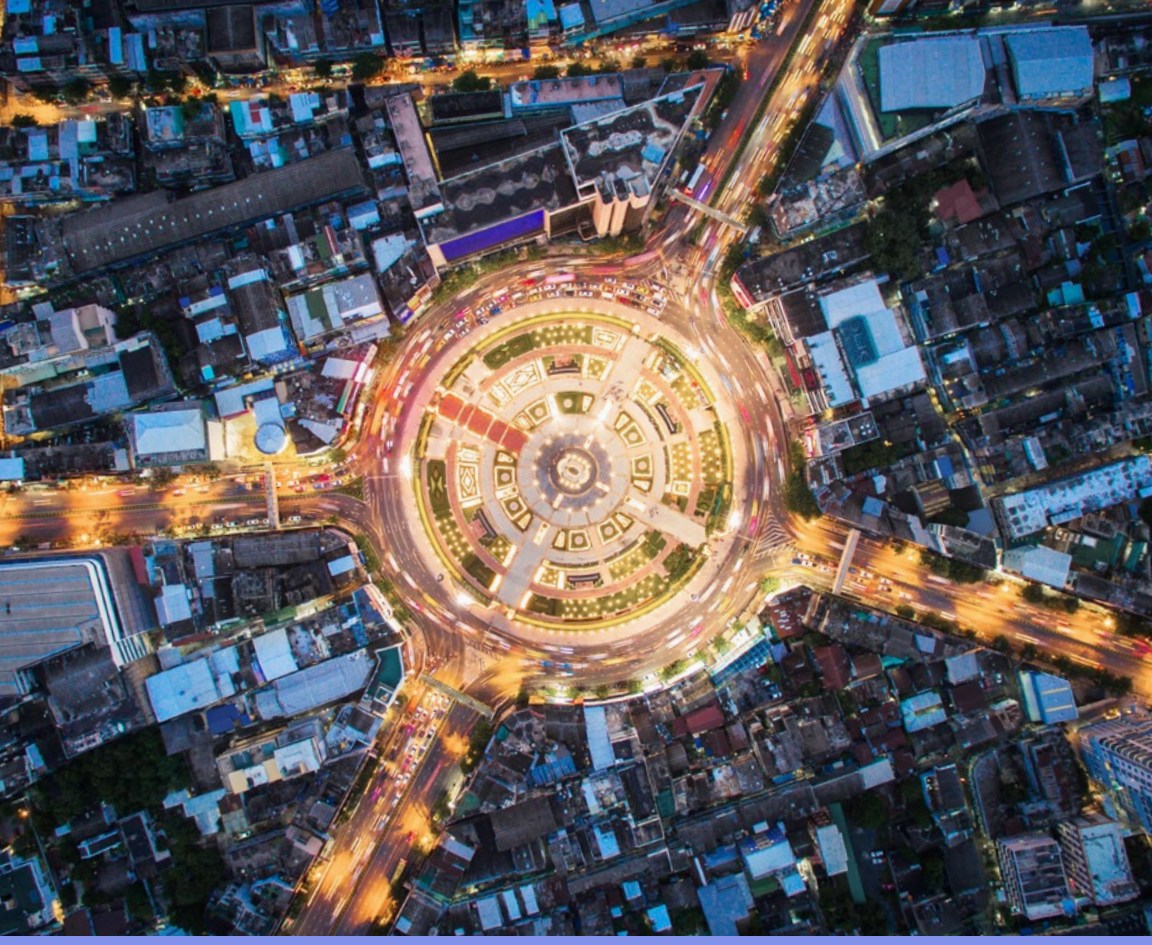


Data Hub Guide for Architects

What is a data hub? How does it simplify data integration?

MarkLogic®

MarkLogic Corporation
999 Skyway Road, Suite 200
San Carlos, CA 94070

+1 650 655 2300
+1 877 992 8885
www.marklogic.com
sales@marklogic.com

Data Hub Guide for Architects

What is a data hub?
How does it simplify data integration?

MarkLogic

Lead Author

Ken Krupa heads MarkLogic's Global Solutions Engineering team and has over 25 years of professional IT experience. Prior to joining MarkLogic, Ken advised some of the largest global financial institutions, delivering technology solutions and advising C-level executives on technology strategy and data architecture. In addition to his consulting work, Ken worked as a direct partner to Sun Microsystems and also served as Chief Architect of GFI Group leading up to the company's IPO.

At MarkLogic, Ken is an enterprise architect who plays the role of pragmatic visionary. He was instrumental in developing the vision for the data hub, and now oversees the customer-facing team that helps implement data hub solutions. With years of experience along with deep technical expertise, Ken is the perfect guide to help fellow architects understand the challenges with legacy technologies and how to solve them with a new approach.

Ken holds a B.S. Degree in Computer Science from the State University of New York, Albany.

Additional Support

This publication is the product of many minds and was made possible with input across the MarkLogic team, including but not limited to:

- Gary Bloom, President & CEO
- David Gorbet, SVP of Engineering
- Dr. Damon Feldman, Sr. Director, Solutions Engineering
- Jason Hunter, Sr. Director, Solutions Engineering
- Justin Makeig, Principal Product Manager
- Alicia Saia, Sr. Director of Solutions

And of course, none of this would be possible without the valuable input from our customers. It is their use of MarkLogic and important feedback that has led to the success and refinement of MarkLogic's data hub software and cloud services.

Contents

> **“Architects need to understand the underlying technical capabilities on the back-end that make it possible to simplify data integration for end users on the front-end.”**

1

Executive Summary

There's a new architecture that's simplifying data integration. It's called a ***data hub***. A data hub is a data store that acts as the central hub in a hub-and-spoke architecture, and is powered by a multi-model database.

The data hub first emerged as a pattern due to a technological shift with databases, specifically NoSQL, multi-model databases. While organizations value their relational databases for handling structured data and querying with SQL, they became frustrated by the lack of flexibility and extensive up-front data modeling they require.

For most data integration use cases, the time and cost associated with a "relational database plus ETL" approach is too great. A multi-model database, by contrast, enables organizations to ingest raw data immediately, lower schema-design costs, and deliver faster value for more use cases.

The data hub approach has evolved, and now MarkLogic® and other vendors provide comprehensive data hub solutions. As a leading vendor for this emerging technology, MarkLogic's Data Hub Platform provides a unified data platform to ingest, discover, curate (enrich, harmonize, master), and access data. And, it is powered by MarkLogic Server, a leading multi-model database.

Unlike data warehouses or data lakes, a data hub is significantly more agile to keep up with today's fast-moving business cycles. A data hub provides transactional and analytical capabilities for both *run-the-business* and *observe-the-business* use cases. And, it has the security and governance required for mission-critical data.

For years, “accepted wisdom” forced architects to design architectures that resulted in a significant accumulation of silos and technical debt. Solving that problem with more point solutions or throwing all the data into a data lake has only created more technical debt. A data hub is a proven approach to simplifying an organization’s architecture, providing greater agility, lower costs, and increased data governance.

With its Data Hub Platform, MarkLogic provides the best of its kind for building a data hub architecture. Along with Data Hub Service, MarkLogic’s cloud service, organizations can do agile data integration and speed their transition to the cloud.

In practice, the MarkLogic Data Hub abstracts much of the technical complexity so that most end users don’t have to worry about modeling entities and harmonizing data. All they have to do is login and explore the end result: well-curated data. But, architects need to understand the underlying technical capabilities on the back-end that make it possible to simplify data integration for end users on the front-end.

The underlying capabilities of the MarkLogic Data Hub Platform, made possible by MarkLogic Server, include:

- Multi-model approach to handle all data types without limitations
- Sophisticated indexing for immediate search
- Ability to represent complex and evolving semantic relationships
- Ability to store data and metadata together for lineage and governance
- Elasticity to handle massive enterprise-wide data volumes
- Robust governance and easy data access for safe data sharing

In the chapters that follow, we will delve into how these underlying capabilities are critical to support a data hub architecture and how that architecture supports today’s rapidly changing business needs.

We’ll start by looking at how businesses accidentally built up piles of technical debt and what is the root cause. We’ll discuss how a rush to build data lakes was the wrong approach. Then, we’ll launch into discussing how data hubs are the right solution — how they really solve the problem, how they work, and how they integrate into your environment.

Our goal is nothing short of changing your perspective of enterprise data integration.

2

Data Integration

How It All Began

Large organizations have multiple departments and/or business units. Each usually maintains its own IT systems, leaving the larger organization with the challenge of integrating the data across departments/units. The *data warehouse* emerged as one of the first enterprise integration patterns to tackle cross-line-of-business integration.

Largely credited to Bill Inmon, the data warehouse was described by him as "a subject-oriented, non-volatile, integrated, time-variant collection of data in support of management's decisions." Additionally, Ralph Kimball – another data warehouse pioneer – described it as a "copy of transaction data specifically structured for query and analysis." By either definition, its purpose was (and still is) to integrate multiple upstream line-of-business systems for subsequent analysis, most often quantitative in nature[1]. Put another way, the data warehouse is designed to support the *observe-the-business* functions of an enterprise.

Though the concept was conceived as early as the 1970s, its adoption began to accelerate in 1990, creating an entire decision support ecosystem within technology, consisting of myriad supporting tools and techniques. Today this segment of the software industry is measured in the billions of dollars annually and estimated to be worth $20B by 2022[2].

Just before the turn of the millennium, however, there was recognition that an additional type of enterprise-scale integration was needed to support *operational* activity across business units. As new business challenges and opportunities

1 https://www.1keydata.com/datawarehousing/data-warehouse-definition.html
2 https://www.marketanalysis.com/?p=65

emerged, applications across these units needed to communicate with each other, while exchanging data to support outcomes that spanned multiple lines-of-business. Put more simply, the various business unit applications needed to communicate with each other in order to support *run-the-business* functions.

To meet these types of integration requirements, a different set of technologies and techniques emerged that centered on real-time message and data interchange between cooperating systems. These approaches included such things as web services, *service-oriented architectures* (SOA), and *enterprise service bus* (ESB) implementations. Categorically the technologies that comprise this space are referred to as *enterprise application integration* (EAI).

The net result of all of this is that enterprises adopted two distinctly different ways of integrating data for run-the-business and observe-the-business domains.

> *“The net result is that enterprises adopted two distinctly different ways of integrating data for run-the-business and observe-the-business domains.”*

The State of Enterprise Architecture Integration Today

With our two integration patterns in mind, let’s fast-forward to present day. If we map out a thirty-thousand-foot view of the data flow within any large enterprise, we would find something similar to the diagram in Figure 1.

Here, on each side of the image, we see the run-the-business and observe-the-business domains. The run-the-business domain consists of a number of applications to support the operations of the enterprise. Integrating the data for these applications is enabled by EAI technologies (SOA, ESB, etc.), while decision support technologies and processes (data warehouses/marts) are used to support observe-the-business integration. We’ll also notice a third zone on the diagram, which we call *enterprise data management* (EDM). This zone is mostly comprised of technologies and implementations that are responsible for “connecting” the two other zones.

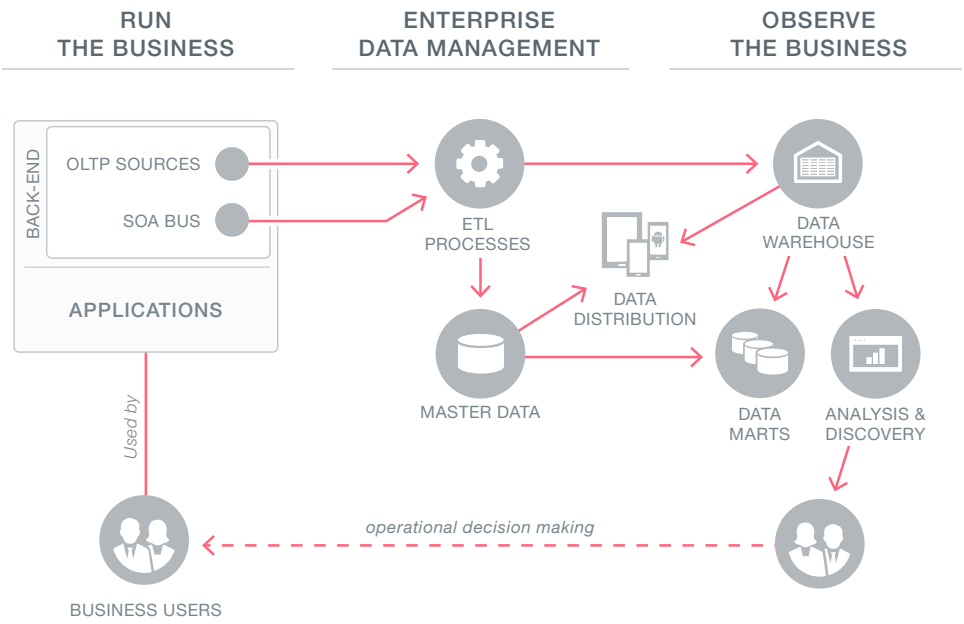


Figure 1: Simplified Enterprise Data Flow

Examples here include things such as extract-transform-load (ETL) products and implementations, as well as processes and tools associated with master data management (MDM). In addition, the EDM zone includes functions related to enterprise data distribution within and outside of the enterprise. At this high level, things appear mostly straightforward with data seemingly flowing freely throughout the enterprise.

However, as we get closer to the details, we begin to see some issues, particularly with respect to data integration. It is with this critical task that we start to see the following characteristics emerge:

- A hard distinction between observe-the-business data processes and run-the-business data processes
- The emergence of a non-trivial enterprise data management function to provide a bridge between the run-the-business and observe-the-business domains
- A dependency on data transformation and data copying throughout the pipeline to meet various goals

On the last point in particular, when we dig more deeply, we can't help but see a very slippery slope as data is copied and changed for *technical* reasons as opposed to valid *business* reasons. What happens is that unnecessary data wrangling leads to problems that accrue over time as an exponentially increasing volume and variety of data must be processed within an ever-shrinking time window. Yet most businesses have relied on the same tools and techniques for the past three decades (or longer)! And when exponentially-increasing requirements are met by only marginally-increased resources – the net result is a significant shortfall that compounds over time.

It's no wonder then that this phenomenon is sometimes referred to as *technical debt*.

“Unnecessary data wrangling leads to problems that accrue over time as an exponentially increasing volume and variety of data must be processed within an ever-shrinking time window.”

3

Our Technical (Data Integration) Debt

The term “technical debt”[3] is often used to describe decisions made during software development that result in future rework, often because shortcuts are taken in lieu of more thoroughly vetted approaches. The term, however, may also be applied to **unintended** consequences resulting from “not knowing any better” and/or as a result of following “accepted wisdom.”

In the case of today’s data integration challenges, it has become clear that much of IT has participated in accrual of technical debt, simply by using accepted tools and techniques. To better understand this, let’s go back to our diagram in Figure 1 and take a closer look at the participants in the data flow. For this, we’ll modify the diagram with numeric labels for cross-reference purposes.

