same argument for real-time events: when events carry state, the same events can be used for operational decision making and data distribution.

This trend requires a different integration architecture, one that better manages both the operational and analytical systems at the same time. It also requires data integration to work at different velocities, as these tend to be different for operational systems and analytical systems. In this book, we'll explore the options for preserving historical data in the original operational context while simultaneously making it available to both operational and analytical systems.

Organizations Operate in Collaborative Ecosystems

Many people think that all business activities take place within the single logical boundary in which the enterprise operates. The reality is different, because many organizations work closely with other organizations. Companies are increasingly integrating their core business capabilities with third-party services. This collaboration aspect influences the design of your architecture because you need to be able to quickly distribute data, incorporate open data,[9] make APIs publicly available, and so on.

These changes mean that data is more often distributed between environments, and thus is more decentralized. When data is shared with other companies, or using cloud or SaaS solutions, it ends up in different places, which makes integration and data management more difficult. In addition, network bandwidth, connectivity, and latency issues inevitably arise when moving data between platforms or environments. Pursuing a single public cloud or single platform strategy won't solve these challenges, because decentralization is the future. This means that if you want APIs and SaaS systems to work well and use the capabilities of the public cloud, you must be skilled at data integration, which this book will teach you how to do.

The trends discussed here are major and will affect the way people use data and the way companies should organize their architectures. Data growth is accelerating, computing power is increasing, and analytical techniques are advancing. Data consumption is increasing, too, which implies that data needs to be distributed quickly. Stronger data governance is required. Data management also must be decentralized due to trends like cloud services, SaaS, and microservices. All of these factors have to be balanced with a short time to market, thanks to strong competition. This risky combination challenges us to manage data in a completely different way.

Enterprises Are Saddled with Outdated Data Architectures

One of the biggest problems many enterprises are dealing with is getting value out of their current data architectures.[10] The majority of existing data architectures use a monolithic design—either an enterprise data warehouse (EDW)[11] or a data lake—and manage and distribute data centrally. In a highly distributed and consumer-driven environment, these architectures won't fulfill future needs. Let's look at some of the characteristics of each.

The Enterprise Data Warehouse: A Single Source of Truth

Most enterprises' first-generation data architectures are based on *data warehousing* and *business intelligence*. The philosophy is that centralization is the silver bullet to solve the problem of data management. With this approach there's one central integrated data repository, containing years' worth of detailed and stable data, for the entire organization. But when it comes to distributing data at large, such a monolithic design has several drawbacks.

Enterprise data unification is an incredibly complex undertaking and takes many years to complete. Chances are relatively high that the meaning of data differs across domains,[12] departments, and systems, even if data attributes have similar names. So, we either end up creating many variations or just accepting the differences and inconsistencies. The more data we add, and the more conflicts and inconsistencies that arise, the more difficult it will be to harmonize the data. We will likely end up with a unified context that is meaningless for the end users. In addition, for advanced analytics such as machine learning, leaving context out can be a significant problem because it's impossible to correctly predict the future if the data is meaningless or unrecognizable.

Enterprise data warehouses behave like *integration databases*, as illustrated in Figure 1-2. They act as data stores for multiple data-consuming applications. This means that they're a point of coupling between all the applications that want to access the data. It's often assumed that centralization is the solution to data management—this includes centralizing all data and management activities using one central team, building one data platform, using one ETL framework and one canonical model, etc. However, with a centralized architecture, changes need to be carried out carefully because of the many cross-dependencies between different applications. Some changes can also trigger a ripple effect of other changes. When you've created such a highly coupled and difficult-to-change design, you've created what some engineers call a "big ball of mud."

