ORACLE

Technology Brief:
Dynamic Data Fabric and Trusted Data Mesh using the Oracle GoldenGate Platform

Core Principles and Attributes for a Trusted, Ledger-based, Low-latency Streaming Enterprise Data Architecture

January 2021, Version 2.1

Disclaimer

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Document Purpose

The intended audience for this document includes technology executives and enterprise data architects who are interested in understanding Data Fabric, Data Mesh and the Oracle GoldenGate streaming data platform. The document's organization and content assume familiarity with common enterprise data management tools and patterns but is suitable for individuals new to Oracle GoldenGate, Data Fabric and Data Mesh concepts.

The primary intent of this document is to provide education about (1) emerging data management capabilities, (2) a detailed enumeration of key attributes of a trusted, real-time Data Mesh, and (3) concise examples for how Oracle GoldenGate can be used to provide such capabilities.

This paper applies to Oracle Cloud and also to multi-cloud enterprise environments.

Table of contents

Executive Summary

Business transformation initiatives are being held back by outdated thinking about data and an older generation of monolithic data tools. Modern enterprise data estates are becoming more decentralized and multi-cloud, data entropy is unavoidably increasing over time. To achieve business transformation goals, business teams need freely streaming, well governed data.

Monolithic data architectures, whether in-cloud or on-premise, will not deliver the necessary modularity and speed required for business success. By bringing together new thinking and modern technologies, Dynamic Data Fabric and Trusted Data Mesh are set to provide a new path forward that will unlock more value from the enterprise data estate. Adopting this new approach will empower faster innovation cycles and lower cost of data operations as a result of smarter automation and fewer complexities in the data supply chain.

All enterprise scale businesses have a data estate. Hundreds or even thousands of applications, data stores, data lakes and analytics may run the business or drive decision-making across many lines of business. Market winners will be those enterprises that succeed at driving more value from their data estate by disrupting existing markets and creating new opportunities. Successful digital transformation initiatives are rooted in sound strategy and excellent execution (ie; people and process), but technology has always had a very important role to play in creating new efficiencies, powering new innovations, and opening up new opportunities.

Dynamic Data Fabric and Trusted Data Mesh offer an entirely different and better approach for enterprises to build and govern their data estates. This new approach is a new kind of enterprise data architecture that prioritizes data product thinking, fully embraces the decentralization of data, and is built to preserve trusted, correct data within real-time streaming data platforms. An industry trusted Data Fabric foundation is core, with a focus on dynamic, streaming data and value-based data product management. Data Mesh concepts are also a central aspect of the approach, but with a technical foundation that can ensure data consistency, correctness, and trust.

This new kind of data architecture will empower faster innovation cycles and lower costs of operations. Evidence from early adopters and pioneers of this approach indicate significant large-scale benefits that are possible today:

- **Total clarity into data's value chain** – through applied 'data product thinking' best practices
- **>99.999% operational data availability** – using microservices based data pipelines for replication
- **10x faster innovation cycles** – shifting away from ETL, to continuous transformation and loading (CTL)
- **~70% reduction in data engineering** – using no-code and self-serve data pipeline tooling

It is indeed a rare opportunity to find a solution that can work equally to reduce costs associated with ongoing operations while also powering up innovation for the business units working to use data as a competitive advantage – this is one of those moments.

A Dynamic Data Fabric for Enterprise Needs

First popularized in the early 2000's the Data Fabric was initially most associated with in-memory object grids. Then Forrester began writing about more general data fabric solutions and by 2013 the Data Fabric became a full-fledged research category[i]. The concept of a Data Fabric has become pervasive, and Gartner has even declared that "*Data Fabric Is the Future of Data Management*"[ii]. Today, the Data Fabric topic applies to a wide set of data technologies.

Generally, there is consensus that there is no single tool that encompasses the full breadth of a Data Fabric. Rather, for organizations that adopt a data fabric, it is a design concept that spans many 'styles' of data integration and governance to achieve a harmonized and cohesive solution. Forrester's definition is that a Data Fabric "*delivers a unified, intelligent, and integrated end-to-end platform to support new and emerging use cases. The sweet spot is its ability to deliver use cases quickly by leveraging innovation in dynamic integration, distributed and multicloud architectures, graph engines, and distributed in-memory and persistent memory platforms. Data fabric focuses on automating the process integration, transformation, preparation, curation, security, governance, and orchestration to enable analytics and insights quickly for business success.*"[iii]

Oracle is an independently recognized[iv] leader in the Data Fabric and the full portfolio includes:

CLOUD-NATIVE, COMMON PLATFORM DATA FABRIC	BEST-OF-BREED DATA FABRIC FOR MULTI-CLOUD & ON-PREMISE
• **Self-Service ETL for Analytics & Autonomous DB** • **OCI Data Catalog, OCI Data Integration, OCI Data Flow** • **OCI GoldenGate and Stream Analytics for OCI** • **Integration Cloud and Oracle Cloud SQL**	• **Oracle Data Integrator (w/ETL, Quality, Messaging)** • **Oracle GoldenGate and Stream Analytics** • **Oracle Big Data SQL (Data Federation)** • **Oracle Data Visualization (Data Preparation)**

Within the Oracle portfolio, the Oracle GoldenGate platform is distinctly focused on the concept of a dynamic Data Fabric – focusing on real-time replication, streaming, time series analytics and in-memory data processing for decentralized, multi-cloud ecosystems.

Oracle Concept of a Trusted Data Mesh

The term 'Data Mesh' has been used as far back as the 1990's in reference to 3D laminates and digital microscopy[v] and by the early 2000's the term appeared as a way of explaining how TCP/IP networks work[vi]. In the context of data management in the late 2000's, we see Data Mesh first become more common in papers[vii] describing the early Semantic Web initiatives such as the Web Ontology Language (OWL) and Resource Description Framework (RDF). Around this time, in 2007, there is also an informal "Data Mesh" wiki definition referencing "*a network for routing data from any point to any other point […] across heterogeneous networks that are active only intermittently*"[viii].