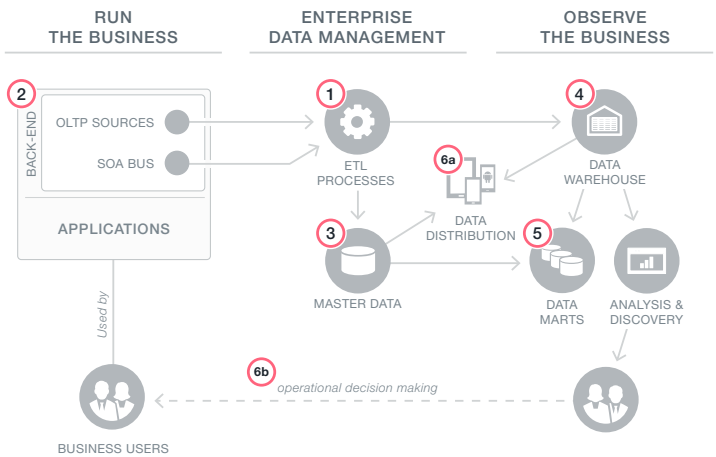


Figure 2: Simplified Enterprise Data Flow, Annotated

3 https://en.wikipedia.org/wiki/Technical_debt

While each of the technologies in figure 2 plays a positive role in the enterprise, we should also understand and acknowledge their unintended contributions to technical debt. In the table that follows, we examine each technology category and assess the positive contributions of each, as well as their respective contribution to technical debt.

Table 1: Impact of Traditional Data Flow on Technical Debt

#	Name	Purpose	Intended Benefits	Impact to Technical Debt
1	**ETL tools and processes**	To automate the transformation of data between different database models across systems.	To accelerate the creation of transformation routines by providing friendly user interfaces (UIs), auto-generation of code, and other process acceleration tools.	High costs resulting from the proliferation of data copying and transformation that impede the enterprise data discovery process. It is estimated that upwards of 60 percent of data warehouse costs are associated with ETL.[4]
2	**Service-oriented architecture (SOA)**	To connect operational business applications to each other to support cross-line-of-business activity as part of enterprise application integration (EAI) via message passing and coarse-grained services.	Accelerate enterprise-wide application-to-application integration by providing abstractions between applications, and scalable dedicated infrastructure for information exchange.	A proliferation of point-to-point application interchange throughout the enterprise. Because the integration is more function-focused and often not data-focused (except for a small amount of message interchange), the result has been to accelerate duplicated data living across data silos.
3	**Master data management (MDM)**	To achieve a "single source of truth," or golden copy, for critical business entities through a set of processes and programs.[5]	To provide consistency for referenced data across multiple applications.	An ambiguous quest to have one data standard across silos. Many challenges are compounded by ongoing ETL. The irony is that many solutions call for creating yet another "golden" copy of the data.
4	**Data warehouse**	To provide the ability to do cross-line-of-business analysis, typically quantitative in nature.	To allow for analysis of the enterprise as a whole and/or for a large complex multi-business-unit portion of the enterprise that would not otherwise be possible without combining the data.	Stale information arises from the heavy dependency on ETL. Also, because of the difficulty integrating disparate data sets, data warehouses contain a small subset of source data entities and attributes, precipitating the need to duplicate more data by way of data marts (see below).
5	**Data marts**	To store narrowly-focused analytical data (unlike the broader data warehouse), often containing a subset of data warehouse data, combined with other domain-specific data not contained in the data warehouse.	To provide similar capabilities to the data warehouse for more domain-specific use cases than are typical of an enterprise-wide data warehouse.	Proliferation of data silos and data copying. As a reaction to the compromises of an enterprise-wide data warehouse (slow pace of delivery, subset of line-of-business [LoB] data), data marts have proliferated.
6	**Data distribution and operational decision making**	To deliver pertinent information to data-dependent stakeholders including, but not limited to, internal decision-makers, compliance officers, external customers, B2B partners, and regulators.	To streamline processes and improve the consistency of data access and sharing.	Because these functions are the farthest downstream in a traditional data integration architecture, they are the most negatively impacted by the challenges of integrating data. These negative impacts manifest as issues associated with time-to-delivery, data quality, and data comprehensiveness.

4 In a report sponsored by Informatica, analysts at TDWI estimate between 60 percent and 80 percent of the total cost of a data warehouse project may be taken up by ETL software and processes.
Source: https://tdwi.org/~/media/TDWI/TDWI/WhatWorks/TDWI_WW29.ashx

5 According to TechTarget: "Master data management (MDM) is a comprehensive method of enabling an enterprise to link all of its critical data to one file, called a master file, that provides a common point of reference."
Source: http://searchdatamanagement.techtarget.com/definition/master-data-management

The net result of all of the above for nearly every large enterprise is an accumulation of technical debt, most of which is related to attempts to integrate data from various silos. This technical debt manifests itself in many ways, resulting in negative impacts on day-to-day business operations and hindering IT innovation. For most large enterprises, the problems seem intractable, leaving many wondering how we got to such a point in the first place.

The Five Whys: Addressing the Root Cause

So, just what is the root cause of this technical debt? Japanese inventor Sakichi Toyoda is largely credited with formalizing the concept of the "Five Whys" interrogative technique for discovering root causes. The technique states that to get to the root cause of a problem, one must simply keep asking "why." When the process is repeated at least five times, a root cause is often identified.

When it comes to the technical debt associated with integrating data from silos, there are often myriad seemingly unrelated business problems arising from a common root cause. Consider the following two examples:

"For most large enterprises, the problems seem intractable, leaving many wondering how we got to such a point in the first place."

Example 1: Customer Service Call Center

Problem: *The amount of time customer service representatives spend trying to service customer requests is very high.*

1. ***Why?*** They may need to potentially search across 16 different systems to find the information they're looking for.
2. ***Why?*** The systems all have data about a customer, some of which overlaps, yet they each have different data models.
3. ***Why?*** They serve different operational functions and the data has yet to be integrated.
4. ***Why?*** Combining that data in a cohesive way, to allow for a customer 360 has proven to be difficult and error-prone.
5. ***Why?*** Creating a relational database model must consider all data model variances up front, and the schema must be created before development can begin.

In the above case, the root cause lies with limitations of modeling with a relational database management system (RDBMS), where schema modeling is a prerequisite activity to development. Moreover, as we'll discuss in the subsequent section, because nearly every model change in a relational database is often accompanied by non-trivial code and back-testing changes, modelers attempt to design schemas to account for as many scenarios as possible, potentially making the modeling exercise very complex and time-consuming. In many cases, because of complexity, compromises are made in the modeling process in an attempt to meet a deadline or otherwise "save time."

Example 2: Investment Bank – Derivatives Post-Trade Processing, Project Delivery

Problem: *After more than 18 months, the project team still has not started development and does not expect to start for another 3-6 months.*

1. ***Why?*** The data model is not finished.
2. ***Why?*** They haven't accounted for all of the asset classes.
3. ***Why?*** Every time they look at a new asset class, the model has to be redone.
4. ***Why?*** The source models of each entity are very different, causing difficulty for the modeling team.
5. ***Why?*** Creating a relational database model requires considering all data model variations up-front and the schema must be created before development can begin.

Again, despite the different business use case, the root cause lies with the limitations of RDBMS schema, where integrating multiple valid models is difficult, time-consuming, and brittle.

The Challenges With Relational Databases

In the prior section, using the Five Whys technique, we identified limitations of RDBMS technologies as a common root cause of data-related technical debt. In reality, however, we can go even further and ask the question of "why" exactly integrating data using RDBMS models is difficult. In this section, we'll answer that question, with a bit more detail than a single-sentence answer.

RDBMS Data Representation

An RDBMS represents data by using tabular structures, similar to a spreadsheet, albeit with more controls. Data is stored as rows of information described by column attributes. When we want to model a business entity, we create one or more tables to hold that information as in the customer example below:

Table: Customers

id	first_name	middle_name	last_name	birth_date
1	John	Q	Public	1970-01-01
2	Austin	Danger	Powers	1941-09-06

We create more tables when we need to represent more complex entities or more complex information about entities. For instance, a customer may have more than one phone number or address, which would indicate the need to create other tables to capture such data.

Table: Phones

id	cust_id	type	country_code	area_code	number
1	1	home	1	212	555-1234
2	1	mobile	1	646	555-1111

Finally, we must represent relationships between data items, by using primary keys and foreign keys to link data together between tables.

Tables: Customers & Phones

primary key (id)				
id	**first_name**	**middle_name**	**last_name**	**birth_date**
1	John	Q	Public	1970-01-01

	foreign key				
id	**cust_id**	**type**	**county_code**	**area_code**	**number**
1	1	home	1	212	555-1234
2	1	mobile	1	646	555-1111

The more complex an entity becomes, the more tables and relationships that are needed to represent the business entity. Likewise, the more business entities we have, the more complex the overall modeling process becomes. For a large business unit within an enterprise, the modeling process can be very significant.

Relational Databases and Change

With relational databases, the models are represented as sets of *constraints*. In other words, only data that conforms to a *predefined* set of schema may be processed by the database.

Think about that for a moment – and the implications.

The people responsible for creating schema (DBAs mostly) are on the *critical path* to delivery, since data cannot be loaded into an RDBMS without a defining (and constraining) schema. Furthermore, those same people have the profound task of making sure that they get things right up-front. After all, they are making or finalizing decisions around what data is welcome (and not welcome) inside the database.

This not only has the potential to negatively impact delivery times at the inception of projects, but also every time there is a requirements change that impacts the database schema. This can be particularly troublesome for applications where the development has already begun, or for those that have loaded data and/or have gone into production.

This is because changing a database schema is often a non-trivial task, where activities might include:

- Taking a snapshot of any data that may have already been loaded
- Creating new models and/or altering existing models to provide new columns and/or tables to allow for new data
- Changing ETL code and/or other existing application code to account for the database changes
- Re-creating indexes so that the database may be queried efficiently
- Re-testing all dependent applications and processes

In other words, relational databases do NOT handle change very well.

What we're left with is a very rigid process, where even the smallest of changes comes at a steep price. Furthermore, because relational databases force data into rows and columns, business entities are not always modeled in ways that represent their natural state. Finally, because of the limited ways in which relationships may be represented in a relational database, a good deal of the business context about relationships between entities is not explicitly captured.

Which Schema Is Correct?

Large enterprises consist of multiple business units. Many of these business units deal with overlapping data across multiple applications and databases. This means that these different business units will often have differing – and sometimes conflicting – "opinions" as to how some of the same business entities should be modeled. For instance, a customer on-boarding application may model a customer entity one way, whereas a downstream customer service application may have its own system and model for customer data.

In an attempt to "standardize" the customer entity across the enterprise, a customer relationship management (CRM) system will decide that it should contain the "master copy" of customer data with yet another model definition. Finally, to create a downstream data warehouse for analytical purposes, the customer data would then be transformed into yet another model to support analytical activities, leaving the source models behind. Because relational databases have to conform to a single model for a limited context, the database not only has trouble with handling differing representations of business models, but also exacerbates the problem by sometimes enforcing differing models purely for technical reasons.

Enterprise data integration and complexity

In the previous section we refer to "handling business change" as a weakness of RDBMS technology. However, such a statement begs the question as to why the use of RDBMS technology is so prevalent. After all, doesn't technology always adapt to the changing needs of the business? And if so, why have we leaned on technology that is so ill-suited to change? The answer is a matter of perspective.

From within the context of a large enterprise, changes to the business may arrive in a multitude of ways. Some changes are minor, others more impactful. The frequency of change may also vary, as well as which business units within a large organization are impacted. Sometimes changes are (seemingly) isolated to a single department. Other times, multiple groups within an organization are affected. And, it is on this last point – the difference between single-department and enterprise perspectives – that we may best appreciate the impact of change with respect to modern data integration.

When viewed through the lens of a single department, many business changes may be viewed as "manageable" with respect to how the changes impact the underlying data models. Thirty to forty years ago, this was certainly the case. Through the 1980's and even into the 1990's, delivery time expectations for system changes were much longer. And, the volumes and variety of data managed by systems were significantly less. Additionally, the interdependence between different departments' data was not nearly as great as it is today. They were simpler times indeed.

Today, things are more complicated – even when viewed through the lens of a single department. Data is greater in volume and variety, and the turnaround time for responding to change has shrunk. As a result, various technologies and techniques have sprung up in response to these challenges, oftentimes to aid the RDBMS technologies for which the various departments' systems were built. The "middle tier" emerged to map business entities to the underlying databases, supported by object-to-relational mapping (ORM) tools. Various agile delivery methods also sprung up to shrink the delivery cycle from years to months (or less). By and large, the new tools and techniques worked (and still do) to address the complications, with one significant caveat. The context within which they work is within the perspective of a single set of related business problems, such as for a single department or division within a larger enterprise.

However, when one considers multiple business units or divisions, with different budgets, leaders, perspectives and people, things go from complicated to complex. And it's at this inflection point of complexity that RDBMS technology breaks down.

In his bestselling book *Team of Teams*, U.S. General Stan McChrystal references mathematician Edward Lorenz's famous "butterfly effect" to make a distinction between problems that are complicated and ones that are complex.[6] He proposes that *complicated* problems have a level of predictability about them that makes it possible to address them systematically, oftentimes by applying efficiencies to existing processes. For *complex* problems, however, he speaks of an increasing number of interdependent interactions that emerge, making the system too unpredictable for traditional systematic processes to address. With these complex problems, a more agile approach is required to solve them, one that "embraces the chaos" so to speak, by recognizing that rigid processes can only go so far.

This is exactly the case with today's challenges around enterprise data integration. There are far too many interdependent systems within a large enterprise to assume that the rote processes associated with RDBMS data modeling will scale to meet the level of complexity of the overall system. In other words, we have crossed the complexity threshold.

This distinction between complicated and complex is applicable to data management when considering the differences between modeling data for a business-unit/department and modeling data for purposes of data integration across an enterprise. The former may become complicated, however the latter is always complex.

Consider the following comparison:

Table 1 - Departmental vs. Enterprise Modeling

Department/Business-unit modeling	Enterprise modeling
Single system or small set of related systems.	Multiple interdependent systems, some seemingly unrelated.
Manageable requirements under one set of business owners.	Frequently changing requirements across multiple business owners.
Relatively stable model that may follow a set process for managing change.	Interdependent models resulting in frequent changes.
Solvable complications resulting in many successful implementations.	Spotty track record for enterprise integration when relying on RDBMS-based tools exclusively.

The key takeaway here is that modeling business entities for the enterprise (i.e. modeling for data integration purposes across business domains) requires an altogether different approach to modeling than the traditional approach that has been used for decades. In subsequent sections, we explore these new modeling methods and learn how a data hub makes such modeling possible.

6 McChrystal, Stanley. *Team of Teams: New Rules of Engagement for a Complex World*. Penguin Random House. 2015.

New Technologies, New Integration Challenges

As should now be obvious, the flow of data through the modern enterprise is not simple, and it leaves technologists with significant technical debt. As new technology continues to be introduced, data architectures will only get more complex. There is always shiny new technology. Let's consider three recent industry tech trends that have impacted the enterprise:

- **Big data.** The three Vs of the "big data" trend – volume, velocity, and variety – cast a spotlight on the notion that traditional data management approaches were ripe for disruption. As a result, a number of technologies emerged that created a good deal of hype. Although there has since been inevitable disillusionment and in some cases outright abandonment (i.e. Hadoop), we have benefitted by the increased focus on the transformational power of data and the recognition that enterprises do need a way to handle the big "V's" (volume, velocity, variety, and also veracity).

- **Cloud.** The mainstream adoption of various as-a-service offerings across the entire infrastructure and application continuum is a major disruptive force. SaaS deployment dramatically reduces the time to get an application running, and reduces the ongoing maintenance burden for large enterprises. Through the lens of data integration, there is also a catch: there are a greater number of operational applications for which data must be integrated.