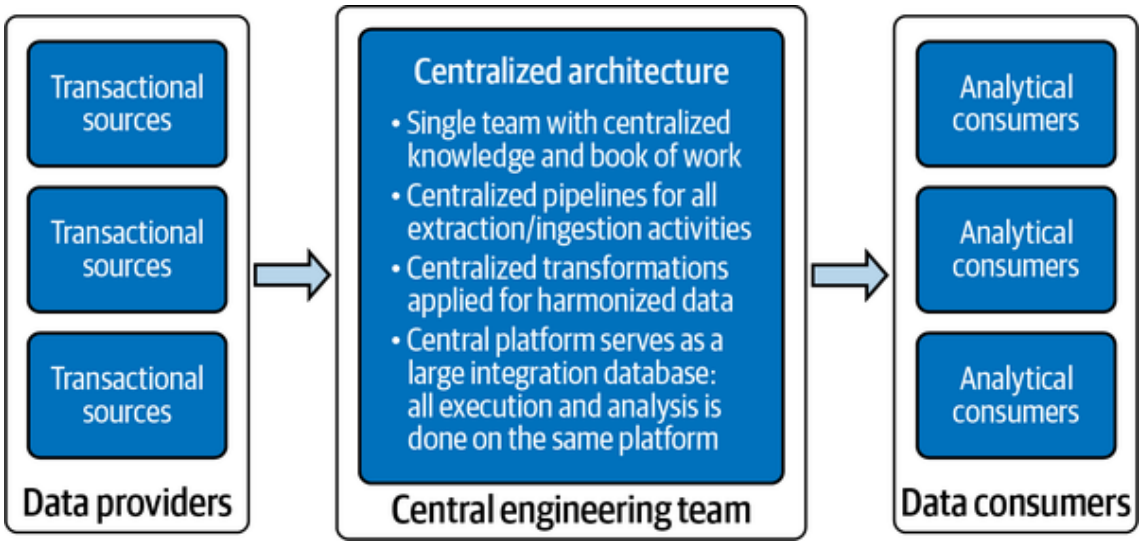

Figure 1-2. Centralized architectures are a bottleneck for many organizations: one team must wait for another team to finish its work

Big Ball of Mud

A big ball of mud is a haphazardly structured, sprawling, sloppy, duct-tape-and-baling-wire, spaghetti-code jungle. The term, introduced by Brian Foote and Joseph Yoder, describes a system architecture that is monolithic, difficult to understand, hard to maintain, and tightly coupled because of its many dependencies. Figure 1-3 shows a dependency diagram that illustrates this. Each line represents a relationship between two software components.

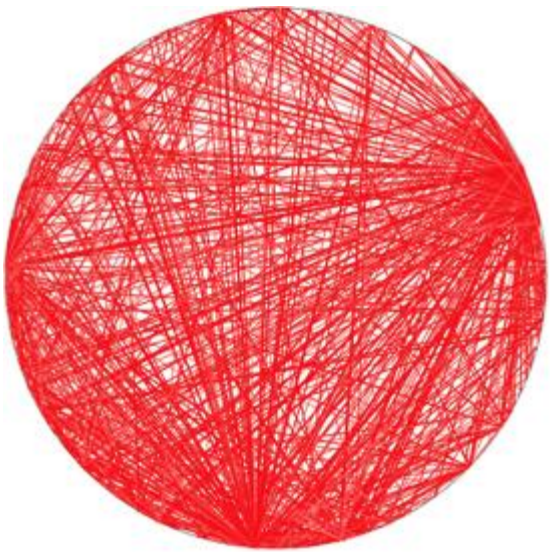

Figure 1-3. Big ball of mud dependency diagram

A big ball of mud, as you can see, has extreme dependencies between all components, which makes it practically impossible to modify one component without affecting others. Data warehouses, with their layers, views, countless tables, relationships, scripts, ETL jobs, and scheduling flows, often result in such a chaotic web of dependencies.

Because of the data warehouse's high degree of complexity and the fact that there's one central team managing it, the lack of agility often becomes a concern. The increased waiting time sparks the creativity of taking shortcuts. Engineers, for example, might bypass the integration layer and directly map data from the staging layer to their data marts, while other developers create workarounds using views that quickly combine data from different layers. The *technical debt* (future rework) that accumulates as a result will cause problems later. The architecture will become more complex, and people will lose insight into the creative solutions crafted to ensure timely delivery.