More recently, the concept of a Data Mesh for enterprise data was noted in a 2016 Gartner report "*Maverick* Research: Revolutionizing Data Management and Integration With Data Mesh Networks*"[ix]. Then, in 2019 the ThoughtWorks paper "*How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh*"[x] has reanimated the concept as it began to catch on more broadly.

As you can see, there are different sources for the origin of the Data Mesh term, but for the purposes of this document we will be using the ThoughtWorks paper as the most recent point of reference. In that paper, the case is made that "*the data mesh platform is an intentionally designed distributed data architecture, under centralized governance and standardization for interoperability, enabled by a shared and harmonized self-serve data infrastructure.*" A principal focus of the Data Mesh is to place emphasis on 'domain data products' as a first-class concern and to promote the decentralization of data using 'immutable data sets' as products.

The heritage and DNA of this new conceptualization of a data mesh stems from Microservices design-thinking concepts such as Domain-driven Design (DDD) and Event Sourcing. ThoughtWorks leadership (Martin Fowler, et al) have been long-time leaders in distributed software, Agile development, and Microservices mesh design patterns. It is

this heritage of thinking that have informed some of the best aspects (de-centralization, monolith decomposition, data product thinking, etc) of their framing of Data Mesh, but it also leads to some substantial blind-spots (eventual consistency, dependency on developer heuristics, physics of data management at petascale) as well.

In this document, the Oracle definition builds on prior definitions of Data Mesh and goes further to emphasize elements of strong data consistency (trusted data transactions), governance and verifiability of data (data validation) and enterprise-scale demands (peta-scale data movement on mission-critical data). In this converged definition, **a trusted Data Mesh is a data architecture approach focused on outcomes (data products), IT agility in a multi-cloud world (mesh), trusted data of all kinds (polyglot data streams), and faster business innovation cycles (using event-driven data ledgers).**

Of all the words which could have been chosen to describe a Data Mesh, the word 'mesh' is familiar, emotive and uniquely fits the bill. Like other technology 'meshes' that you are familiar with (eg; WiFi Mesh, Smart Home Mesh, 5G Mesh, Service Mesh/Kubernetes etc) a recurring and central attribute of Data Mesh is the archetype of a networked, de-centralized architecture. At core, the mesh is about rejecting centralized monolithic architectures.

There are many other crucial attributes of a Data Mesh that we discuss in this document, but it is this "mesh'iness" that is core to understanding the technological break from the past and why this new approach better prepares us for a future where all applications, services and analytics are inherently distributed and multi-cloud.

GoldenGate as a Bridge to Fabric / Mesh

A San Francisco startup founded in the 1990's, GoldenGate's original purpose was to provide business continuity and data high availability for networked ATM/cash machines running from Tandem NonStop databases.