- **Machine Learning (ML) and Artificial Intelligence (AI).** Occurring somewhat hand-in-hand with the explosion of data, there has been a significant advancement in the fields of machine learning and artificial intelligence (ML/AI). And, such a co-occurrence is not a coincidence. Advancements with ML/AI are heavily dependent on large amounts of data required to train the models. The promising early results associated with ML/AI continue to feed the demand for more and more data. But again, there's a catch — the data must be well-governed, quality data as the output of the models are only as good as the inputs.

These examples reinforce the fact that technology does not stand still (nor do business needs), with potential disruption at any turn. They also show that with new opportunities, there are also unintended consequences.

“...the power of a pattern... lies in the fact that they are discovered from best practices that are based on evidence of repeated success.”

To illustrate this point, let us look at how enterprises fared with the hype surrounding data lakes. We choose this example because it speaks to data integration challenges, and because the places where data lakes are deficient are areas specifically addressed by a data hub. After this brief assessment of data lakes, we'll dive right into data hubs for the remainder of the text.

The Rise and Fall of Hadoop Data Lakes

When the "big data" hype cycle began, among the issues that were purportedly addressed were those related to the inflexibility of modeling data inside of an RDBMS. The thinking went that by creating data lakes of enterprise data, the need for prerequisite modeling would be ameliorated.

As a result, large enterprises rushed to create these data lakes using a variety of technologies from within the Hadoop ecosystem. For the most part, the basic design of the data lake involved dumping large volumes of raw data into the Hadoop Distributed File System (HDFS), and then relying on a myriad of programming tools to make sense of this raw data. There were some gains; however, as with all hype cycles, there was also a lot of disillusionment.

Following are some of the areas where data lakes came up short:

- **Operational integration.** Though data lakes expand the scope of data processing (e.g., the big data "Vs"[7]), they focus mostly on integrating data to support analytics or observe-the-business functions. In other words, operational integration is left to legacy technologies.

- **Security.** Many early data lakes took a *laissez-faire* approach to data security, pushing the problem to "other layers" in the architecture. Even newer ones have taken an "after the fact" approach to securing data in the lakes, resulting in a compounding of data security concerns.

- **Harmonizing disparate models.** Proper integration of data requires some kind of reconciliation of varying definitions of similar business entities. Legacy ETL processes do this, but only with the prerequisite rigid modeling requirements (resulting in compromised models), and usually accompanied by traceability issues. Most data lakes simply call for dumping data into large HDFS clusters. But, this approach does not address the critical activity of mapping disparate data to business entities, often leaving it to data access code to perform transformation. This has led to data quality issues with many lakes, oftentimes at a very large scale.

7 http://whatis.techtarget.com/definition/3Vs

- **Provenance and lineage.** Being able to report on *how* a query arrived at a result (e.g., where the data was sourced and how it may have been transformed) is often as important as the result itself. This activity is sometimes called data *lineage* or *provenance*. Because most data lakes simply load data into HDFS, there is often no provenance, lineage or similar traceability associated with the loaded data.

- **Data governance.** The net result of a lack of security, provenance, and lineage is a lack of overall data governance. Many data lake implementations are ungoverned from the outset, resulting in a compounding of data quality and related governance concerns. Even newer data lakes that consider data governance as a critical activity employ a post-data-load strategy that often compounds existing, already-difficult enterprise data governance initiatives.

- **Trading business silos for technical silos.** The Hadoop ecosystem consists of a multitude of tools and technologies that must be pieced together to address data integration. However, this approach often trades business silos for technical silos, where data is duplicated across different types of technologies unnecessarily, creating a new kind of technical debt.

The result for many enterprises has been an *exacerbation* of many pre-existing problems, even as other problems get addressed. In most cases, the overall effect is a net increase in the pain associated with managing data. It is for these and other reasons that Hadoop and data lakes fall short of expectations.

Modeling Complexity: The Enterprise Entity

It is a very complex endeavor to consider the perspectives that multiple business units have about data integration, and even more so when accounting for how things may change over time. For this reason, some architects conclude that it is easier to just give up and simply throw all of the data together and/or just not model it at all, or just leave it to downstream consumers to sort through it and create models as they see fit. However, as recent history has demonstrated, such approaches have been tried and failed. Quality suffered greatly, governance and security challenges were significantly exacerbated, and promises to the business were left unfulfilled.

As it turns out, modeling business entities is still a critically important activity, particularly for data integration. However, if the rigid "boil the ocean" modeling associated with relational databases does not work, and neither does the hand-wavy, 'pass-the-buck' modeling (or non-modeling) approach of data lakes, then what does work? How do we model complex, changing data sets in an agile way?

To answer these questions, we will start by giving a name to one of the primary deliverables of the modeling approach: the **Enterprise Entity**.

The term "entity" is well known in data modeling. Relational database modelers are familiar with it within the context of an entity-relationship diagram (ERD) where a business entity (or more often part of a business entity) is transcribed into a table in a relational database. In the object-oriented world, more fully composed business entities might be referred to as classes. In each case though, the general concept is similar, namely the things (i.e. nouns) that a business cares about (e.g. customers, products, pharmaceutical compounds, stock trades, insurance claims, oil rigs, etc.) are modeled as some kind of entity.

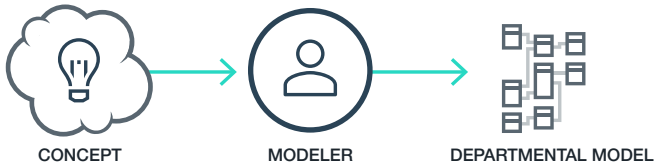


Figure 3: Departmental modeling

When it comes to data integration, the entities to model must be crafted from a collection of other previously modeled entities as opposed to "modeling them from scratch." In other words, we are creating a "model of models" at the enterprise level.

These diagrams depict how modeling data for a department differs from modeling data for the enterprise:

Figure 4: Enterprise modeling

As Figure 4 shows, the enterprise-wide model (consisting of multiple entities) has a high inter-dependency on multiple departmental models (also consisting of multiple entities). To achieve this, we cannot simply scale the departmental approach to meet enterprise integration needs. This is where the enterprise entity modeling concept comes into play.

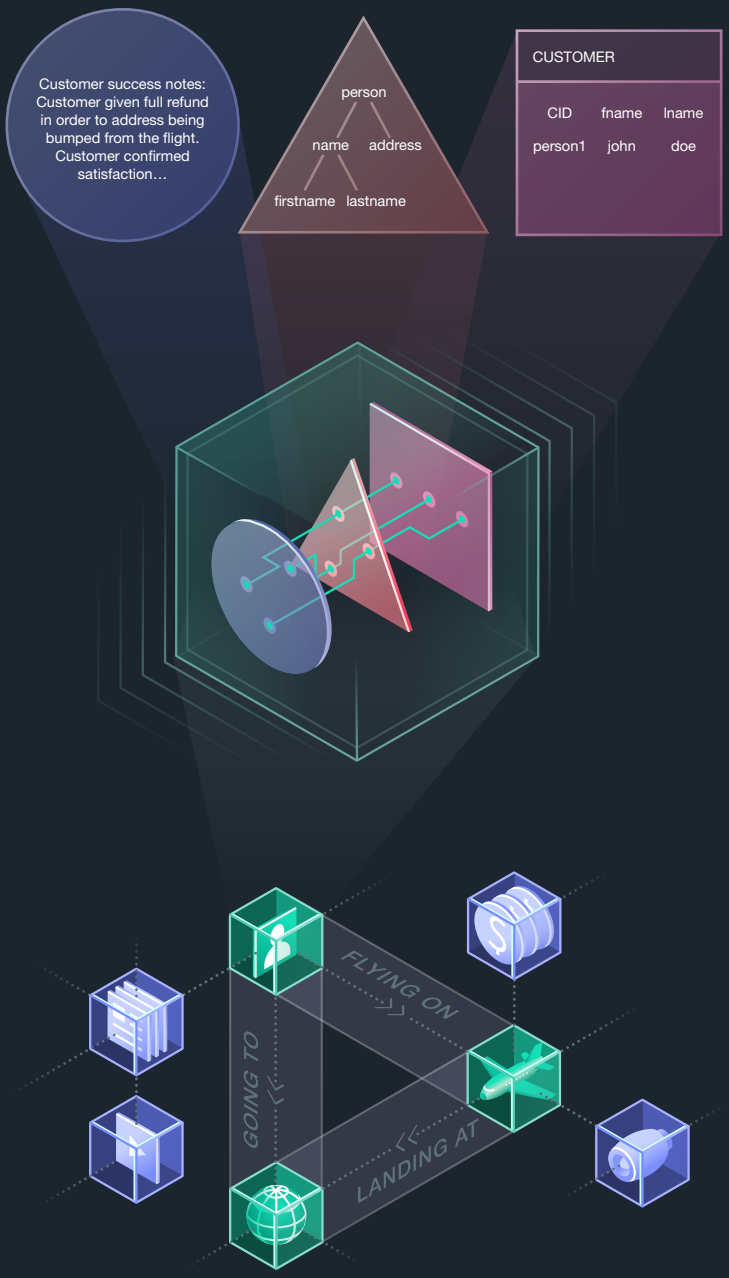
To model an enterprise entity, the following must be achieved:

- The entity model must be capable of representing multiple source models of the same business entity simultaneously. In other words, if there are multiple sources of customer data with overlapping information, the enterprise entity should be able to retain all of the original source models for that customer.
- The model must also be able to support the creation of a canonical model that is separate from, and in addition to, the source models. The intention of the canonical model is to harmonize structural differences, naming differences, and semantic differences. Additionally, this canonical portion of the model must be flexible such that it may change over time.
- Relationships between enterprise entities should be flexible and contextual. The approach should allow for any entity to be related to any other entity without remodeling and in a way that is explicitly descriptive.
- As the enterprise entity undergoes changes (e.g. new attributes are added, additional source models are incorporated, new relationships emerge, etc.) metadata about the changes must be storable at the instance level.

The creation of such a model for a given entity results in a richer representation that more closely mirrors reality. Instead of trying to merge and flatten multiple source models into a least-common-denominator model subset, we instead get a multi-dimensional model superset that is rich with information, and dynamic enough to handle ongoing changes.

Conceptually, a high-level representation might look like this:

Multi-dimensional Enterprise Entity Model



What the preceding picture progression shows is three differing source models for a customer, how they are retained, linked, harmonized and enriched into an enterprise entity, and then finally how they are linked with other enterprise entities to create a rich data integration graph.

With this conceptual building block now in place, we will turn our attention to the Data Hub itself and show how it is a foundational pattern for creating and maintaining these enterprise entities. In doing so, we will also get a primer on how to model an enterprise entity by way of a simple example.

4

The Data Hub Pattern Emerges

Here we are at last. Admittedly, it has taken us almost halfway way through the text to accurately describe more than three decades of data integration technical debt. But, that's what it takes to truly understand what is involved!

We finished the last section with a brief discussion of the "big data" data lake hype cycle. And we arrived here at a transition into something that sounds much more mundane – a pattern.

At MarkLogic, we have since productized the data hub to become our flagship offering, but it first emerged as a new enterprise architecture pattern to address data integration. This is important to note because as a pattern, it was based on best practices and evidence of repeated success with real customers. This is in contrast to many fads that are often *contrived* on less-proven theories or have never been used to solve actual business problems.

"Pattern" Explained

In technology, patterns were first popularized with software development (i.e., code), and later expanded into more coarse-grained concepts such as inter-system design and data flows (e.g., enterprise architecture). In all cases, technology patterns are *discovered* when looking across a number of successful implementations for common best practices.

When a collection of these best practices can be coalesced into an implementable thing that is distinct from other previously implemented things, it is then described and codified as a pattern worthy of repeating within certain defined scenarios.

Although our Data Hub Platform is a fully-baked product – database and UI included – it is still helpful to think of the general data hub architecture as a pattern or blueprint. As such, it is similar to other patterns such as the data warehouse or data lake.

Data Hub: A Formal Definition

A ***Data Hub*** is an authoritative multi-model data store and interchange, where data from multiple sources is curated in an agile fashion to support multiple cross-functional analytical and transactional needs.

To understand the rationale behind the definition, let's break it down into its components. Then we can talk about the principles behind it, the inner workings, as well as the type of technology needed to implement it.

Table 2: Breaking Down the Data Hub Definition

"An authoritative multi-model data store and interchange…"	Multi-model within this context is itself layered. Later, we will cover the concept of a multi-model database as technology that provides multiple modeling techniques (e.g., document, RDF triple, tuple, etc.) to represent data. That is true of the larger data hub pattern. Another complementary perspective is that the data hub may also simultaneously represent multiple different models for similar business entities. The key point here is that models are so important for data integration, that they must be afforded the same agility as data, instead of treating them as rigid constraints. The reference to an authoritative data store and interchange refers to the data hub as the best, first place to integrate data. In other words, it functions as an authoritative **integration point** as opposed to a "system of record" for all data.
"… where data from multiple sources is curated in an agile fashion…"	Data curation in the data hub context refers to the ability to enrich data (add more value with metadata, etc.), harmonize that data (synthesize all source data, without having to decide which data to "throw away"), and master the data (match and merge data like an MDM tool). And, it's all done using an agile approach in which only some of the data can be curated and made fit-for-purpose at a given time without a major up-front modeling project.
"… to support multiple cross-functional analytical and transactional needs."	A data hub supports the entire enterprise data lifecycle (e.g., run-the-business, observe-the-business) as opposed to part of it. It is also able to perform transactional business functions that are only possible when combining data from multiple lines-of-business. It typically serves multiple downstream consumers of enterprise data, often for different purposes.

> *"Models must be afforded the same agility as data, instead of treating them as rigid constraints."*

Principles of a Data Hub

Though data hub implementations may vary from one to the other, they all have in common certain foundational principles as follows:

1. **Use of document/object models to represent business entities.** Self-describing documents (such as XML or JSON) are a natural way to represent business entities or objects. They do not suffer from the so-called "impedance mismatch"[8] associated with object-to-relational mapping, and come with many benefits such as:

   a) Ability to treat **schema as data**, given that every payload may also include schema information. This is what gives schema and models the same level of agility as the data itself.

   b) Ability to allow for multiple models representing the same class of business entity. For example, multiple systems may model customer data in different ways. In a data hub architecture, all of those models may be represented concurrently.

   c) Ability to store metadata and data together, allowing for provenance and lineage to be captured, and providing a strong foundation for data governance.

2. **Data Curation and Harmonization.** Most approaches to integrating disparate data models involve coming up with a new model followed by attempts to "force fit" (by way of ETL) all source data into the new model. Data curation in a data hub, on the other hand, starts with the premise that all source models are not only valid, but also *valuable*, and hence should be retained *as is* in an integrated context. These source models are then leveraged to create an integrated canonical model (or models) on an as-needed basis, all the while recording valuable provenance and lineage metadata inside the data hub itself. The result is that instead of a lowest-common-denominator subset of integrated data, the data hub creates an *agile superset* of the source data.