Furthermore, data warehouses are tightly coupled with the underlying chosen solution or technology, meaning that consumers requiring different read patterns must export data to other environments. As the vendor landscape changes and new types of databases pop up, warehouses are increasingly required to distribute data away from the location where it is managed. This trend undermines the grand vision of efficiently using a single central repository and utilizing the underlying (expensive) hardware.

Life cycle management of historical data is often an issue too. Data warehouses are seen as archives of truth, allowing operational systems to clean up irrelevant data, knowing the

data will be retained in the warehouse. For advanced operational analytics—something that emerged after data warehouses made an appearance—this might be a problem. If data has been transformed by a central team that manages all of the organization's data, it may no longer be recognizable to the team that delivered it (or usable for the intended operational use case). Additionally, making the data available quickly may be difficult, given that many warehouses typically process data for many hours before storing it. This is necessary because before being stored, the data must be transformed, combined, and integrated with other data.

Data quality is often a problem as well. Who owns the data in a data warehouse? Who is responsible if source systems deliver corrupted data? What I often see at organizations is that a central engineering team takes care of data quality issues caused by other teams. In one instance, the engineers fixed the data in the staging layer so that it would load properly into the data warehouse. These fixes became permanent, and over time hundreds of additional scripts had to be applied before data processing could start. These scripts aren't part of trustworthy ETL processes and don't allow for data ownership and lineage that can be tracked back.

Regulatory issues can crop up too. Warehouses often lack insight into ad hoc consumption and further distribution, especially when data is carried outside of the boundaries in which it is managed. With new regulations, data ownership and insight into the consumption and distribution of data is important because you need to be able to explain what personal data has been consumed by whom and for what purpose.

Given the total amount of data in a typical data warehouse, the years it took to develop it, the knowledge people have, and intensive business usage, a replacement migration would be a risky and time-consuming activity. Therefore, many enterprises continue to use this architecture and feed their business reports, dashboards,[13] and data-hungry applications from a data warehouse, despite the high maintenance costs and lack of flexibility.

The Data Lake: A Centralized Repository for Structured and Unstructured Data

As data volumes and the need for faster insights grew, engineers started to work on other concepts. *Data lakes* emerged as an alternative for access to raw and larger volumes of data.[14] Providing data as is, without having to structure it first, enables any consumer to decide how to use it and how to transform and integrate it.

Data lakes, like data warehouses, are centralized (monolithic) data repositories, but they differ from warehouses in that they store data before it has been transformed, cleansed, and structured. Schemas therefore are often determined when reading data. This differs from data warehouses, which use a predefined and fixed structure. Data lakes also provide greater variety by supporting multiple data formats: structured, semistructured, and unstructured.

Many data lake implementations collect pure, unmodified, raw data from the original source systems. Dumping in raw application structures—exact copies—is fast and allows data analysts and scientists quick access. However, the complexity with raw data is that use cases always require reworking the data. Data quality problems have to be sorted out, transformations are required, and enrichments with other data are needed to bring the data into context. This introduces a lot of repeatable work and is one reason why data lakes are typically combined with data warehouses. Data warehouses, in this combination, act like

high-quality repositories of cleansed and harmonized data, while data lakes act like (ad hoc) analytical environments, holding a large variety of raw data to facilitate analytics.

Designing data lakes, just like data warehouses, is a challenge. Companies like Gartner, VentureBeat AI, and Deloitte see high failure rates for big data projects—often more than 60%.[15] Data lake implementations typically fail, in part, because of their immense complexity, difficult maintenance, and shared dependencies:

- Data that applications push into a data lake is often raw and likely a complex representation of how the individual applications internally organize their data. When you follow this approach, your data lake quickly becomes a collection that includes tens of thousands of tables, incomprehensible data structures, and technical values that are understood only by the applications themselves. Additionally, there's tight coupling with the providing applications, since the inherited structure is an identical copy. When consumers use this data directly, there's a real risk that data pipelines will break when those providing applications change their data structures.
- Analytical models in data lakes are often trained on both raw and harmonized data. It isn't unthinkable that both data engineers and data scientists are technically data plumbing, creating data and operating these data pipelines and models by hand or in their data science projects. Data lakes therefore carry substantial (operational) risks.
- Data lakes are often a single platform shared by many different use cases. Due to their tight coupling, compatibility challenges, shared libraries, and configurations, these platforms are hard to maintain.