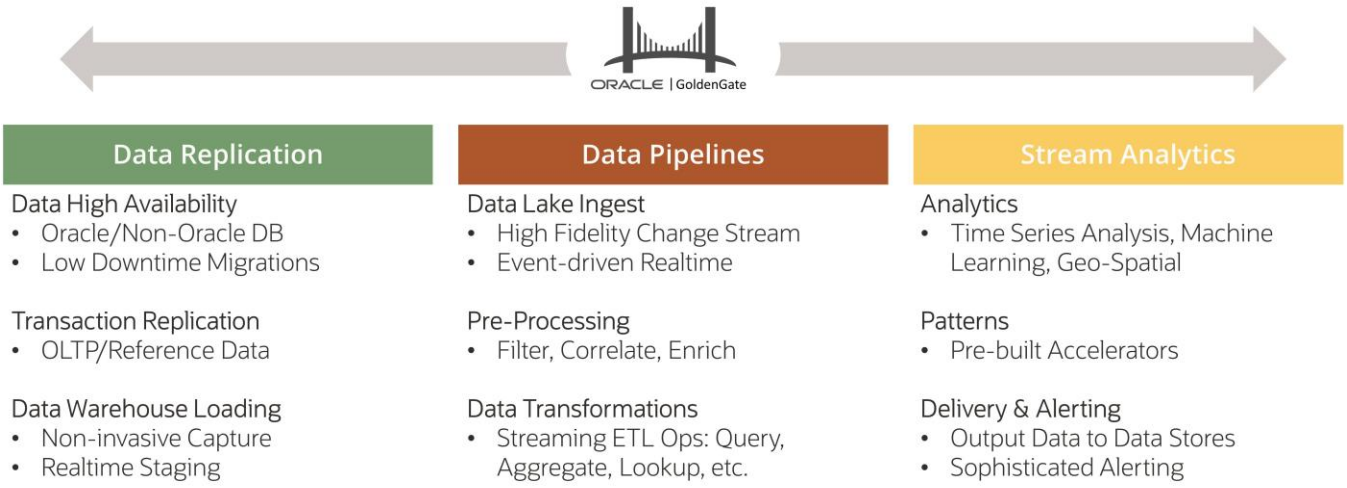


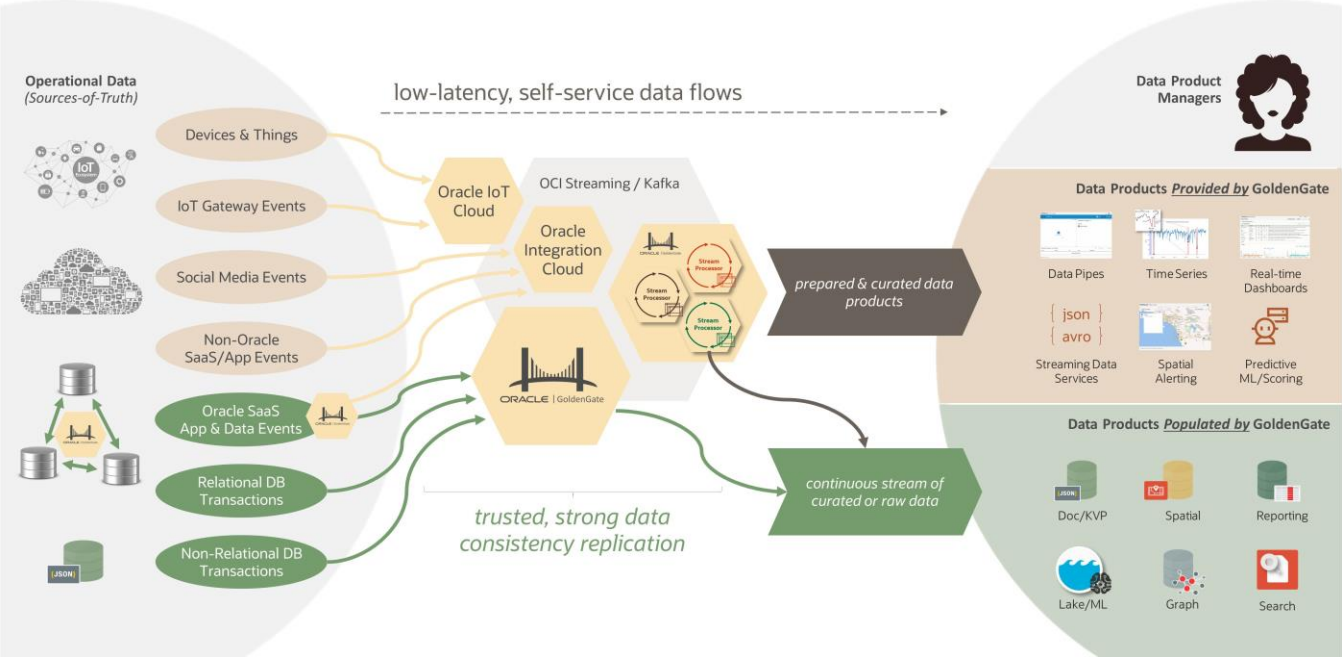
Figure 1: GoldenGate platform capabilities

Today Oracle GoldenGate software still provides business continuity and data high availability for databases like NonStop, DB2 iSeries, LUW and mainframes, SQL Server and GoldenGate is also the pinnacle of the Oracle Database Maximum Availability Architecture 'Platinum Tier' service level. Many thousands of global banks, retailers, telecoms, healthcare companies etc. run their operational data platforms on the trust foundation of Oracle GoldenGate.

At core, GoldenGate is a real-time data replication tool that can detect data events and route them across networks at very low latencies. The GoldenGate technology is used for geographic sharding of operational databases, low-downtime data migrations, multi-active (online) data stores, real-time data ingestion to cloud, data lakes, and data warehouses etc. Since 2015 GoldenGate has been increasingly focused on polyglot big data and noSQL data payloads and has been completely refactored for native Microservices 'as a service' deployments.

In 2018 the GoldenGate platform added Data Pipelines and Stream Analytics with a robust complex event processing (CEP) core engine that scales to billions of events per second while preserving ordered data processing down to the nano-second scale. This event engine can use very powerful semantics for transformations or analytics and runs on an open-source Apache Spark for massively parallel processing (MPP).

With these new capabilities, GoldenGate provides high-value data products directly to data consumers. In the past, GoldenGate would have mainly been used to deliver low-latency raw data to data pipelines or other data products. Today, GoldenGate can push raw data as well as provide high-value data products.



DATA PRODUCTS PROVIDED BY GOLDENGATE	DATA PRODUCTS POPULATED BY GOLDENGATE
• **Data Pipelines & Streaming Data Services** • **Time Series Analytics and Production ML/AI Scoring** • **Geo-Spatial Alerting and Real-time Dashboards**	• **Data Marts or OLAP/ROLAP Analytics** • **Data Lakes and Cloud Analytics** • **Doc, KVP, Search and Graph Data Stores**

Figure 2: Creating and populating data products with GoldenGate

Going forward, business leaders will require more streaming data, more real-time events, greater data availability, and strong governance for transactions. Trusted data fabrics must preserve a strong data consistency from the data source-of-truth all the way to any downstream data lake or analytics tooling that may consume the data. GoldenGate is a proven, trusted Data Fabric platform to make the pivot into a Data Mesh future.

What's next? In the remainder of this document, we will diver deeper into the following:

- **A Time for Change –** explore the outmoded thinking and cultural changes that must change, and evaluate next-gen tech innovations that are the springboard for this new data architecture approach
- **Principles of the New Paradigm** – succinctly summarize the 4 key principles: Data Product Thinking, Decentralized Mesh Networking, Enterprise Data Ledgers, Polyglot Data Streams
- **Dynamic Data Fabric as a Trusted Data Mesh** – deep dive into attributes and characteristics of what makes a Data Fabric dynamic, and how a Data Mesh can provided trusted, 100% consistent data streams
- **Explainer for Oracle GoldenGate** – mapping the product characteristics of existing GoldenGate technologies into the Dynamic Data Fabric and Trusted Data Mesh patterns

A Time for Change

After 35 years of preeminence, centralized and monolithic data architectures are ripe for change. As with any software pattern that has remained relevant for multiple decades, we can certainly say that the monolithic data architecture has been successful. In fact, so successful that all enterprise IT organizations no doubt already run many data monoliths in production. These data monoliths may include batch ETL tools, Operational Data Stores (ODS), Enterprise Data Warehouses (EDW), and Data Lakes on-premises or in a public cloud.

> ***For more than 35 years the Data Monolith has been the dominant paradigm of thinking for enterprise data architecture and now that paradigm is about to change.***

In the wider software development ecosystem, the winds of change have already been blowing. Widespread adoption of Microservices and Service Mesh (Kubernetes et al.) architecture has been directly a result of the desire to move beyond the classical software monoliths of the past. Business and developer desires for greater agility, modularity, change tolerance and faster innovation cycles have created a sea-change in the way enterprise software is written.