3. **Use of Semantic RDF triples to represent relationships.** The Resource Description Framework (RDF) is a set of W3C standards for representing machine-readable concepts about things and relationships between things. It also forms the basis for the concept of the Semantic Web[9]. The unit of representation is called a *triple*, which consists of a *subject*, a *predicate* and an *object*, collectively comprising a fact/concept or a relationship (e.g., "Euro typeOf currency"

8 https://www.techopedia.com/definition/32462/impedance-mismatch
9 https://www.w3.org/2001/sw/

or "Thor sonOf Odin"). In a data hub, RDF triples provide a myriad of capabilities with respect to managing data and the complexities of data integration. This will be discussed in a bit more detail in the sections that follow.

4. **Indexing to support real-time ad-hoc query and search.** Unlike a data lake that depends on subsequent brute-force processing for data querying, a data hub indexes *all* data on ingestion to ensure that data is queryable as soon as it is loaded.

5. **Support for bidirectional data access.** Unlike patterns that support either "run-the-business" or "observe-the-business" functions, the data hub supports both. By allowing real-time updates with transactional support, alongside the ability to impact schema and data in a way that may be tracked and audited, the data hub is a safe place for direct updates to integrated data without negatively impacting data governance and accuracy.

In the subsequent section, we'll demonstrate by example how the above principles are used together.

Self-describing Schema and Data Harmonization

As previously mentioned, the use of self-describing documents allows multiple different schema representations of the same business entity to coexist, with little to no DBA intervention. We've also mentioned the use of semantics and RDF as a more powerful way to represent relationships and concepts. In this section, we'll demonstrate the technique by way of example. The examples will leverage what is known as the *envelope pattern*. Here we're using the term *pattern* in a slightly more fine-grained context, referring to a data modeling technique that allows incoming data to be "wrapped in an envelope" such that changes to that data may be recorded in other sections of the document object.

For the most part, you may view the envelope pattern as an implementation detail for which the more important point is that the original data is left unaltered[10] – a crucial feature for both governance and provenance.

> *"The data hub is a safe place for direct updates to integrated data without negatively impacting data governance and accuracy."*

10 Other techniques may accomplish similar things, however the envelope pattern does so within the context of a single document, which is easier to demonstrate.

Consider the following models for a customer:

```
<!-- Customer model for John Smith -->
<cust id=”123”>
   <name>
      <first>John</first>
      <last>Smith</last>
   </name>
   <addr1>1 Avenue A</addr1>
   <city>New York</city>
   <state>NY</state>
   <zip_code>10009</zip_code>
</cust>

<!-- Customer model for Jane Smith -->
<customer>
   <cust_id>ABC</cust_id>
   <fname>Jane</fname>
   <lname>Smith</lname>
   <address>
      <street1>1 Avenue A</street1>
      <street2>Apt Z</street2>
      <city>New York</city>
      <state_or_province>NY</state_or_province>
      <postal_code>10009</postal_code>
      <country>US</country>
   </address>
</customer>
```

In each example, the address is represented slightly differently. In example 1, there is a field referring to `zip_code`, whereas in example 2, `postal_code` is used. Because these two schema attributes are also treated as data items, they may be indexed and searched. Therefore, it is possible to execute a query searching for either `zip_code` or `postal_code` for a given value (e.g., 10009 in this case).

This capability is important, **in that no prerequisite schema definition or complex ETL is required to perform discovery operations** on the data. However, as useful as this is, there are nonetheless valid reasons to also create and maintain standard (i.e., canonical) data models, particularly with respect to enterprise data integration. One might therefore argue that the traditional ETL approach, which enforces model standardization (albeit with a lot of up-front effort), provides a much-needed capability. And while that may be true in some sense, when it comes to creating and managing canonical models, the data hub pattern is much more flexible and ultimately much more powerful.

To demonstrate this, we revisit our customer records for John and Jane Smith:

```
<!-- Enveloped model for John Smith with provenance -->
<customer>
   <metadata>
      <transformation type="CanonicalTransformer">
         <sem:triple>
            <sem:subject>/env/object/12345.xml</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#wasGeneratedBy
            </sem:predicate>
            <sem:object>http://client.com/postTransform_01</sem:object>
         </sem:triple>
         <sem:triple>
            <sem:subject>http://client.com/postTransform_01</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#endedAtTime
            </sem:predicate>
            <sem:object
               datatype="http://www.w3.org/2001/MLSchema#dateTime">
               2018-01-01T12:01:42.987
            </sem:object>
         </sem:triple>
      </transformation>
   </metadata>
   <canonical>
      <postal_code>10009</postal_code>
   </canonical>
   <source>
      <cust id="123">
         <name>
            <first>John</first>
            <last>Smith</last>
         </name>
         <addr1>1 Avenue A</addr1>
         <city>New York</city>
         <state>NY</state>
         <zip_code>10009</zip_code>
      </cust>
   </source>
</customer>
```

```
<!-- Enveloped model for Jane Smith with provenance -->
<customer>
   <metadata>
      <transformation type="CanonicalTransformer">
         <sem:triple>
            <sem:subject>/env/object/12340.xml</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#wasGeneratedBy
            </sem:predicate>
            <sem:object>http://client.com/postTransform_01</sem:object>
         </sem:triple>
         <sem:triple>
            <sem:subject>http://client.com/postTransform_01</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#endedAtTime
            </sem:predicate>
            <sem:object
               datatype="http://www.w3.org/2001/XMLSchema#dateTime">
               2018-01-01T12:01:42.988
            </sem:object>
         </sem:triple>
      </transformation>
   </metadata>
   <canonical>
      <postal_code>10009</postal_code>
   </canonical>
   <source>
      <customer>
         <cust_id>ABC</cust_id>
         <fname>Jane</fname>
         <lname>Smith</lname>
         <address>
            <street1>1 Avenue A</street1>
            <street2>Apt Z</street2>
            <city>New York</city>
            <state_or_province>NY</state_or_province>
            <postal_code>10009</postal_code>
            <country>US</country>
         </address>
      </customer>
   </source>
</customer>
```

In the above examples, we use the aforementioned envelope pattern. As the name implies, we simply wrap the original data payload into a metaphorical envelope, so that we may enhance it with other information without altering the original contents. In the examples, we also created additional sections, one of which is referred to as the *canonical* section. Here, we added another attribute, `postal_code`, and placed in it the values from the zip and postal fields of the respective records for John and Jane Smith.

By using this approach, we realize the following three benefits:

1. All original source data has been preserved (crucial for governance) without having to make up-front and potentially lasting decisions about which source data is important or not.
2. All original **models** are also preserved and searchable.
3. We have begun to create a canonical model on an **as-needed** basis, as opposed to trying to figure out all possibilities up-front.

Also contained in the sample data is a newly added metadata section. Ignoring some of the syntactical particulars for a moment, the astute reader will notice what appears to be information about source systems and timestamps. This is exactly the case, as that section of the document contains examples of *data provenance* being maintained alongside the data itself, using what is called PROV-XML, which is a serialization format conforming to what is known as the Provenance Ontology or PROV-O[11].

PROV-O is a W3C standard for recording provenance metadata in a machine-readable way. Though full coverage of PROV-O is outside the scope of this document, it is important to note that by way of the envelope pattern, PROV-O metadata may be captured alongside the data to which it refers. This technique of maintaining important metadata alongside the referenced data is a key principle of the data hub.

Semantics and RDF Triples

Thus far, we've focused on the advantages associated with using documents and the envelope pattern to provide agility. Semantics and RDF triples also contribute to flexible modeling and agility by providing the ability to manage facts, concepts and complex relationships associated with data, within a very rich context. This is illustrated by continuing with our harmonization example as follows:

11 https://www.w3.org/TR/prov-o/

```
<!-- John Smith with spouse triple -->
<customer>
   <metadata>
      <transformation type="CanonicalTransformer">
         <sem:triple>
            <sem:subject>/canonical/object/12345.xml</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#wasGeneratedBy
            </sem:predicate>
            <sem:object>http://client.com/postTransform_01</sem:object>
         </sem:triple>
         <sem:triple>
            <sem:subject>http://client.com/postTransform_01</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#endedAtTime
            </sem:predicate>
            <sem:object
               datatype="http://www.w3.org/2001/XMLSchema#dateTime">
               2018-01-01T12:01:42.987
            </sem:object>
         </sem:triple>
      </transformation>
   </metadata>
   <triples>
      <sem:triple>
         <sem:subject>/canonical/object/12345.xml</sem:subject>
         <sem:predicate>http://client.com/rels#spouseOf</sem:predicate>
         <sem:object>/canonical/object/12340.xml</sem:object>
      </sem:triple>
   </triples>
   <canonical>
      <postal_code>10009</postal_code>
   </canonical>
   <source>
      <cust id="123">
         <name>
            <first>John</first>
            <last>Smith</last>
         </name>
         <addr1>1 Avenue A</addr1>
         <city>New York</city>
         <state>NY</state>
         <zip_code>10009</zip_code>
      </cust>
   </source>
</customer>
```

```
<!-- Jane Smith with spouse triple -->
<customer>
   <metadata>
      <transformation type="CanonicalTransformer">
         <sem:triple>
            <sem:subject>/env/object/12340.xml</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#wasGeneratedBy
            </sem:predicate>
            <sem:object>http://client.com/postTransform_01</sem:object>
         </sem:triple>
         <sem:triple>
            <sem:subject>http://client.com/postTransform_01</sem:subject>
            <sem:predicate>
               http://www.w3.org/ns/prov#endedAtTime
            </sem:predicate>
            <sem:object datatype="http://www.w3.org/2001/XMLSchema#dateTime">
               2018-01-01T12:01:42.988
            </sem:object>
         </sem:triple>
      </transformation>
   </metadata>
   <triples>
      <sem:triple>
         <sem:subject>/env/object/12340.xml</sem:subject>
         <sem:predicate>http://client.com/rels#spouseOf</sem:predicate>
         <sem:object>>/env/object/12345.xml</sem:object>
      </sem:triple>
   </triples>
   <canonical>
      <postal_code>10009</postal_code>
   </canonical>
   <source>
      <customer>
         <cust_id>ABC</cust_id>
         <fname>Jane</fname>
         <lname>Smith</lname>
         <address>
            <street1>1 Avenue A</street1>
            <street2>Apt Z</street2>
            <city>New York</city>
            <state_or_province>NY</state_or_province>
            <postal_code>10009</postal_code>
            <country>US</country>
         </address>
      </customer>
   </source>
</customer>
```

You'll see we've added another section – named "triples" – to each document. In these sections, we have inserted RDF triples that assert a relationship between the documents for John and Jane Smith, namely that they're spouses. Though this example is simple, it is already apparent how expressing such a relationship is more powerful than how relationships are expressed in an RDBMS.

Additionally, we can begin to imagine how more complex and interesting sets of facts and relationships may be expressed. For instance, the aforementioned PROV-O standard is expressed by way of RDF as triples. Expressing data as triples is quite powerful, so much so that it is the underpinning of the promise of the *semantic web*[12]. In other words, the promise and potential of semantics and RDF triples go far beyond the domain of data integration and what we'll discuss with respect to the data hub.

As for the smaller (yet still expansive) scope of what's possible with respect to data integration, the following is a brief list of examples:

- **The ability to represent relationships in a natural and agile way.** In an RDBMS, representing a relationship between two entities is quite rigid. A modeler essentially chooses a *cardinality* (e.g., one-to-one, one-to-many, many-to-many), and encodes these choices as *constraints* via *primary keys* and *foreign keys* across a number of *tables*. Once these constraints are in place, they may not be violated unless and until a DBA gets involved to change the rules. Triples, on the other hand, are not constraints but instead are data items that are created at any time for any entity. Any time a business relationship occurs – a triple may be created.
- **The ability to explicitly encode context and intent.** In an RDBMS world, relationships are typically devoid of explicit context, requiring some implicit knowledge of the designers' intent. RDF triples, on the other hand, encode full context by naming the type of relationship (i.e., the predicate) between entities (subject and object).
- **The ability to create complex graphs of relationships between things.** For example, representing a social graph (e.g., friends, friends of friends, etc.) is quite easy using RDF, and there are W3C standard ways to do so[13].
- **The ability to encode facts at any time.** As mentioned previously, triples don't necessarily have to be relationships between things but may also be additional context about an entity. Such a capability expands what's possible with respect to metadata representation.

12 https://www.w3.org/standards/semanticweb/
13 In fact, one well known semantic vocabulary is named "foaf" after "friend of a friend" (e.g., https://www.xml.com/pub/a/2004/02/04/foaf.html)

- **The ability to make *inferences* about data and complex relationships via *rule sets*.** For example, we might create rules such that when two facts are true (or two relationships exist), we may infer that a third fact or relationship exists without it being explicitly encoded in the data.

And it's the last point that is perhaps the most compelling. While it's intrinsically powerful to be able to represent various facts and relationships with triples (particularly when combined with documents), it's the ability to *reason* over these data representations – in ways that simply were not before possible – that is particularly exciting.

Again, the wider topic of semantics is quite expansive, even when scoped specifically to data integration, and is beyond the scope of this document. However, for the curious reader, we've included some links in the Bibliography at the end of the text.

The Role of the Multi-Model Database

Up to this point, we've covered quite a bit of ground regarding the data hub. We've gone over the basic principles, not only conceptually, but also by way of sample data. However, thus far in the text, the style of database that has been mentioned the most has been the relational database, largely as the root cause of our data integration challenges. We can, therefore, assume that a relational database is not the best choice of database for building a data hub. So then, what kind of database should we use?

If we recall from the Principles of a Data Hub section, we need to provide a number of capabilities. Being able to efficiently manage self-describing schema (without DBA intervention) suggests a NoSQL document store would be part of the equation. Similarly, the need to handle RDF calls for a triplestore[14], while the deep indexing and ad-hoc search capability suggest a full-text search engine. Finally, the data enrichment required for harmonization, alongside the concurrent read/write capability to support run-the-business functions strongly indicate a need for ACID transactions[15].

All of this suggests that we either need multiple databases to build a data hub (possible but leaving us with technical silos, separate indexes, etc.), or a single – albeit special – database.

The answer to the requirements is to use what is known as a *multi-model database*. As the name suggests, such a database allows multiple different data types to be stored and queried in their native formats from within the same context. In the case

14 https://en.wikipedia.org/wiki/Triplestore
15 https://en.wikipedia.org/wiki/ACID

of the data hub, this means that the database must seamlessly process documents/objects, RDF triples and full text, all within a single secure transactional platform.

MarkLogic Server is such a multi-model database, with the ability to natively store and index XML, JSON, multiple text formats, as well as various serialization formats of RDF triples. Additionally, as an enterprise-class DBMS, features such as ACID transactions and fine-grained security are built into the core of the product. Later in the text, in the chapter titled The Data Hub in Use, we will cover use cases implemented using MarkLogic. Beyond that, however, coverage of the full features of the MarkLogic Server is outside the scope of this text. We nonetheless provide reference material for MarkLogic in the Bibliography

Data Hub: A Simple Picture

Putting all of the previously mentioned concepts together, we can now draw a simple picture of what a data hub looks like from a simple data flow perspective.