These are just a few reasons why the failure rate of big data projects is so high. Other reasons include management resistance, internal politics, lack of expertise, and security and governance challenges.

The Pain of Centralization

Enterprise data warehouses or central data lakes can be scaled up using techniques like metadata-driven ELT, data virtualization, cloud services, distributed processing, real-time ingestion, machine learning for enrichments, and so on. But there's a far bigger problem: the centralized thinking behind these architectures. This includes centralized management, centralized data ownership, centrally clustered resources, and central data models that dictate that everybody must use the same terms and definitions. One consequence of this centralization is that removing data professionals from business domains limits creativity and business insights. Teams are forced into constant cross-communication and ticket management, because they are at a distance. *It's the central operating model that is the bottleneck*: the central team can't handle all questions and requests quickly enough. It's for good reason that organizations are embracing decentralized methodologies like the data mesh and advocating domain-driven design. A federated model with domain teams can be a superpower: it promotes independence and accountability; it enables scaling because autonomous teams govern, share and consume data in parallel; and it fosters improvements in quality, collaboration, and productivity.

However, there's a caveat to decentralization. Federation of responsibilities always starts with centralization, which involves on an enterprise level defining standards, setting boundaries, and providing expertise and scalability. Without a central authority, decentralized empowerment becomes a huge problem. I've observed situations where teams started to define their own technology, interoperability, and metadata standards; where departments mandated complete control over their own data, so it couldn't be combined with data from other teams; and where teams applied their own data-modeling standards, reference-data standards, granularity levels, and so on, making it impossible to combine or integrate data between domains. To summarize: scalability through decentralization doesn't come without risks, and those risks can be best mitigated by central alignment of organization, governance, technology, and architecture. So, what you need is a *data strategy*: a plan that (re)defines culture, organization and people, architecture, governance and processes, and rules required to turn yourself into a data-driven organization.

Defining a Data Strategy

Now that you understand the importance of data, recent trends, and the challenges that come with centralization and decentralization, what might a strategic path toward a successful data transformation look like? To answer this question, a strategy exercise must be carried out, taking into account the company's business models and the role of data within them. Strategists and architects should consider whether changes to the existing business models are warranted, and whether conditions are right for, for example, applying federation.

It is convention to begin by analyzing the organization and the conditions in which the company operates. From there, an enterprise view must be developed that connects the building blocks of the enterprise to applications and data sources, processes, people, and other technology solutions that are used to realize the strategy. This view enables visualization of the (data) architecture that will allow the company's strategic objectives to be met. In the next chapter, I'll connect back to all of these aspects, but first let's look at the steps for defining a data strategy.

Developing a strategy can be difficult. It often involves a prediction of what business value can be created or what may be true in the future. It requires connecting high-level ambitions to resources, processes, and supporting capabilities. If you don't already have a data strategy or are unsure about your ambitions, please consider the following action points:

- First, focus on your business goals and strategy. Don't get hooked up in the hype about the latest technology changes.
- Embrace the company's vision and adapt it to how data will boost it. Determine how data can play a better role in solving your business problems. Try to quantify the business impact. Next, define what your core business strategy will look like for the next three years and after.
- Determine the correct balance between "defensive" and "offensive." Is full control a top priority? Or do you want to have the flexibility for disruptive innovation? How does regulation impact your strategy? These considerations will influence your initial design and the pace of federating certain responsibilities.