In the data world, monolithic data architectures have many of the same fundamental characteristics (of monolithic applications) which now block important improvements necessary for data modernization and business transformation. Attributes of classical data management monoliths include:

TREAT DATA AS AN IT ARTIFACT	*Data is treated as a byproduct of the application functions, requiring the semantics of domain modeling to be done and re-done many times in the IT lifecycle*
MONOLITHIC AND CENTRALIZED	*Hub-and-spoke style architecture dominates the IT ecosystem, but each App, ETL, Mart, Warehouse, Lake etc presumes itself to be the center of the data landscape, thereby requiring IT to constantly fund projects that integrate and align data using incompatible tools*
WATERFALL DATAOPS / DEVOPS	*In the "data landscape" of databases, ETL, Marts, Warehouse, Lakes etc etc there remains a strong bias towards waterfall-style operations, the Agile methodologies of development haven't been able to provide a repeatable CI/CD lifecycle approach across the monoliths*
BATCH PROCESSING CENTRIC	*Movement of data for most Analytic (OLAP) domains remains predominantly batch-oriented… to be driven by the scheduler (clock) rather than the events of the business itself*
OLTP VS. OLAP (DECOUPLED)	*Different operational applications (OLTP) and analytics (OLAP) are often separated organizationally, politically, and technically – causing IT friction (delays), data domain semantic issues (reduced data trust), and least-common-denominator solutions (low innovation)*

Changes in classical tooling and architecture design are necessary and important, but perhaps more importantly, there is also a need for widespread paradigmatic change in thinking about the intrinsic value of data. Increasingly, some business leaders are already thinking about data as an asset. In fact, data is a kind of capital. This isn't a metaphor like "data is the new oil" or "data is the new gold". Data fulfills the literal, economic textbook definition of capital. Capital is a produced good, not a natural resource. You have to invest to create it, not just dig it out of the ground. More importantly, data capital is a necessary factor of production into other goods and services. In this new paradigm of thinking, data is a tangible asset of value that should be cohesively managed across the many decentralized, digital infrastructures that contain it.

To move forward we have reached a point where a paradigm shift is necessary. This shift must affect both our mindset (approach to thinking) as well as the tools and technologies we bring to bear on the solution space.

Model Crisis: The Data Monolith

As any good enterprise architect knows, there are no silver bullets or magical solutions for software problems. All technology comes with tradeoffs, costs and consequences. The same is true with Monoliths and Mesh. Since this document's focus is on explaining the new and emerging areas of Data Mesh (and Data Fabric) it is necessary to discuss the many issues with Data Monoliths which have brought us to this point.

Paradigm shifts occur when the dominant paradigm becomes incompatible with new phenomena, facilitating the adoption of a new theory or paradigm[xi]. For more than 35 years the Data Monolith has been the dominant paradigm of thinking for enterprise data architecture and now that paradigm is about to change. In the language of paradigm theory, a 'model crisis' is the third step of a cycle that occurs when the established dominant paradigms can no longer adequately cope with the emerging new systemic realities facing an organization.

The rapid adoption of public cloud technology since 2015 is causing an increase of data entropy

The emerging truth of modern IT is that an enterprise 'data estate' (the set of data stores and data centric applications that contain an organizations data capital) are more decentralized than ever. The rapid adoption of public cloud technology since 2015 is causing an increase of data entropy, the distribution of data capital across a wider set of physical locations, data stores and networks. But the old Data Monoliths are not ideal for this new reality.

Classic monoliths of data management are not optimal for multi-cloud ecosystems where data must be (a) cohesively managed across infrastructure separated by 100's/1000's of kilometers, (b) use different data formats (polyglot) and (c) stay connected at very low latencies. These challenges affect the act of managing and governing data that is inherently decentralized. Data Monoliths, such as the following, are not well suited for these newer more decentralized data challenges.

- **Classical ETL and Data Preparation Tools** – which typically require processing in a centralized processing engine. And, don't be fooled by claims of 'push down' or 'E-LT' processing, most tool vendors who claim these capabilities still require data to be centralized into a single infrastructure hub for staging and workloads, which is the classical definition of a monolith. The claims about 'E-LT' often refer to the tool's use of a data store based processing engine (such as a Database, Hadoop, Spark, or a Serverless Cloud option) rather than an embedded proprietary engine. Only a few batch processing tools, such as Oracle Data Integrator, run as an 'agent' and allow developers to move the workloads around to many different engines and locations.
- **Classical Data Replication Tools** – a longstanding part of operational data architectures, the use of Change Data Capture (CDC) tools emerged in the 1990's as a more event-driven alternative to the ETL tools. The classical architecture for CDC tools is a Data Monolith, requiring 'hub style' deployments and tightly-coupled databases for metadata and systems frameworks. These designs are not suitable for modern mesh style deployments discussed in this document.
- **Operational Data Stores (ODS)** – often used as part of classical Kimball and Data Vault architectures, the ODS is used to collect and store application data. This application data is typically used by downstream Data Marts or Data Warehouses, and the usual flow of data in/out of the ODS is typically done on a batch process with conventional ETL tools.
- **Enterprise Data Warehouse (EDW)** – in many ways the EDW is the classic definition of a data monolith. There are many conceptualizations of what an EDW can be (Kimball, Inmon, Data Vault, etc) but for several decades the central idea was that there could be one single source of the truth for enterprise reporting, with a common set of 'subject areas' and data domains. In practice, what most organizations typically wind up with are many data warehouses and data marts, often aligned to particular business units.
- **Data Lakes (reservoirs, ponds, swamps, lake houses etc)** – really, nothing says 'monolith' quite like Big Data, which is pretty much the mainframe of data analytics. In earliest form, the Hadoop ecosystem literally required the tight coupling of storage to compute and mandated nearly 200 different Apache and Java frameworks run. Newer data lakes still require technology centralization into a particular cloud vendor or Big Data technology stack, and just as with EDWs the modern enterprise is most likely running several different data lake technologies in combination and not just one mega centralized one.