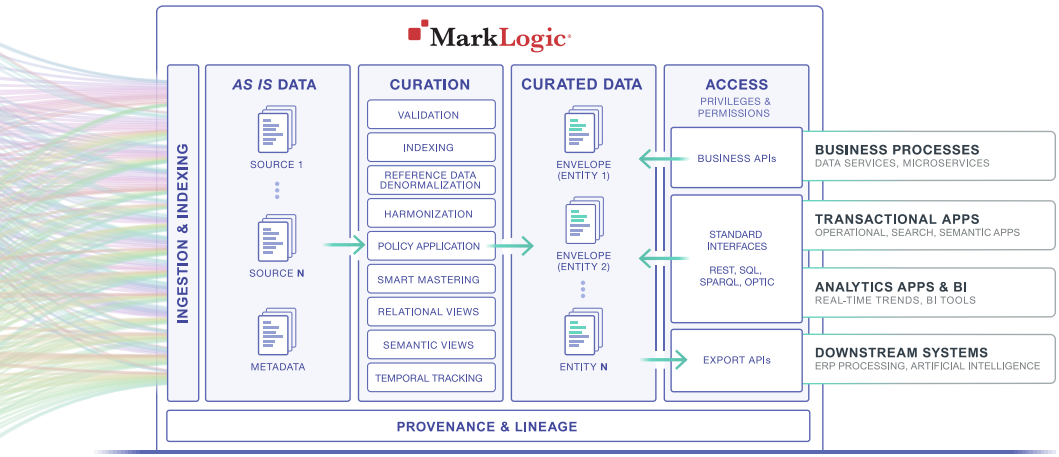


Figure 5: The MarkLogic Data Hub Platform

Looking at the diagram from left-to-right, we start with the sources of data. In this example, we show data coming from multiple sources such as relational databases, a message bus, as well as other content feeds (e.g., file systems, system interfaces, etc.). Data from these sources gets loaded *as is*, directly into the multi-model DBMS – in this case, MarkLogic. When data is loaded into the data hub, it is indexed as part of the ingestion process, as opposed to being a separate, optional step. It is this capability to index *as is* data during ingestion that contributes to allowing data to be queried and searched without extensive ETL processes.

After ingestion, data may then undergo any number of *curation* processes in support of the many uses of the data. The process of data harmonization, covered previously, is a foundational curation process associated with the data hub.

On the right side of the diagram, we show how the data hub provides multiple ways to access the data by way of data-centric APIs, that not only provide search and query capabilities against the data but also provide transactional capability to support cross-line-of-business operations on harmonized data. This is one of the key aspects that make it operational, allowing it to take part in critical run-the-business functions. Another key aspect of the pattern – specifically associated with the right side of the diagram – is that the APIs that are created are based on *incremental* usage of the integrated data, in line with the principle of harmonizing data on an as-needed basis.

Finally, the diagram calls out the cross-cutting aspects of tracking and reporting on data provenance and lineage, a critical capability made possible by performing incremental harmonization inside the database.

With this picture in mind, we may now turn our attention to fitting a data hub into existing enterprise architectures, and the associated positive impacts.

> **“…because the data hub allows for and encourages incremental implementations, nearly all data hubs start out with a smaller scope before growing into something enterprise-wide.”**

5

Fitting Into an Existing Architecture

Though most large enterprises share many common characteristics (as we've posited in The State of Enterprise Architecture section), they nonetheless differ from one another when we get into the details.

And, while products and tooling have and will continue to emerge around the pattern[16] (as has occurred with the data warehouse), the data hub is best classified as a pattern given its flexibility. Additionally, since a data hub covers more ground across the "run-the-business" and "observe-the-business" spectrum, and across more types of data and metadata than a data warehouse does, where it is placed within the enterprise may vary. This section is therefore dedicated to providing some guidance around where to implement a data hub inside an existing enterprise architecture.

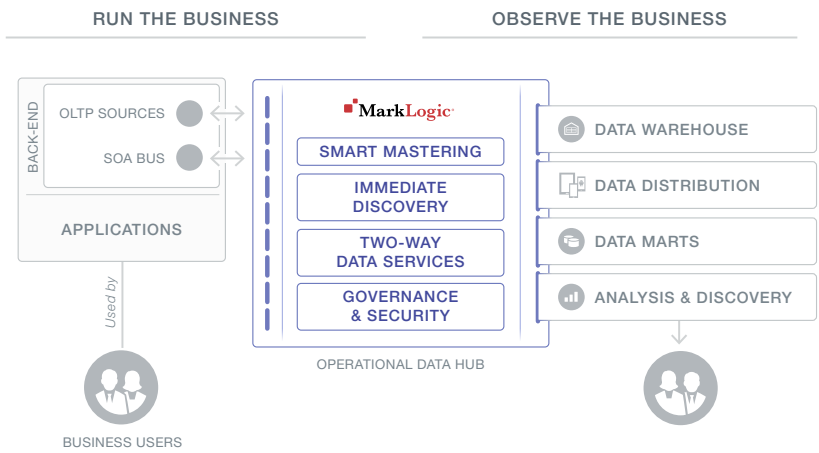


Figure 6: Simplified Enterprise Data Flow With the Data Hub

16 https://www.marklogic.com/product/marklogic-database-overview/database-features/data-hub-framework/

Starting Point: An Idealistic View

We'll start first with the idealistic view and associated implications, which includes some oversimplification. And, while the natural skeptic in all of us might be inclined to look warily on oversimplified models, distilling complicated topics into simpler models and pictures is often a good place to start. To that end, we will go back to our very simplified model of an enterprise and modify it.

Okay, that looks very different from Figure 1. What happened to the ETL? Where did the MDM go? Does this imply that the data hub is now the new center of my entire enterprise?

These and more are among the first questions asked when a diagram similar to Figure 6 is shown. As is typical of technology discussions, the answers to each are very much contextual, particularly for large enterprises. Nonetheless, the high-level view above is a good place to start and, as we'll see, doesn't necessarily imply sweeping statements about the entire enterprise, at least not at the outset. It may just as easily be referring to a smaller subset of an enterprise architecture as opposed to the enterprise as a whole. In fact, because the data hub allows for and encourages incremental implementations, nearly all data hubs start out with a smaller scope before growing into something enterprise-wide.

Let's look at ETL. In the above diagram, it is not represented. The reason for this is that, depending upon the state of the data pre-ingestion, no ETL would need to be done to force-fit things into any particular canonical model, since all source models may be staged *as is*. That said, an ETL tool may still be leveraged to connect to various sources or perhaps perform some very light format marshaling (e.g., RDBMS to CSV). However, in many scenarios, transformation beyond such simple payload translation would not be needed. In scenarios such as this, the above diagram would be largely accurate within the use case's context.

MDM is another area that is disrupted in our conceptual diagram. As a natural side effect of the as-needed harmonization process, data can also be matched and merged. In MarkLogic, it's called Smart Mastering™ due to the addition application of a probabilistic approach using AI and fuzzy matching, to complement rule-based capabilities. More importantly, we have moved this critical function inside the data layer and at the point of integration instead of creating yet another data silo in an attempt to store master copies of data. In our scenario, the "golden" copy of a business entity (e.g., customer) is formalized in an agile way, incrementally over time within the context of integrated data.

That said, along this journey, traditional MDM-based data sources may remain in place for periods of time, or may even evolve to use the data hub approach. Additionally, such incremental mastering for one business entity (e.g., trades) may coexist alongside traditional MDM for a different but related business entity (e.g., instruments). Again, going back to context, the diagram might be referring to the data strategy for but one part of an enterprise for a particular point in time.

And, that is a good segue into the last question above about context. When we draw the diagram, it is not necessarily the case that a data hub has to immediately be placed at the center of an enterprise for all data sources (though it could be). Many MarkLogic customers implement enterprise-wide data hubs, but most actually start out with a smaller scope against a defined business need.

This, of course, begs the question: when do we know when a data hub is appropriate for a particular place within a very complex enterprise? To answer that, let's look at a few simple guidelines:

1. The use case requires multiple different data sources (or input sources) to meet the requirements.
2. These data sources must be integrated in some way.
3. The integrated data will be used in multiple different ways, some of which may not even be directly related, across a variety of consumers (i.e., multiple output consumers and/or destinations).
4. The number and type of input sources and consumers are expected to change over time (i.e., change over time is the norm).
5. There are direct operational requirements against the integrated data (i.e., run-the-business use cases depend on the integrated data).
6. (Bonus) The operational requirements are such that more than read-only access against the data is required.

When represented pictorially, data flows that meet even the first three criteria above, take on a common look when represented at a high-level as follows:

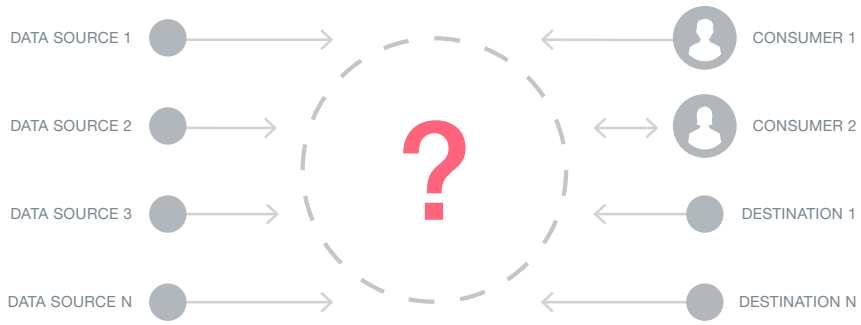


Figure 7: Simple "Bowtie" Diagram

We sometimes refer to the above picture as a "bowtie diagram." In fact, when you look at the diagrams in the subsequent section titled The Data Hub in Use, many of them resemble the bowtie diagram in some way with many inputs and many outputs.

Those high-level data hub diagrams will represent *solution* states, as opposed to pre-existing *problem* states. This is why our above bowtie diagram has a circle with a question mark in the middle of it, since that is the part of the picture that may vary the most depending on circumstances. But, even in these cases, we're able to draw some simplified pictures to illustrate certain "from state" scenarios as follows:

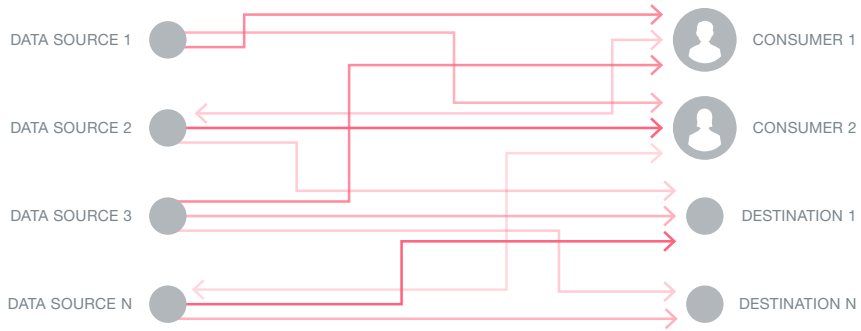


Figure 8: Point-to-point "Integration"

The scenario depicted in Figure 11 shows a number of applications exchanging information with each other via point-to-point information exchange. In this case, data is passed around as needed between applications that require information from other applications. This is most common between applications on the run-the-business side of the house.

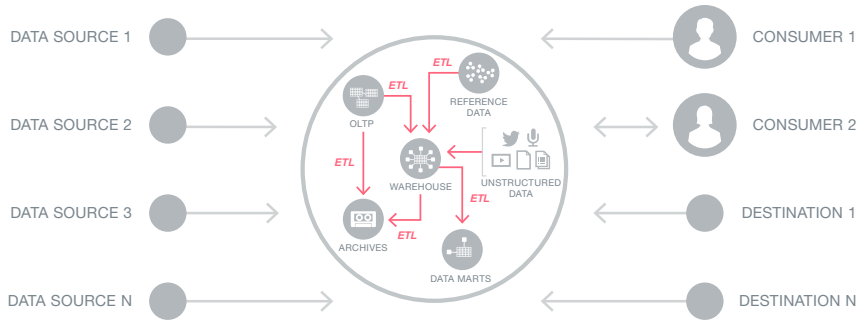


Figure 9: The "ETL" Quagmire

In Figure 9 we see something slightly different. In this scenario, data interchange is managed via complex ETL processes, as one might expect to see when integrating data in a data warehouse or across data marts. This is most common on the observe-the-business side of the house, and is what has given rise to the non-trivial enterprise data management functions first mentioned in The State of Enterprise Architecture section and depicted in Figure 1 .

Nearly all large enterprises contain a mix of both scenarios, often for overlapping use cases within various places throughout the enterprise, each accumulating their own technical debt. However, in every case where this many-in/many-out pattern exists, chances are a data hub implementation would make a positive contribution to not only reducing the associated technical debt, but also allowing for innovation not before possible.

Complementing Existing Patterns

Because the data hub deals with many inputs and many outputs, it invariably comes in contact with other components in existing architectures. Most enterprise data hub use cases are brownfield, not greenfield use cases. Usually, a data hub is complementary to other approaches. For example, it works very well alongside other data warehouses (as destination points), as well as and even newer tools like MuleSoft and Kafka for moving data into a data hub. Though a detailed look at every architectural scenario is beyond the scope of this text, we can explore the possibilities at a summary level within the context of integration.

The Data Hub and Data Warehouses/Data Marts

As we've mentioned previously, data warehouses and data marts solve observe-the-business problems. Additionally, *how* a data warehouse or mart might solve these problems is constrained in a number of ways.

Consider the following:

- **Categories of data.** Traditional data warehouses/marts are good at processing structured data that conforms to a preset model. This makes it difficult to include unstructured or semi-structured data into the observe-the-business space. In addition, multiple models and complex models are difficult to represent in a data warehouse or mart.
- **Modeling techniques and schema changes.** Data warehouses and data marts typically rely on a technique known as the star schema[17]. This modeling technique limits analysis to being mostly quantitative in nature. In addition, due to their relational nature, star schemas are brittle and resistant to change. In an extreme case, adding a single column to a star schema fact table[18] might force the entire data set to be reloaded.
- **Data quality and data governance.** Because data warehouses are limited in terms of what they can store at a point in time, they cannot easily store ad-hoc metadata pertaining to such things as provenance and data lineage, which then must be handled outside the database (e.g., in ETL layers).

When used in conjunction with a data hub, however, a properly-scoped data warehouse or mart can be much more effective as a downstream consumer of data hub data.

For instance, with the ability to handle multiple different categories of data (structured, unstructured, etc.), having multiple modeling techniques available (e.g., documents, triples), all the while being able to handle multiple different schemas simultaneously, the data hub removes the ETL dependency from the observe-the-business function. In addition, by providing mechanisms that allow for ad-hoc metadata capture (e.g., the envelope pattern), data provenance is kept close to the integrated data, improving data quality overall.

> *"In an extreme case, adding a single column to a star schema fact table in a relational database might result in a complete change in data granularity, forcing the entire data set to be reloaded."*

17 https://en.wikipedia.org/wiki/Star_schema
18 https://en.wikipedia.org/wiki/Fact_table

Given these advantages, one might naturally ask if a data hub simply replaces a data warehouse. As a rule of thumb, a data hub is **not** a drop-in upgrade/replacement for a data warehouse or data mart. It is a different solution meant to solve different problems.

That said, there are many cases where a data warehouse/mart was built in an attempt to solve problems it wasn't ideally suited to solve. For many years, after all, these were the only available observe-the-business data integration patterns. In these cases, where data warehouses and marts were built in an attempt to provide many-in/many-out integration points, with business requirements that change frequently, a data hub implementation is a much better replacement.