- Establish clear and measurable milestones. Create a plan to determine when your data strategy is successful and provides value. Define metrics and key performance indicators (KPIs) for measuring outcomes.
- Create a communication plan to ensure your strategy is understood and communicated throughout the organization.
- Determine your first focus area. This might be research and development, regulatory compliance, cost reduction, increased revenues, or something else.
- Identify barriers within your organization that might prevent a successful outcome, and look for ways around them.
- Rethink your strategy for talent and determine whether your organization is fit to fully take advantage of data. If not, create a (cultural) transition and hiring plan for aligning your business departments by making them responsible for managing and using data at large.
- Define what your next-gen IT capabilities will look like. Is future IT responsible for all execution? Or does future IT acts as a center of excellence for providing leadership, best practices, training, research, and support?
- Define what a data-driven culture looks like. For example, identify what skills and education are needed. Do you plan to allow full self-service data usage by business users? Is there willingness to share data with others?
- Gather data about your existing environments. For example, does every organizational business unit participate within the same ecosystem boundaries? Or do you see different rules applied within different environments? Start by creating a high-level overview of your data landscape by analyzing the different environments and data sources. Study where data is created, and how that data is distributed and used.
- Determine whether you will need to transform your software development methodology. If so, determine how this might impact your delivery model for building your next-gen architecture.
- Establish what a transition toward the future will look like. Is your transition a radical remodel or an organic evolution?
- Define some first high-level technology choices. For example, do you plan to take advantage of scalable cloud computing for processing data? Do you prefer flexibility and adaptability, so you will be developing your IT capabilities to be vendor-agnostic? Or do you prefer consuming native services, with the risk of vendor lock-in?
- Create an executive presentation with the most important elements, all high-level with sections: why, what, gaps, road maps, and investment cases.
- Create a cost management strategy by aligning your budget with the goals and use cases. Determine a plan for reviewing and willingness to invest in data management. In addition to that, align the budgets with your IT life cycle

management process, which entails planning, outsourcing, procurement, deployment, and potential decommissioning.[16]

Addressing these action points should make your high-level ambitions and strategy clear. It's important to do this first, because the strategic direction will provide input for decisions about architecture, processes, ways of working, transition road maps, governance, and so on.

Note that a data-driven strategy will look different for every organization, depending on its ambitions, the size and type of the business, planned funding and investments, the existing culture and level of maturity, and the underlying business objectives. For example, in my previous role at ABN AMRO, we wanted customers to have a bespoke, seamless, and unified experience throughout their digital journey when interacting with our data-driven bank. Therefore, we decided to strongly align and integrate all retail, corporate, and international business units. This meant that all business units had to work within the same data-centric ecosystem, follow the same data literacy program, and adhere to the same architectural principles. To facilitate this, we also had to integrate the different enterprise architecture functions for designing, guiding, and controlling the enterprise as a whole.

In another organization, the data strategy might look completely different. For example, if the business objectives are less closely aligned throughout the organization, there might be fewer needs for standardization and integration. If you've recently experienced the traumatic event of data leakage, then you will probably be strongly focused on data defense. If your primary objective is to monetize data and build partnerships, your focus will be on delivering interoperability for economic value and data that is valuable in the sense that it's worth consuming for the purchasing party. The most important point is to align your data strategy with your current and future business needs.

Before we wrap up, I want to acknowledge that some people advocate that an overarching strategy isn't required. For them, it's too much up-front work. Instead of developing a plan, they believe you should just get started and see what happens along the way while implementing use cases. This approach will absolutely deliver immediate value. However, in the long term it will have disastrous consequences for your architecture because there will be no commitment to consistency and standardization. Without a guiding map, the transition could be a long march in the wrong direction, requiring frequent reboots and backpedaling exercises.

Wrapping Up

A strategy is the starting point of every digital transformation. Data undoubtedly is a core ingredient. But how do you ensure that data actually contributes to your long-term goals and ambitions? And how do you facilitate the organizational change that is needed for using data at large? This is where a helicopter view is required, to enable a real understanding of the business from end to end. Planning for success requires you to first get a complete picture. You need to zoom out and get an abstract view of your business, before zooming in to specific problem spaces. After that, you need to define an architecture and organizational structure that will support your transition and align with your overall planning, way of working, and governance model. As we progress further into the book, you'll learn more about these concepts.

Designing an architecture that matches all of your data strategy requirements and ambitions is the hardest part of the transition. First, you must create a holistic picture of your organizational structure, applications, processes, and data landscape. Next, you can work out different potential scenarios by balancing dimensions such as flexibility, openness, control, time, cost of realization, risks, and so on. Each potential scenario might include road maps, transition architectures, and different future states. This process is a dynamic and collaborative exercise and goes far beyond creating simple visualizations. An important note on this: setting out the strategic and future direction starts top-down, from the business. It shouldn't come from only the IT department.