Many individual data stores are monolithic to some degree, and a holistic enterprise data estate will always out of necessity have many, many data stores (some old and some new). Data preparation, pipeline and integration tools don't have to be and shouldn't be monoliths, for the sake of DevOps, DataOps and CI/CD (continuous integration,

continuous delivery). Data monoliths aren't going away overnight, and there may occasionally be situations where monoliths are even a preferred architecture to lead with. But just as organizations have adopted Microservices and Service Mesh to transition away from application monoliths, the Data Monolith remains a barrier to progress that must be overcome in order to unlock the imperative for large-scale data-driven business transformation.

Old Ways of Thinking and Assumptions to Abolish

Far too many times, "because we have always done it this way" has become an excuse for avoiding change. To move forward with a new paradigm, we must change some established ways of thinking. These mindset changes are often the most difficult to materialize in the culture of complex business organizations. The following critical assumptions about data management will need to be inverted for a dynamic Data Fabric or a trusted Data Mesh to succeed.

Old Assumption 1: Treating Data as a By-product / IT Artifact

Conventional wisdom for application development going back for decades has been to treat data as a byproduct of the software, rather than as an asset to be directly managed and governed. Perhaps in the beginning this was a pragmatic decision since physical data (as documents or in database tables) often needed to be physically optimized for efficient and durable processing in older technologies. The physical form of the physical data was typically different from the logical form of the data being held in-memory by the software application. This kind of 'impedance mismatch' creates a logical separation between data that the application developer works with (as objects) and what the data engineer works with (as physical structures).

Today however, the 'impedance mismatch' problem is far more complicated. The universe of data structures and formats in the 2020's is much more complex than it was in the early 2000's. Broad industry acceptance of 'polyglot persistence' and 'converged database' strategy means that business data is now regularly formatted in many diverse formats that include relational, key-value-pair, document, graph, time-series and object structures. Both the in-memory representations as well as the physical durable formats of data are more polyglot than ever.

Not coincidentally, we now see the critical need to manage data in a more format-agnostic manner. Rather than managing data as some sort of 'application exhaust,' the need arises to conceptualize the essence of the data as domains, so that the logical ideas that the data represent can be more effectively managed no matter their physical or in-memory formatting. Therefore, we invert the thinking of managing data as an IT by-product, and we begin to see data as belonging to business domains and as a kind of product to be managed and cared for within a lifecycle.

Old Assumption 2: OLTP → ETL → OLAP Fragmentation

For many decades, the industry has depended on batch-oriented data architecture to link the online transaction processing (OLTP) data stores to the online analytic processing (OLAP) data stores. Scheduler-driven ETL tools move files and run daily/weekly batch jobs to move data around. The underlying meaning of the OLTP data objects, data models and integrations is largely decoupled from the meaning of the analytic data objects, data models, machine learning features, and data integrations. It is only by way of a loosely associated complex graph of programs, mappings, store procedures, scripts, data pipelines (and occasionally a disassociated data catalog) that we know how our analytic data relates back to the application sources-of-truth. This is a fundamentally broken way of managing data in distributed architectures.

To invert our thinking we need to consider how simple it would be if the same Data Fabric that joined up OLTP sources-of-truth could also be used to join and manage data within Analytic frameworks. Keeping transactions cohesive and consistent while also keeping impedance mismatch to a minimum would greatly reduce the sheer complexity of data pipelines needed to move data around. A dynamic Data Fabric and trusted Data Mesh should provide a blueprint for OLTP and Analytic domains to interoperate at scale on the same foundation.

Old Assumption 3: Waterfall Process for Data Projects

Because OLTP and Analytic projects have conventionally been split between different functional business units or IT organizations, the people and process methodologies that drive projects typically failed to harmonize. As such it is

exceptionally difficult to provide for an Agile development methodology in large-scale data management initiatives – the traditional optimizations provided for by Agile methodologies either do not apply or are not shared between organizations. Compounding these cross-organizational challenges is the fact that most large-scale data management is still using monolithic data tooling that has many layers of complexity in DevOps and CI/CD (Continuous Integration, Continuous Delivery).

For precisely the same reasons that Agile development methodologies eventually drove a revolution in thinking around monolithic software architectures (towards Microservices and Service Mesh), the desire to be more Agile in data management is driving a re-think of the data management architectures (eg; Data Fabric and Data Mesh). In order to maximize agility in data management, it is a necessary step to decompose our data architectures into smaller, more modular and more dynamic components that are loosely-coupled.

Old Assumption 4: Batch-first Design Thinking

Since the very beginning, IT data management has revolved around batch processing for data movement and data processing workloads. Sure, there have been real-time systems around for decades but they have existed as outliers; a 'bolt-on' to the central paradigm of moving and processing data in batches. Batch processing is not going away anytime soon, but the time has come for data architects and data modelers to invert their 'batch-first' design assumptions. When we imagine the future, we don't think about waiting hours for stale data to arrive at our devices. Planning for and modeling an IT infrastructure that is 'streaming-first' is possible today, and it is the foundation for planning a dynamic Data Fabric and a trusted Data Mesh.

Old Assumption 5: Hub-and-Spoke Architecture Bias

In the same way that mainframes and minicomputers (eg; DEC VAX) have always had remote terminals, the hub-and-spoke architecture has been a defining pattern of computing since the earliest days. Even now, with the rise of public cloud computing data lake concepts, we see the natural urge to centralize data and workloads into singular centralized physical locations. However, as the world metaphorically shrinks and more businesses go global, enterprise data naturally becomes more fragmented across a myriad of infrastructure spanning different applications, clouds, edge devices, and on-premise systems.

Invert these old hub-and-spoke assumptions and we can create a future where the multitudes of enterprise data producers and data pipelines are efficiently modeled in their natural state as a DAG (directed acyclic graph), or a mesh. Workloads (eg; data ingest, ledger events, transformations or streaming analytics) can occur on any node of the graph. Monolithic hubs (such as ETL tools, EDW's and Data Lakes) will not necessarily go away, conceptually these monoliths as just a small part of a wider Data Fabric. Once the Fabric/Mesh is in place, we free up IT to be more agile in experimenting with alternatives to the monoliths. Rather than becoming decades-old IT 'cruft'[xii] and 'technical debt', data monoliths can become a point-of-time evolution to simply iterate away from when and if the business needs arise. Note that a 'graph of hubs/monoliths' is not the same thing as a data mesh.