However, there are also cases where a data warehouse (or mart) is either the right choice or at least *part of* the right choice, in which case the warehouse/mart simply stays in place and is complemented by a data hub, reducing the overhead associated with ETL, and providing discovery help in other areas. Often a data hub will feed a downstream data warehouse or data mart. In all cases, however, the introduction of the data hub will serve to reduce the friction imposed by a traditional ETL + warehouse approach to data integration by simply removing the need to have to model all of the data perfectly before getting value from it.

What this means from the perspective of an entire enterprise, is that the proper implementation of the data hub will dramatically *reduce* the overall number of instances of data warehouses and data marts, as well as *rationalize* what is expected from them.

The Data Hub and Service-Based Architectures

On the run-the-business side of the house, data integration mostly involves application-to-application interchange, either via direct API communication between the applications, or via some kind of central message bus or service bus. In either case, there isn't much in the way of true data integration or harmonization as these types of integrations are not particularly data-centric at all, as they mostly focus on the functions performed between the cooperating applications, leaving the data persistence side of things to the applications themselves. In other words, what happens to the data behind the cooperating applications is considered mostly "black boxed" and secondary.

With a data hub in place, true data-centric integration is now available to the run-the-business side of the house, allowing real-time applications to be brought to the data in many cases, as opposed to needing to move the data around to the

applications whenever operational integration needs to occur. When we consider how much technical debt accrues as a result of unnecessary data duplication within the enterprise – particularly on the operational side – the impact of technical debt reduction that's possible with a data hub is significant.

And technical debt is not the only consideration. When a data hub enables real-time, data-centric integration, it becomes possible to support cross-line-of-business operations that might otherwise not have a home. For instance, a global investment banking customer needed to keep track of a particular regulation regarding OTC derivative trades that could only reasonably be gleaned after a number of trading desk data was integrated. At that point, any trades that needed remediation would need to be tagged and tracked for further processing.

In this example, the best first place to tag these trades was in the data hub, recording the fact that they needed certain types of upstream and downstream remediation, all of it, however, based on an integrated context. In other words, the operational workflow *began in the middle*, so to speak, even though the trades originated elsewhere. Only with a data-centric integration pattern with operational (i.e., run-the-business) capabilities was this possible.

The Virtues of Incremental Implementation

Perhaps the most significant advantage of implementing a data hub is the ability to execute a long-term data strategy *incrementally*.

Integration that depends on prerequisite modeling and data copying is not agile. Every time a business change forces a change to a schema, a non-trivial amount of work must be done. Tables must change, ETL code has to be rewritten, and data has to be reloaded. For these reasons, data modeling in the RDBMS world is a somewhat paranoid activity, forcing modelers to account for every possible scenario. This makes many large-scale data integration initiatives either very slow, very brittle, or both.

With a data hub in place, incremental implementation is one of its guiding principles. No longer do modelers and implementers have to worry about coming up with the perfect canonical model before everything is understood. It is now perfectly acceptable (and encouraged) to create parts of a canonical model on an *as-needed* basis, all the while preserving all incoming original models, without having to decide which data is kept and which data has to be "thrown away." With this new level of data integration freedom, incremental technology milestones can be aligned neatly to incremental business outcomes, improving time to delivery, ensuring data quality, and reducing overall project risk.

“...despite the differences across industry and business outcomes, the goals and approaches had a great deal in common, even for seemingly dissimilar industries.”

6

The Data Hub in Use

We mentioned previously that a multi-model database is ideally suited to power a data hub, citing requirements such as self-describing data payloads (e.g., XML, JSON), RDF triples, and integrated search, preferably in a single product so as to avoid technical silos. As this section is dedicated to concrete real-world examples, the descriptions are all product-specific, discussing data hub implementations specifically using MarkLogic's Data Hub Platform, which is powered by MarkLogic Server, a leading multi-model database.

Earlier, we also said that lasting technology patterns are often discovered as opposed to contrived. This happens over time as organizations learn from each other and best practices emerge. To illustrate this point, our case study examples cover a variety of industries and enterprises that both contributed to and reinforced core concepts in their own unique ways. But, despite their uniqueness, they all have one thing in common—they are all large organizations with complex problems with an urgent need to simplify data integration.

The Data Hub in Financial Services

Background

The Global Financial Crisis (GFC) of 2008 created an existential crisis for not only capital markets and investment banking, but the world's economy as a whole. For the first time since the Great Depression, the very fabric of the global economy was threatened. Iconic institutions either went out of business (Lehman Brothers) or were forced into acquisition (Bear Stearns).

In the process, Wall Street took a reputational beating as governments and regulators stepped in to first try to make sense of the mess, then limit the damage, and finally assert themselves via guidelines and legislation to protect against such events from being repeated. In these ways, there were parallels to the Great Depression. Yet in other ways there were challenges unique to the 21st century. Because unlike the Great Depression, the role of technology, specifically modern ***data*** challenges, loomed large in the discussions.

"What's our exposure to Lehman Brothers?" was a question often asked by frantic participants in global financial markets at the onset of the GFC. But despite technology budgets that in some cases were measured in billions of dollars annually, the answer wasn't always clear. It seemed at the time that financial engineering had outpaced technology engineering, rendering full fiscal transparency next to impossible.

A Top Five Global Bank Takes a Leap

In 2009, a top five global bank had a $70 trillion derivatives portfolio and one of the largest holdings of credit default swap (CDS)[19] positions. By this time, regulators had made it clear that transparent and holistic views of banks' exposures would be the new reality for large banks, and credit default swaps were but one of many types of derivatives instruments within scope.

For the technologists who had to make sense of the various sources of data, it was clear that the traditional approaches to data integration would not meet their intended goals. So they charted a different course, using MarkLogic as the underlying technology, and (what was then) a different approach to data integration.

A summary of how the challenges were addressed follows in Table 3.

19 https://www.investopedia.com/terms/c/creditdefaultswap.asp

Table 3: Top Five Global Bank–Data Hub for Derivative Trading

Business goal	Consolidated view for all "over-the-counter" (OTC) derivative trades to support cross-asset-class discovery and post-trade processing.
Challenges	Multiple trade execution systems with varying asset classes and complex models to support them presented a challenge to creating an integrated model representative of all trades. Non-trivial developer and DBA intervention to translate the complex models of all derivative types and/or variations to relational structures. Inability to analyze their total market exposure and launch new products in an aggressive marketplace, causing them to be at risk and lag behind competitors. A need to **incrementally** include new data sources with minimal re-work, creating significant pressure on the data modelers to "get it right the first time."
Input sources	More than 20 different trade execution systems across various OTC derivative asset classes.
Consumers and outputs	Various consumers to support post-trade processing (matching, clearing, allocations, settlement, etc.), internal risk, as well as regulatory and compliance reporting.
Result	A single operational data hub for OTC derivative trades eliminating the need to create multiple bespoke data marts and complex ETL. The first month of production, the system had scaled to 1,600 requests/second supporting discovery-based and operational workloads. Labor savings alone were estimated to be US $4M over the first three years.

> *"It was clear that the traditional approaches to data integration would not meet their intended goals."*

The following is a high-level pictorial representation:

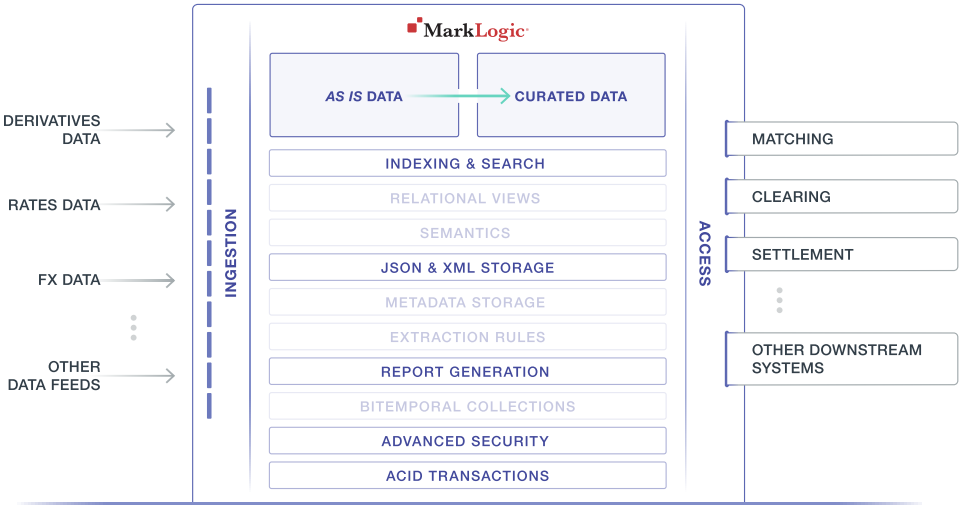


Figure 10: Top Five Global Bank–Data Hub for Derivative Trades

Like many other data hub implementations, the diagram depicts a many-in/many-out data flow that is characteristic of the pattern.

Other Data Hub Implementations in Financial Services

Since this early mission-critical implementation in investment banking, MarkLogic has been the underpinning of many other data hub implementations across the broader financial services industry, addressing a number of use case categories including:

- Other post-trade processing use cases
- Operationalizing regulatory reporting and compliance
- Customer 360 for consumer banking
- Automation of “know your customer” (KYC) workflows
- Investment research
- Investment management

Today, there are data hub implementations at investment banks, commercial banks, inter-dealer brokers, information providers, and FinTech firms, with a list of use cases and customers that continues to grow.

The Data Hub in Media and Entertainment

Background

Long before global finance had its 21st-century existential crisis, publishers had their moment. At the turn of the millennium, many were faced with the stark choice of "digitize or disappear" as information delivery morphed from being based on distinct publication types (e.g., books, magazines) to being based on digitization of content across multiple media types. In other words, product silos were no longer considered a scalable model. During this time, many in the information delivery business reinvented themselves by managing and repurposing their content more fluidly across a variety of media types, by implementing data hub-centric architectures.

Over the past 10 years, the entertainment industry has similarly been upended, as online and digital delivery of content has radically changed how consumers get their entertainment content. Streaming and download services in the late 2000s contributed to an explosion of options for delivery. The number of distribution partners went from 100s to 1000s almost overnight, while the number of formats and versions of the content also exploded as release windows narrowed or in some cases disappeared altogether. As a result, media organizations have had to reinvent nearly every aspect of how they manage and deliver their content.

A Major Studio Leads the Way

Starting in the early 2000s, a major Hollywood studio took the plunge to reinvent their content distribution to take advantage of the new digital and online distribution opportunities. The first step was the digitization of the assets themselves. This resulted in every asset in one of the largest libraries in Hollywood being instantly accessible, but also highlighted another problem: it was difficult to find the assets. The information that was required to select assets was much more than just the technical metadata of the digital library. Studio personnel looking to find assets and teams creating distribution feeds needed to access descriptive metadata as well as title, product, and rights data.

All of this data resided in separate systems, and as a result, internal teams looking to re-use assets would have to call up experts with access to these separate systems and enlist them to help find the right clips. And teams creating automated feeds had to connect to these different data sources and execute custom logic to select the correct assets. Creating a new feed would take 2-3 weeks per feed, on top of additional time to update existing feeds as specifications changed.

To address this critical issue of access to the asset data, the studio turned to MarkLogic to create a data hub for digital distribution. This system was able to integrate critical data from multiple source systems and then make that data available to internal users, teams creating distribution and other feeds, and even external partners to enable self-service. This system greatly reduced the manual and technical effort needed to find the right assets and get them to partners. Making this data easily accessible is also a key factor in the studio's success in taking advantage of the many new opportunities that the new online and digital entertainment marketplace has created. The following table summarizes the use case.

Table 4: Data Hub for Media – Solutions Summary

Business goal	Create an up-to-date map of digital assets globally, with information from multiple disparate systems, which allows end-users to find and leverage content in real-time.
Challenges	Every time assets were needed, searches required reaching out to multiple source-system experts. The approach was not scalable to meet real-time needs.
Input sources	Data on the entire asset catalog of the studio. About six million records including technical metadata from multiple digital asset systems, descriptive metadata and title data from multiple systems, rights and product information from multiple systems.
Consumers and outputs	The system needed to provide a robust set of APIs that multiple teams could access to create both end-user functionality and custom applications. These include feeds to distribution partners, a custom application for partners to select content, integration with distribution partner systems, and integration with custom in-house asset search applications.
Result	A single data hub that powers every aspect of the studio's digital supply chain. Using the data hub pattern, the first application was released in just eight weeks. The final system has led to saving "hundreds of hours" of individuals' time to find assets and helped reduce the overall cost of distribution by 85 percent.

This system shares the multiple input and access to data for multiple downstream systems of other data hub solutions. It also directly powers a critical application in the partner portal.

Following is a high-level pictorial representation:

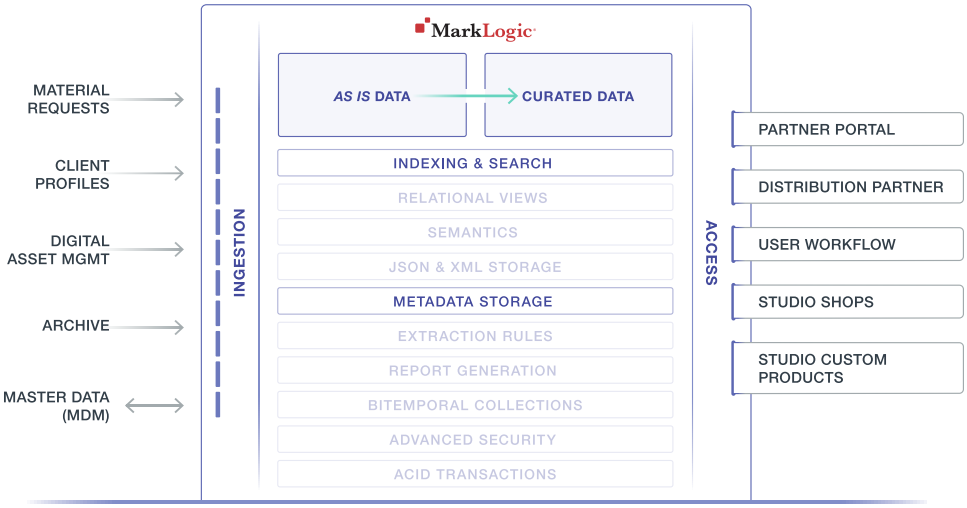


Figure 11: Entertainment Studio–Data Hub for Digital Distribution

Other Data Hub Implementations in Media

This studio was at the vanguard of taking a new approach to managing the metadata critical to digital distribution. Other entertainment organizations have used the data hub pattern to:

- Create the platform to power the online presence for the host country Olympics including distribution feeds to partners and custom views of athletes
- Power a popular online application that leverages not just the asset metadata, but data describing the characters and eras of the show as well as the fans' interactions with the content
- Create an authoritative view of descriptive metadata for shows across a broadcaster, including detailed tagging of scene and character interaction to enable better re-use and content placement

Today, data hub implementations at media companies are helping customers stay ahead of the many more changes to the industry, most notably connecting directly to customers and getting content to new platforms and markets at any time. The changes in how people consume and pay for content are continuing to evolve, and MarkLogic data hub solutions are allowing our customers to stay ahead of the changes by making sure critical data is integrated and available.

The Data Hub in Defense and Intelligence

Background

In the defense and intelligence community, existential crises go far beyond the confines of industry. Data and information are at the heart of every tactical and strategic activity from local policing to international defense. As 9/11 and other tragedies have demonstrated, timely, fluid, and safe sharing of information can sometimes be the difference between a successful outcome and tragedy.