An important aspect of your data strategy is balancing the imperatives of centralization and decentralization. For example, do you federate governance and activities, or architecture and infrastructure, or both? There's a lot of confusion on this subject, as we often see extremes. For example, the data mesh, as a reaction to centralized data organizations, architectures, and governance models, promotes a 180-degree change of direction. It embraces decentralization in all aspects by advocating that all data must be decentrally owned by domains using decentrally managed infrastructure. On paper, decentralization resonates well with business leaders and IT managers, but in practice, things aren't that easy.

Decentralization involves risks, because the more you spread out activities across the organization, the harder it gets to harmonize strategy and align and orchestrate planning, let alone foster the culture and recruit the talent needed to properly manage your data. The resulting fragmentation also raises new questions, like "How many business models or domains do I have?" and "How many data platforms do I need?" What I have observed in practice is that many architects and data practitioners find domain decomposition and setting boundaries the hardest part of decentralization. Often, there's a fundamental lack of understanding of business architecture and domain-driven design. Many data practitioners find the conceptual notions of domain-driven design too difficult to grasp. Others try to project oversimplified or object-oriented programming examples onto their data landscape.

The principles underpinning decentralization are also subject to discussion with regard to how data and platforms must be managed. Among data engineers, there's skepticism because no open interoperability standards for data products exist yet. In addition, there are concerns about the usability of data when individual teams operate in bubbles.

These observations are a nice transition to the following chapters. In the next chapter, we'll fly our helicopter one level lower, taking you close enough to identify your architecture, data domains, and landing zones. I'll help you with this by simplifying the vocabulary of domains and providing you with pragmatic guidance and observations from the field. After that, we'll fly over to the source system side of our landscape and stay there for a few chapters before our journey continues. Bon voyage!

1 The data analytics market is surging. It reached $200+ billion in annual spending in 2022, according to IDC.

2 The Body of Knowledge is developed by the DAMA community. It has a slow update cycle: the first release was in 2008, the second one in 2017.

3 *Software architecture* refers to the design and high-level structure of software needed to develop a system that meets business requirements and includes characteristics such as flexibility, scalability, feasibility, reusability, and security. If you want to learn more, read *Fundamentals of Software Architecture* by Mark Richards and Neal Ford (O'Reilly).

4 A blueprint is also known as a *reference implementation*. It consists of the Infrastructure as Code (IaC) definitions to successfully create a set of services for a specific use case.

5 We use the term "master data management" (MDM) in this book because it has been widely adopted and has no industry-wide alternative at the time of this writing, but we welcome more inclusive alternatives to "master/slave" terminology.

6 If you want to learn more about the different phases of data modeling, see my blog post "Data Integration and Data Modelling Demystified". The conceptual model is sometimes also called the *business model*, the *domain model*, *domain object model*, or *analysis object model*.

7 A *feature store* is a tool for storing commonly used features (individual measurable properties or characteristics of a phenomenon).

8 *Personal data* is any information related to an identified or identifiable natural person.

9 *Open data* is data that can be used freely and is made publicly available.

10 An enterprise's data architecture comprises the infrastructure, the data, and the schemas, integrations, transformations, storage, and workflows required to enable the analytical requirements of the information architecture.

11 For background information on (enterprise) data warehouses and why they don't scale, see my blog post "The Extinction of Enterprise Data Warehousing".

12 A *domain* is a field of study that defines a set of common requirements, terminology, and functionality for any software program constructed to solve a problem in the area of computer programming.

13 Reports tend to be mainly tabular, but they may contain additional charts or chart components. Dashboards are more visual and use a variety of chart types.

14 James Dixon, then chief technology officer at Pentaho, coined the term *data lake*.

15 Brian T. O'Neill maintains a reference list on failure rates.

16 Technologies are volatile, so at some point you may need to replace or upgrade them.