Old Assumption 6: Storage-centric Modeling

Data is dynamic. All data is born fast, as an event, and the highest-value data is often in motion – utilized within milliseconds of its inception. But historically our data models and our entire data architecture approaches data as if it were some static thing existing as ones-and-zeros on an aluminum hard drive platter somewhere. Over time, this storage-centric approach to data becomes ridged and inflexible as the attributes of data inevitably change. It is this 'data drift' across document payloads, application upgrades, ETL mappings, and changing dimensions on EDWs that result in the very high percentage of costs associated with integration spending[xiii].

To prepare for the inevitable future of data in-motion, we must invert our thinking of data to model the transactions, event logs, and data artifacts as objects that inherently exist in a temporal plane. Like a chessboard, the position of the pieces at any moment in time is simply a snapshot, whereas the sequence of moves (eg; the events) is what actually tells the story of a chess match. Similarly, to fully understand the full story of enterprise data we must able to

descriptively model the dynamic nature of data – where both the schema and data-values of discrete business objects will change over time.

Why Now? Enabling Technologies

There is no singular invention or programming language driving the Data Fabric and Data Mesh innovations. Wholesale architecture shifts rarely happen from singular lightning-bolt breakthroughs, and more often emerge over time from a cohort of smaller innovations and step-improvements that happen as enabling technologies are more pervasive and easier to use. These enabling technologies serve as the foundation from which new system-wide architecture possibilities emerge.

The previous section described many old ways of thinking and assumptions which must be inverted, but there are also an array of important technology step-change improvements worth noting in more detail. These foundation technical innovations enable the dynamic Data Fabric and trusted Data Mesh in ways that simply were not possible even a few years ago.

Enabling Tech 1: Service Mesh, Kubernetes etc

Taking the software development ecosystem by storm since 2015, the Microservices architecture revolution has also empowered a sea-change in how we approach DevOps and CI/CD (continuous integration continuous delivery). Paired with workload containerization (eg; Docker) and a heavy application of the 'Sidecar Pattern' (for distributed administration) in tools like Kubernetes and OpenShift, we can now apply Agile DevOps methodologies in life-changing ways that just were simply not possible before. For example, conventional DevOps tasks such as patching, upgrades, code injection, re-platforming, auto-scaling etc. can now often be done much more simply using pod-management controls in a Service Mesh, with much less risk than previously possible.

As the next generation of data management concepts like Data Fabric and Data Mesh evolve, the application of Service Mesh technology is a fundamental enabler to power greater agility, better DevOps, CI/CD and to provide the foundation for superior DataOps process methodologies.

Enabling Tech 2: Mature Software Defined Networking

It is difficult to over-state the impact of SDNs (software defined networks) on the ability of IT to conduct effective multi-cloud operations. Before the rise of SDNs, the enigma of networking was strictly in the domain of highly specialized network engineers working with physical switches, gateways, routers, load-balancers, VPN hardware etc. Today, effective secure networking is still an incredibly difficult challenge, but it can now be largely managed from within the software tier of major cloud providers. It is astounding to be able to coordinate network traffic across secure domains, tenancies, IPSec VPN tunnels, VNCs (virtual networks), private endpoints / reverse-connections etc. all via APIs and cloud user interfaces.

This SDN simplification has allowed IT generalists to take on more self-service networking tasks, and has caused a massive shift in the accessibility of managing data/applications across de-centralized infrastructure like never before. Radical flexibility to move data and workloads across data centers is both (i) driving demand for and (ii) powering the new capabilities of a dynamic Data Fabric and a trusted Data Mesh.

Enabling Tech 3: Microservices for CDC Replication

Changed data capture (CDC) tools detect events within data stores and data replication tools move data across LAN/WANs. They often, but not always, go together. In the beginning these kinds of tools were like all other data integration tools, they were monolithic applications with all of the disadvantages that come with the territory. But since 2016 some tools (Oracle GoldenGate is described in detail in part four of this document) have shifted towards a true microservices foundation that include 100% REST based API-driven, fully encapsulated light-weight services, and zero dependencies on external data stores for metadata. Technically, this re-factoring of the foundation empowers a Service Mesh and has a life-changing impact on DevOps, CI/CD and Agile development possibilities.

Enabling Tech 4: Polyglot and ACID Payloads in DB Replication

As far back as the 1990's data replication tools have been focused on replication of highly-consistent data transactions that originate from ACID (atomicity, consistency, isolation, and durability)-capable databases. Logical replication has been used to support mass-scale High Availability (HA), Disaster Recovery (DR), Global Data Services (GDS), and geo-graphically distributed data sharding. This laser-focus had the key benefit of preserving the reliability and trust of data transactions that require ACID properties (banks, fin-tech, telecommunications, back-office tools, ERP systems etc.). With data replication, we can move data quickly across vast physical distances with confidence that data integrity won't be lost.

From 2015 onwards, the rising popularity of big data, data lakes and NoSQL document stores ushered in a number of valid use cases for 'relaxed data consistency' particularly in analytic data stores supporting non-transactional workloads. As such, some of the data replication tools (including Oracle GoldenGate) have already made the pivot to cover non-relational data payloads as well. A modern data replication tool (like Oracle GoldenGate) is now capable of 'polyglot replication' – meaning it can equally cover strong ACID data consistency transactions, as well as many-formatted payloads such as XML, JSON, Avro, ORC, Parquet etc.

Unlike popular open-source messaging frameworks (such as Apache Kafka, which are not ACID-capable), data replication tools (like Oracle GoldenGate) can correctly handle ACID transactions and bring a much-needed degree of trust to a Data Mesh.