"Need to know" meets "need it now"

One particular elite military group had an ongoing struggle to move from a 'need to know' mindset to a 'need to share now' mindset. Adapting this new mindset was particularly important due to the changing nature of their missions, their focus on irregular warfare, their global area of responsibility, and their need for interoperability with the Department of Defense, Intelligence Community, and international partners.

The challenge they faced was in building a command-wide knowledge management (KM) and information sharing (IS) system for an increasingly diverse dataset that would be consumed by a wide variety of programs and people. It was the many-in/many-out problem on a global scale.

To accomplish their goals, they needed a data hub to serve as a centralized system architecture that would integrate various types of information from multiple sources into a consistent, well-defined view. This was something that simply could not be solved by their legacy technology stack that relied on an RDBMS as a primary integration point. The following table summarizes how a data hub met the challenge.

Following is a high-level picture of the architecture:

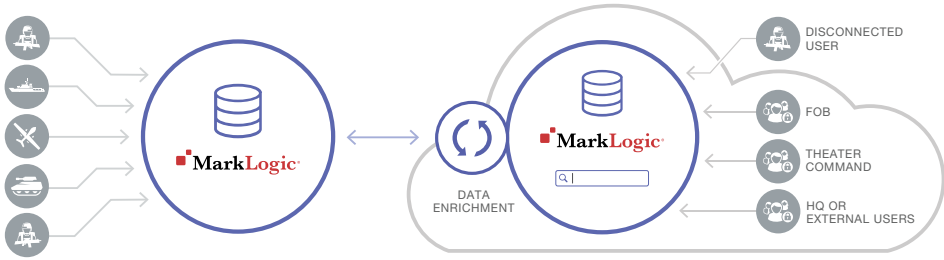


Figure 12: Data Hub for Defense–High-level Architecture

Table 5: Data Hub for Defense – Summary of Capabilities

Goal	Distributed information that harmonized various types of intelligence sources (SIGINT, HUMINT[20], full-motion video, etc.) into a consolidated view. Sharing had to be as real-time as possible and stretch across a distributed mesh of global networks.
Challenges	Unable to handle a variety of sources ranging from structured database data in legacy systems to multi-media data from captured mobile devices. Inability to work across a variety of networks securely, ranging from trusted, high-bandwidth to untrusted, low-bandwidth with intermittent connectivity. Operationalize data in a multi-master environment.
Input sources	Hundreds of different sources of data and metadata, from text to video.
Consumers and outputs	Various defense and intelligence agencies in the US as well as international partners.
Result	A data hub that managed over 100 million data and metadata documents after the first go-live. Query results that were 59x faster than the previous RDBMS solution, covering more data types, with the ability to distribute data globally in a multi-master environment.

Other Public Sector Implementations

Within the Public Sector, the data hub is implemented in more places than defense and intelligence, covering a wide variety of use cases. Non-government organizations (NGOs) across the globe have implemented data hub architectures, as well as local government health and human services organizations, who have even extended their data hub architectures to perform agile on-demand master data management (MDM) of their population data.

20 SIGINT: Signals intelligence, HUMINT: Human Intelligence

The Data Hub in Healthcare and Life Sciences

Background

Healthcare in the United States went through its own seismic change, starting in 2010 with the passing of the Affordable Care Act (ACA). And although discussion of whether it started or addressed an existential crisis to the US healthcare industry would likely be muddied by political debate, what is generally agreed upon is that the passing of the ACA was a profoundly impacting event in the industry. For those in the technology profession, particularly those responsible for data integration, the ACA presented significant challenges and opportunities. It's no surprise then that the data hub pattern would have its moment, and this time in both government and the private sector concurrently.

An Immovable Congressional Deadline

The ACA that was signed into law in 2010 represented an attempt to not only address the soaring cost of healthcare in the US but also to modernize the way in which Americans obtain healthcare insurance by creating a publicly accessible digital marketplace. A data scientist from the Centers for Medicare & Medicaid Services (CMS) called the creation of a digital, public marketplace where people could apply for, be qualified for, and then shop for and purchase health insurance, in many cases for the first time, "an impossible data problem." The ability to integrate and operationalize data from all private insurers, virtually every federal agency including the IRS, and the state exchanges which rely on the same "back end" as the Federal marketplace, created a data integration challenge – that like many other large organizations, the government first tried to solve using legacy relational database technology and legacy methods of data integration.

As the deadline approached (a very public deadline that no one could afford to push back), those at CMS tasked with creating the backbone of the ACA realized that legacy technology and methods were not going to work. The variety of the data that needed to be integrated, the operational demands of the enrollment period, the unknown requirements of something that had never been done before, and the security demands of data that included PHI, PII[21] and protected data from multiple Federal agencies, demanded they find a better way and find it fast.

21 PHI: Protected health information, PII: Personally identifiable information

After nearly two years of trying to create an all-encompassing data model in an RDBMS, data architects eventually realized that the creation of a flexible data hub architecture represented the only realistic way in which to achieve the real-time data integration goals.

Table 6: HealthCare.gov – Solution Summary

Goal	Create a user-friendly public marketplace where consumers may shop for health insurance via an e-commerce style website, all the while linking the application process to all necessary agencies and stakeholders in real-time.
Challenges	Complex modeling of content that was in multiple, different formats from multiple federal agencies, state agencies, and insurers. Initial work had begun with assumptions of ETL and RDBMS technology for handling the integration challenges. When that proved not feasible, work had to begin anew, shortening the deadline further still. The marketplace needed to be a real-time operational system, not only interacting with end-users but choreographing workflows across multiple agency systems (e.g., IRS, CIS, DHS, etc.).
Input sources	Federal income and eligibility systems Health insurers End users
Consumers and outputs	End users State exchanges Health insurers Federal income and eligibility systems
Result	A data hub implementation to support the healthcare marketplace, as well as one to support the data services hub between multiple agency systems. Overall, the implementation met the congressional deadline for delivery so millions of Americans could obtain insurance via the marketplace.

The following picture provides a high-level view of the interactions between the various systems as well as the two hubs created to support the architecture.

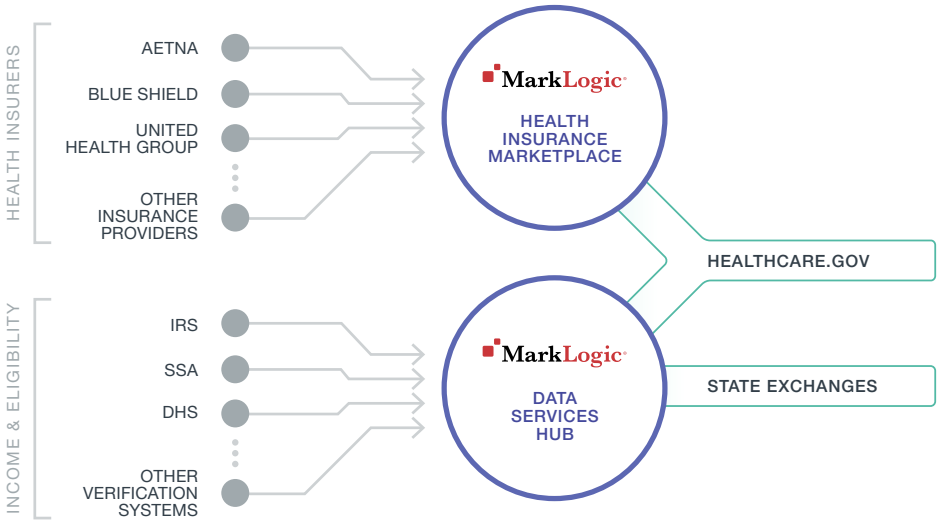


Figure 13: HealthCare.gov–Marketplace Data Hub and Data Services Hub

Other Healthcare and Life Sciences Implementations

Since then, MarkLogic has implemented data hub architectures for commercial healthcare payers, as well as for customers in Life Sciences and Pharmaceuticals customers supporting a wide range of use cases including customer/patient 360, Real World Evidence (RWE), drug supply chain genealogy, life sciences R&D, clinically integrated networks, state Medicaid/Medicare, and mainframe data migration.

The Data Hub in Insurance

Background

Digital transformation has blown up the global map of retail (Amazon and Alibaba), hospitality (Airbnb) and transportation (Uber and Lyft). The venture capital (VC) money that fuels these disruptions has now turned its attention to the insurance industry, funding hundreds of digital "insuretechs" to disrupt an industry that still relies heavily on paper processes and face-to-face transactions.

To the credit of the incumbents, the reaction of the industry giants has been swift and broad-reaching. Many large insurers are exploring new technologies, starting innovation spin-offs, their own VC funds, or simply buying best-in-breed insuretechs to fuel innovation. Mentions of "technology" and "digital" on insurance company earnings calls almost doubled between 2016 and 2017[22].

In the process, insurers are discovering the significant gaps they need to fill in order to leverage their own massive stores of legacy data. The promise of Artificial Intelligence (AI), machine learning, and Natural Language Processing (NLP) to automate slow, and mistake-prone manual processes, and increase customer acquisition and retention by offering a frictionless digital experience, all rely on being able to fluidly integrate, understand and operationalize all of this data.

Improving Customer Data to Improve the Customer Experience

So, what happens when a centuries-old industry has to reinvent itself? In one case, a Fortune 500 Property & Casualty (P&C) insurer decided to take its first steps in digitizing its policy processes and modernizing its customer experience. Like many of its peers, it had accumulated a significant amount of customer data over the years. The data processed came in all shapes and sizes:

- CSV-formatted customer feeds
- XML-formatted billing records
- JSON-formatted policy records
- PDF-formatted claims

All were stored across multiple source systems, ranging from mainframe to relational.

22 https://www.cbinsights.com/research/six-big-themes-in-2017/

To deliver a modern customer experience, the P&C firm needed to have a unified view of its customers, including policy details, billing information, as well as any other customer data captured through various channels and touch points. As expected, the firm originally relied on massive ETL processes to match and merge customer data from multiple sources, including a legacy Master Data Management (MDM) platform. This platform – a multi-million-dollar investment – required as much as 18 months of lead time to implement any changes that called for the capture of information for which it was not initially designed.

In other words, the firm dealt with the typical rigid-schema problem of RDBMS-based solutions. These limitations made it next to impossible for the insurer to innovate its way into an increasingly digital marketplace. It recognized that if it were to stay relevant within the insurance digital revolution, the MDM system needed to be retired, and replaced with a modern infrastructure designed to power multiple next-generation applications. Thus, a data hub with Smart Mastering™ became the path forward. The solution is summarized below:

Table 7: Data Hub for Insurance – Summary

Goal	Create a single integrated view of the customer to serve multiple applications and use cases.
Challenges	Content in multiple formats from multiple systems Incomplete view of the customer, with data in various systems often in overlapping and/or conflicting ways 18-month lead-time for changes not scalable
Input sources	Multiple systems of record (CRM, agent, policy, claim, billing, quote) Multiple formats (CSV, XML, JSON, PDF) and models Combination of structured and unstructured data
Consumers and outputs	Online customer management Downstream billing systems Multiple customer-facing web systems
Result	A modern data hub implementation providing a 360-degree view of their customers. Development time was 5x faster, and ability to deal with model changes was 10x faster, across a wider variety of data. They are now able to master data incrementally based on business outcomes.

Following is the high-level pictorial:

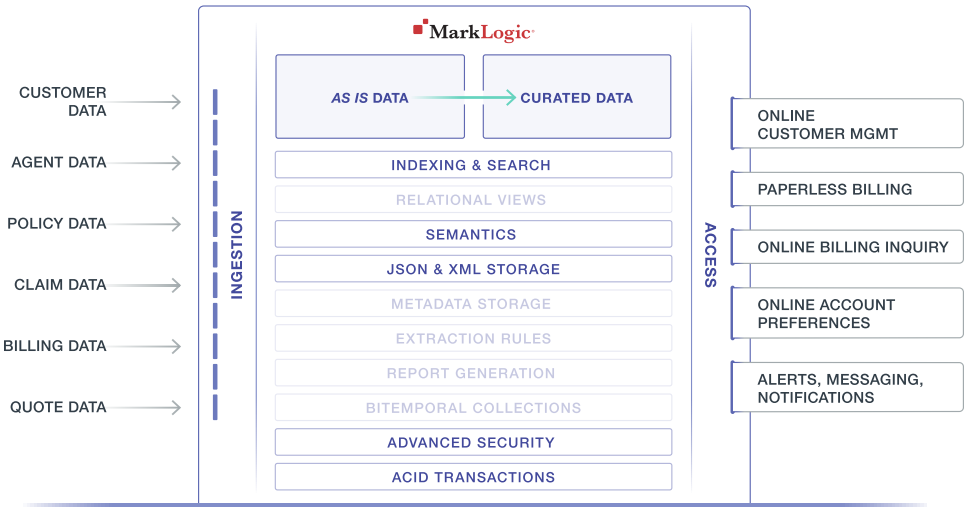


Figure 14: Data Hub for Insurance–High-level diagram

Other Insurance Implementations

Because of the digital transformation facing the insurance industry, the data hub has proven central to its ongoing technology transformation. As of the time of this writing, data hub implementations are in production in the insurance and reinsurance industries, across a variety of customers, on four different continents, supporting a variety of use cases including customer 360, agent systems of record, claims fraud detection, and advanced content distribution.

Other Industries

Whether it's existential crises, opportunities for innovation, or a mix of each, every industry is impacted by the challenges and promises of data integration. In every case, the data hub can play a role. Data hubs for manufacturing, transportation, and logistics, in the energy sector, and in other industries are moving into production at some of the largest organizations in their respective verticals.

> **“In the rush to use new (yet untested) technologies, critical aspects of solving data integration issues were ignored—security, governance, operational capabilities…”**

7

A Look Ahead

Thus far, a good deal of the data hub context in this text has been around rationalizing technical debt accumulated within large enterprises over the last two to three decades. In other words, a lot of the time has been spent describing how to remedy the sins of the past. Yet, if technology has taught us anything, it's that innovation is always right around the corner promising ever-greener pastures. As we've also learned, these promises sometimes lead to their own potential for unintended consequences.

The big data hype cycle had its moment, promising to unlock the latent power trapped inside enterprise data by merely loading huge volumes of it into giant data lakes. That approach is largely a failure. It's not that the data lakes didn't address some issues, it's just that in the rush to use new (yet untested) technologies, critical aspects of solving data integration issues were ignored – security, governance, operational capabilities – creating many new problems (which some of the same vendors happily used as an opportunity to sell more stuff).

As the Hadoop hype cycle fades, there are nonetheless other significant industry shifts that some may look toward as the potential "next big thing" to help make some of their data integration pains go away. Some of these trends are well-established and have been validated within certain contexts (e.g., cloud), while others are still finding their footing (e.g., blockchain). All are tempting to look at when confronted with difficult problems. After all, it's much more fun to look at new things than to toil with the same old seemingly unsolvable problems.

When it comes to trends in general – be they established, unproven, or somewhere in between – they all have the potential for misapplication and/or distraction from core challenges such as those associated with data integration. While it can be difficult to predict how unintended consequences might manifest as a result of technical innovation, there is one piece of advice we offer when considering any new technology innovation.