Enabling Tech 5: ETL Can Run in a Stream

Since the 1990's ETL tools have been based on monolithic architectures and required batch-processing. ETL jobs run on a schedule, not from a singular data event. Of course, it has long been possible to transform data using event-driven messaging engines, but these could never really achieve the scale necessary for database workloads.

Beginning around 2015 the industrialization of event stream processors (eg; an early example being Apache Storm) opened up the possibilities around what came to be called the big data Lambda and Kappa architectures. Lambda and Kappa both provide for ETL in a scaled-out data stream, but Kappa architecture combines both batch and stream processing workloads into a single engine. The origins of the Kappa technology stack were first focused on consumer data (eg; log events, social media etc) however the base technology has now evolved to a state where it is quite capable of enterprise-type data workloads.

Running enterprise ETL in a continuous stream is a fundamental technology and mindset shift – rather than waiting for schedulers we can now transform and load data transactions as soon as they become available. By eliminating the natural 'time windows' of a scheduler, we can now manage processing windows in the data pipelines themselves. Enterprise ACID-consistency data is usually processed in algebraic sets (eg; joins) and streaming ETL tooling can maintain high levels of precision around which time-series of data sets that the joins occur in.

The elimination of batch windows has long been a holy grail for data integration and we're now able to achieve this at a massive scale that's never been possible before.

Enabling Tech 6: Free MPP DAGs and Serverless

Two of the most monumental changes to IT since 2015 have been the rise of public clouds and big data technology. At core, both of these macro-trends have been about scaling the workloads of software applications. In the public cloud, it is not just about using somebody else's computers, it is also about adopting a 'serverless' computing model where you only pay for the precise amount of computing that you actually use. With big data technology, the shift away from Hadoop (map-reduce) and towards DAG (directed acyclic graph)-based MPP (massively parallel processing) frameworks (Apache Spark, Flink, etc.) put extremely scalable, highly interactive runtime engines into the hands of anyone that can download an Apache gzip package.

In the past, it was truly a monumental and expensive effort to run large scale workloads (using 100's or 1000's of processing cores). But now anyone with a credit card can get started in minutes, using big data frameworks on pay-

per-use serverless engines hosted in the cloud. This kind of radical simplicity has the net-effect of reducing 'data gravity' and increasing the propensity for experimentation with data. As the relative cost of storage and MPP compute continuously moves towards $0, the natural effect is that data will multiply and workloads will more frequently shift across different infrastructures.

Large enterprise IT organizations naturally have heterogenous infrastructure comprised of legacy systems and an array of newer SaaS cloud applications and multiple cloud platform providers. The rise of cloud computing has increased the fragmentation of data producers that at one time long ago may have lived within a small number of applications inside of a single data center. Those days are gone forever.

Widespread availability of cheap, simple MPP frameworks and serverless computing will drive unparalleled demand for a dynamic Data Fabric and a trusted Data Mesh like never before. These foundation capabilities will also provide the infrastructure for Data Fabric/Data Mesh to run workloads on-demand, inexpensively, and at massive scale.

Key Objectives and Tangible Benefits

This new kind of data architecture will empower faster innovation cycles and lower costs of operations. Anecdotal evidence[xiv] from early adopters indicate significant large-scale benefits that are possible today:

- **Total clarity into data's value chain** – through applied 'data product thinking' best practices
- **>99.999% operational data availability** – using microservices based data pipelines for replication
- **10x faster innovation cycles** – shifting away from ETL, to continuous transformation and loading (CTL)
- **~70% reduction in data engineering** – using no-code and self-serve data pipeline tooling

It is a rare opportunity to find a solution that can work equally to reduce costs associated with ongoing operations while also powering up innovation for the business units working to use data as a competitive advantage – this is one of those moments. By rethinking how data ought to be managed, and putting in a modern data fabric/mesh, the enterprise can more freely use the data for competitive benefits.

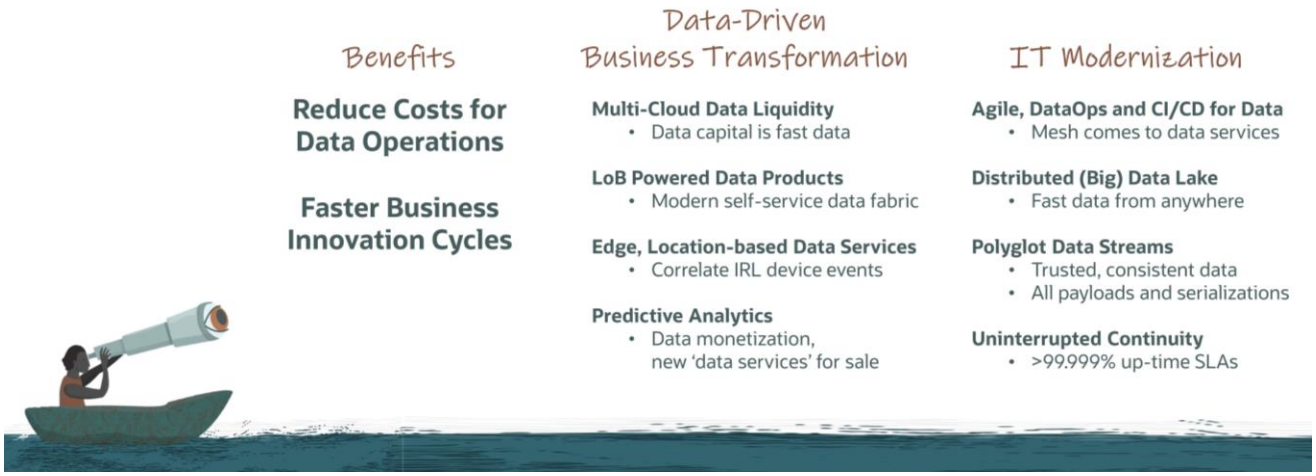


Figure 3: Dynamic Data Fabric and Trusted Data Mesh for tangible business outcomes

Organizations aiming to achieve meaningful business transformation will be focused on people and process changes, but technology also has a critical role to play. Alone and in vacuum, enterprise technology cannot affect widescale change, but when used as an instrument alongside cultural and organizational changes, technology can be a powerful catalyst to empower data-driven business transformations.