> *"…if a proactive data integration strategy does not accompany a cloud strategy, a move to the cloud can accelerate some of the many problems that contribute to data silo creation in the first place."*

Always ask the question "**What's the impact to the data?**"

Such a question may sound trite or simply be considered part of a larger checklist of items to consider. The reality is that data has always been the most important topic of consideration for IT. In fact, turning data into meaningful information – and action – may just be the entire reason IT exists. Data and its use is IT's *raison d'être*.

From this lens, we should expect that any IT innovation must have a net positive impact on the overall data mission. Its benefit must outweigh any technical debt it creates. This is why the data hub is such an important architectural construct. As a key component of enterprise-wide data interchange, it provides a stable platform to protect against technical debt while maximizing the potential for innovation. To demonstrate this point, let's consider three recent data-related innovations and assess the role of a data hub in each case.

Cloud

There is no denying the impact of cloud on IT innovation. It has nothing short of revolutionized the way IT resources are consumed and managed. While its existence can make some "traditional" concerns obsolete (e.g. hardware procurement lead time), it makes others even more critical. In other words, while cloud technologies remove a lot of friction associated with provisioning

infrastructure and services, adopting them does not make challenges associated with data integration simply "disappear into the cloud."

If a proactive data integration strategy does not accompany a cloud strategy, a move to the cloud can accelerate some of the many problems that contribute to data silo creation in the first place. In other words, faster application creation can also translate into faster silo creation. This translates to more data integration headaches. The advice we offer then is very straightforward: When adopting a cloud strategy, a data hub should be part of that strategy as well.

Blockchain

Though some of its initial hype has recently settled a bit, the use of blockchain in the cryptocurrency space has demonstrated that is has some staying power and perhaps even untapped potential. Like other newer and promising technologies, it is accompanied by the capacity to monopolize focus, leaving open the possibility for less-exciting, yet critical, functions to be ignored by blockchain implementers, resulting in unintended consequences. For example, security is one area that should be top of mind, thanks in no large part to some well-publicized digital currency breaches. As these breaches painfully demonstrated, a blockchain implementation is only as secure as its weakest link. Since all blockchain implementations at some point invariably interact with non-blockchain databases, when those non-blockchain databases are not secure or ACID compliant, the results can be disastrous.

In addition to security, there are other more-mundane-sounding considerations that may also become barriers to blockchain success. Take for instance the promise of blockchain to remove certain manual, data-intensive friction points by way of automation. For example, in investment banking, settlement of derivative trades is one potential area where a blockchain may speed the entire process by automating settlement between financial institutions. While automated settlement may result in the elimination of certain related tasks, there would still remain non-trivial data-preparation processes that would need to occur outside of a blockchain – tasks that would not go away simply because a blockchain is present. In these cases, the dependent data-related tasks would no longer have certain built-in governors associated with a slower settlement process, thus requiring the pre- and post-settlement data lifecycle to be much more efficient and accurate than it is today. In such a scenario, a data hub, with the ability to harmonize and govern data from all parts of the enterprise, would be a key enabling feature of any blockchain strategy.

Machine Learning (ML) and Artificial Intelligence (AI)

Of all of the recent tech trends, it is machine learning and AI that has some of the most investment and excitement around it. After all, the goals are nothing short of creating tireless, autonomous, super-intelligent machines capable of solving all sorts of intractable data-related problems on their own. There is almost a temptation to "wait things out" until AI can just "solve the problem". We'll then be able to just reap the business benefits, happily freed from the mundane (while our robot assistants fetch us beverages). However, it is in these particular domains of ML and AI (minus the beverage-fetching) that a data hub plays a critically important role.

Any good data scientist knows that good learners require good data — and lots of it. Data scientists also know that they spend most of their time trying to get hold of data, as well as trying to get it in suitable shape for their algorithms to work their respective magic. Even then, should the results of the algorithms be "off" for some reason (e.g. unintended bias), then it is incumbent upon the data scientist to go back to the data, understand where it came from and from which context, and adjust things accordingly. In such cases, a promising AI project can devolve into a series of mundane data wrangling tasks. In other words, the data scientist (i.e. a human being) is very much in the middle of the machine learning and AI processes, oftentimes doing quite unglamorous things. And, that is just the person who tries to domesticate the wild data algorithms of the AI jungle. What about the non-data-scientist humans who wish to use the results of these algorithms? How might they put the resultant findings into a broader business context and combine them with the more mundane human-centric findings to make informed business decisions?

This is where a data hub comes into play. Removing the friction associated with wrangling data for use by ML/AI algorithms, serving up governed and quality data, providing a lens into data provenance and lineage, chunking machine learning findings into broader context; these are all critically important data-centric functions for successful use of ML and AI. They just so happen to be the very things that a data hub is good at. In other words, in order to get the most benefit from your enterprise ML and AI programs, you need a data hub.

Cloud, blockchain, machine learning and AI: these are just three examples of the more recent innovations that point to the need for a fundamental shift in how enterprise data is managed, lest the promised benefits of these innovations not be fully realized. In this regard, the data hub represents a foundational component to any enterprise data strategy.

Conclusion

Despite the increasingly fast pace of technology advancements, sometimes the inertia of “accepted wisdom” can be a surprisingly limiting force. For years, the accepted wisdom of ETL and data warehouses, the dogma of run-the-business and observe-the-business separation, and the assumption that you must have a rigid schema designed up-front before any data can be loaded to a database have all resulted in the accumulation of significant technical debt.

Today, there is a new architecture that has emerged as a missing piece of the architectural puzzle – the Data Hub. It is a proven approach, refined in multiple mission-critical settings across a variety of industries. And, it arrives not a moment too soon.

The hype cycle of early big data technologies has faded, but many of the challenges remain and are getting more complex as organizations look to migrate to the cloud. We hope that all enterprise architects feel empowered to address those challenges head-on by rethinking how data is integrated and managed using an approach that brings simplicity to their enterprise – not more complexity. The success stories of the data hub are only just beginning to be written.

“We hope that all enterprise architects feel empowered to rethink how data is integrated and managed using an approach that brings simplicity to their enterprise—not more complexity.”

Appendix

A Data Hub Template for Enterprise Architects

Since technology patterns are typically catalogued in a formal way, we have provided a templated description of the data hub here. Thinking of the data hub as an enterprise architecture (EA) pattern, we chose TOGAF[23] as the inspiration for our template (albeit with some modifications).

The information that follows is consistent with the preceding material in the text.

General Information

Granularity	Blueprint
Short Description	A cross-functional, persistent data interchange typically in support of an enterprise data integration strategy.
Long Description	An authoritative multi-model data store and interchange, where data from multiple sources is curated as-needed, to support multiple cross-functional analytical and transactional needs.

Problem Context

The existence of business activities that require data from multiple lines of business to be integrated into a **consolidated** representation. The integrated data is necessary to support cross line-of-business **discovery** and **operations**. Requirements of this integrated data include:

- **Adaptability.** Changes in source data models should have little to no impact on the integration activities. For instance, the inclusion of a new data attribute in a source feed should not, as a rule, require the intervention of IT to simply stage the data into the integrated view.

23 http://www.opengroup.org/subjectareas/enterprise/togaf

- **Real-time processing.** Updates from source systems are required to be processed in real-time.
- **Lineage.** Lineage of and metadata about how the data is integrated must be queryable in an ad-hoc fashion, alongside the data itself.
- **Governance.** Governance of the system must be agile across a time continuum. Examples include:
  - Governance rules around data conformance that may change on the state of the data within a workflow over a time period.
  - Governance rules that change as business rules change over time, requiring not only adaptability as they change, but the preservation of historical governance rules.
- **Security.** Data security capability that not only accounts for individual line-of-business security requirements but also for security implications when combining data from across lines of business.

Pre-conditions

Pre-conditions for consideration of a data hub include:

- Multiple authoritative systems for data that must be integrated.
- Multiple contributors and consumers of the integrated data.
- A variety of representations of data, that may consist of varying structured data models, unstructured data, or a mix of both.
- A pace of business change (and dependent data modeling changes) that makes traditional extract-transform-load (ETL) and remodeling unsustainable.
- Operational requirements as described in the Problem Context section above.

Solution Details

The Solution describes a persistent integrated real-time data store that provides the following capabilities:

- **Self-describing schema.** Ingestion of source data in a self-describing format (e.g., XML, JSON, RDF or combinations of all three). Source data that does not exist in such formats would undergo **minimal** processing to represent the data suitable for ingestion. For example:
  - Tabular data (e.g., RDBMS, CSV, etc.) data would be represented in equivalent name-value format using XML or JSON such that each row instance would be converted to a document or object using the tabular data values with column names enclosing them as entities or attributes as appropriate.

- Textual data with no structure or markup would simply be enclosed in a single document as text.

- **Indexed.** All data would be indexed upon ingestion so that it is immediately available for discovery, regardless of format or changes to format over time.
- **In-database Transformation.** The data store will provide the ability to perform transformations inside the database on an as-needed basis. Original formats are retained.
- **Provenance and Lineage.** Data may also be decorated with metadata to capture source information as well as transformation details.
- **Real-time Capable.** The data store will have the ability to ingest data in batch or real time.

The resulting implementation will consist of multiple data sources contributing to the data hub in either real-time or batch modes. As needed, data across the sources will be inter-related and/or harmonized into canonical (or partially canonical) formats. There will also be multiple consumers of the data, some in batch mode and some in real time.

Resultant Context

Within an existing Enterprise Architecture, a data hub would be the first place where data is integrated outside of source OLTP systems.

Diagram

The following diagram depicts the data hub according to the Archimate standard[24].

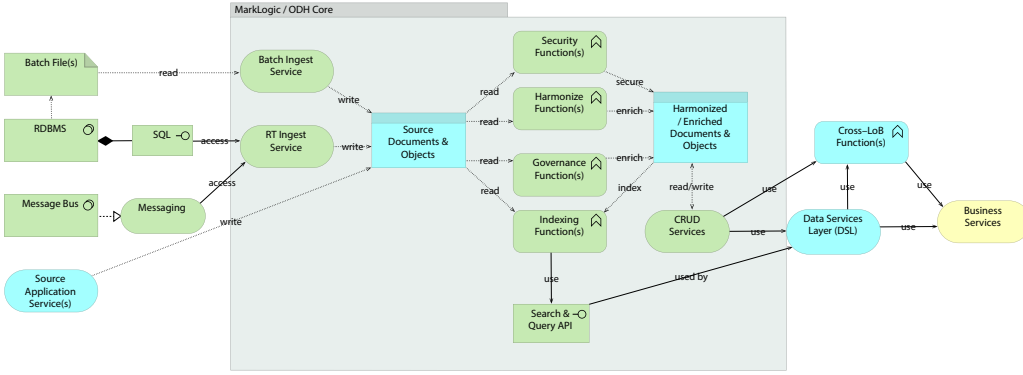


Figure 15: Archimate Diagram

24 http://www.opengroup.org/subjectareas/enterprise/archimate-overview

Bibliography

Allemang, D. & Hendler, J. (2011). *Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL*. Waltham, MA: Morgan Kaufmann.

> This book discusses the capabilities of Semantic Web modeling languages, such as RDFS (Resource Description Framework Schema) and OWL (Web Ontology Language). Organized into 16 chapters, the book provides examples to illustrate the use of Semantic Web technologies in solving common modeling problems.

Aven, P. & Burley, D. (2017). *Building on Multi-Model Databases*. Retrieved from http://www.oreilly.com/data/free/building-on-multi-model-databases.csp.

> This concise eBook examines how multi-model databases can help you integrate data storage and access across your organization in a seamless and elegant way.

Bowers, M. (2017). *Becoming a Document Modeling Guru* [Video and PowerPoint slides]. Retrieved from https://www.marklogic.com/resources/becoming-document-modeling-guru/.

> This presentation from the 2017 MarkLogic World conference features Michael Bowers, Principal Architect, LDS Church and author of *Pro CSS and HTML Design Patterns* published by Apress (1st edition in 2007, 2nd edition in 2011). The session addresses common questions people have when considering a move from a relational model to a NoSQL document/graph model.

Castanedo, F. (2017). *Understanding Data Governance: Practices and Frameworks for Regulatory Compliance and Security*. Retrieved from http://www.oreilly.com/data/free/understanding-data-governance.csp.

This report explains how data governance can be established at the data level, lays out practices and frameworks for regulatory compliance and security, and explores different data governance technologies, including NoSQL and relational databases.

Feldman, D. (2017). *Cleaning Up Your Enterprise Data Architecture* [video file]. Retrieved from https://www.marklogic.com/resources/cleaning-enterprise-data-architecture/.

This on-demand webcast, originally hosted by O'Reilly, features Damon Feldman, Ph.D., Solutions Director at MarkLogic. The presentation explains how a data hub can be combined with the data lake – maintaining the agile, flexible functions that are characteristic of a lake, with the added ability for fast, secure, governed data access. Learn how a data hub can utilize a Hadoop-friendly, multi-model database, without reducing it to relational structures and applying ETL transformations.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley.

Also known as the "GoF book" and now in its 37th printing [March 2009, Kindle edition], this book captures a wealth of experience about the design of object-oriented software. Four top-notch designers (collectively referred to as the "Gang of Four") present a catalog of simple and succinct solutions to commonly occurring design problems.

Hunter, J. & Wooldridge, M. (2016). *Inside MarkLogic Server*. Retrieved from https://www.marklogic.com/resources/inside-marklogic-server/.

An interactive and conceptual exploration of MarkLogic's multi-model database. This comprehensive eBook gives you insight into what's going on under the hood, with a high-level introduction in chapter one followed by a deep-dive into technical capabilities that developers, programmers and data architects will appreciate.

MarkLogic (2015). *Beyond Relational*. Retrieved from https://www.marklogic.com/resources/beyond-relational/.

This white paper discusses why relational databases are ill-suited to handle the massive volumes of disparate, varied, and changing data that organizations have in their data centers.

MarkLogic (2017). *Top Data Security Concerns When Integrating Data*. Retrieved from https://www.marklogic.com/resources/top-data-security-concerns-integrating-data/.

> This white paper discusses the top data security concerns CIOs, architects, and business leaders should focus on at a strategic level – with a particular focus on data integration – and provides an overview of how MarkLogic addresses those concerns as a database.

Moore, G. (2015). *Zone to Win: Organizing to Compete in an Age of Disruption*. New York, NY: Diversion Books.

> Geoffrey Moore's classic bestseller, *Crossing the Chasm*, has sold more than one million copies by addressing the challenges faced by start-up companies. Now *Zone to Win* is set to guide established enterprises through the same journey.

The Open Group (2007). Architecture Patterns. In *TOGAF 8.1.1 Online*. Retrieved from http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap28.html.

> The Open Group Architecture Framework (TOGAF) is a framework – a detailed method and a set of supporting tools – for developing an enterprise architecture. It may be used freely by any organization wishing to develop an enterprise architecture for use within that organization. Chapter 28 of this online publication provides guidelines for using architecture patterns.

Ungerer, G. (2018). *Cleaning Up the Data Lake with an Operational Data Hub*. Retrieved from http://www.oreilly.com/data/free/cleaning-up-the-data-lake-with-an-operational-data-hub.csp.

> Data lakes in many organizations have devolved into unusable data swamps. This eBook shows you how to solve this problem using a data hub to collect, store, index, cleanse, harmonize and master data of all shapes and formats.

999 Skyway Road, Suite 200 San Carlos, CA 94070
+1 650 655 2300 | +1 877 992 8885
www.marklogic.com | sales@marklogic.com