Organizations rarely have net-new budgets to spend indiscriminately on technology for technologies sake. That's why it is critical that the adoption of new approaches are grounded on a foundation of tactical benefits such as cost savings. Data Fabric and Data Mesh capabilities are first and foremost a path towards IT modernization, which can

directly lead to a smaller personnel footprint for maintaining operational data readiness and necessary data integrations, pipelines and governance across existing data assets. Costly data monoliths that require large support budgets and staffing can be phased out while modern, mesh-based integration platforms provide continuity SLAs, polyglot data pipeline capabilities, and a tangible path towards a distributed data lake.

While the data infrastructure is being transformed, the data-driven business transformation becomes more real. Faster and more reliable data flows mean more trustworthy predictive analytics and data monetization opportunities. For businesses that operate physical equipment at the edge in real life (IRL), data events that used to run only in specialized proprietary tools are now available in the general data estate.

The final phases of data-driven business transformation start to materialize when 'data product' ownership is at the line-of-business (LoB) but the smart data factories are operated by IT. IT will provide the data factory as a product to the LoBs, who will themselves create and monetize data products in self-service manner. This continuous flow of data and data products will happen seamlessly across multiple cloud ecosystems. No single cloud vendor will achieve 'lock-in' because the Dynamic Data Fabric and Trusted Data Mesh will empower enterprises to move data and data workloads to the clouds that make the most sense for the business.

Data Liquidity

Data liquidity is the ability to get data from its point of origin to its many points of use efficiently. The essence of data liquidity is reducing the time, cost, and effort to repurpose data for new uses.

The faster, easier, and cheaper a dataset can be repurposed for a new use, the greater its liquidity.

Each digital event gets created in a particular structure that, from the application's perspective, makes it fast and easy to capture. Think of these structures as different shapes. Some are born as a row in a table, others as a set of attribute-value pairs bundled into an object, others as nodes and edges in a graph, still others as a line of text appended to a list. But every operational data story, analytic or AI system, as well as other applications, need the data in a slightly different shape for their own purposes.

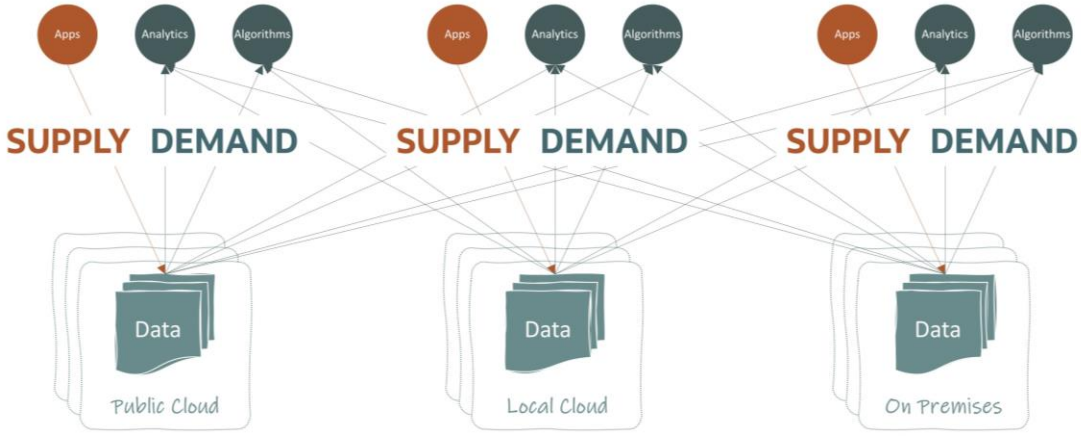


Figure 4: Data Liquidity as a measure of how quickly and easily the supply of data can match the demand

The challenge in increasing data liquidity is to turn your data into a shapeshifter. To allow the app to create data in the shape it needs while making it quickly and easily appear to other points of use that the data was created in a different shape that it needs for its own task. This is a challenge that will only get worse over time. The proverbial genie is out of the bottle and cannot be put back in – with the rise of SaaS and low-cost IaaS clouds, it is cheaper and easier than ever for enterprise data to live anywhere and go anytime. As enterprise data proliferates and decentralizes, tooling for Data Fabric and Data Mesh will become the only data management infrastructure that can retain some semblance of governance on it.

Thus, the Trusted Data Mesh and Dynamic Data Fabric become both a necessity and also a tool for innovation in the modern data architecture. For organizations who are principally worried about data governance, security and trusted data transactions, these Dynamic Data Fabrics are critical in preserving provenance and trust in the data pipelines that whisk data around the many physical locations that use it. For other organizations who are more interested in the use of data for innovation and monetization, data liquidity becomes a powerful enabler to repurpose data quickly while rapidly iterating on new business opportunities. What was once a theoretical agility advantage for IT will quickly become a powerful tactical weapon of competitive advantage in global markets.

New Paradigm for Data Architecture

In the previous two sections we examined how some aspects of data management thinking must be inverted and how several key technology enablers have brought us to this watershed moment in time. The intersection of our changing behaviors and changing technologies are empowering a new way of designing multi-cloud dynamic Data Fabrics and a trusted Data Mesh. These changes can be summarized as:

OLD WAYS OF THINKING TO INVERT	ENABLING TECHNOLOGY CHANGES
• Treating data as a by-product • OLTP → ETL → OLAP fragmentation • Waterfall process for data projects • Batch-first design thinking • Hub-and-spoke architecture bias • Storage centric modeling	• Service Mesh, Kubernetes etc. • Mature SDNs (software defined networks) • Microservices CDC/Replication capability • Replication for ACID/Polyglot in one topology • ETL in a Data Stream • Open Source MPP DAGs and Serverless Compute

Upon the backdrop of these behavioral and technology changes, a new approach becomes possible: an Enterprise Data Fabric that is dynamic and real-time, and a Data Mesh that works well with trusted enterprise data.

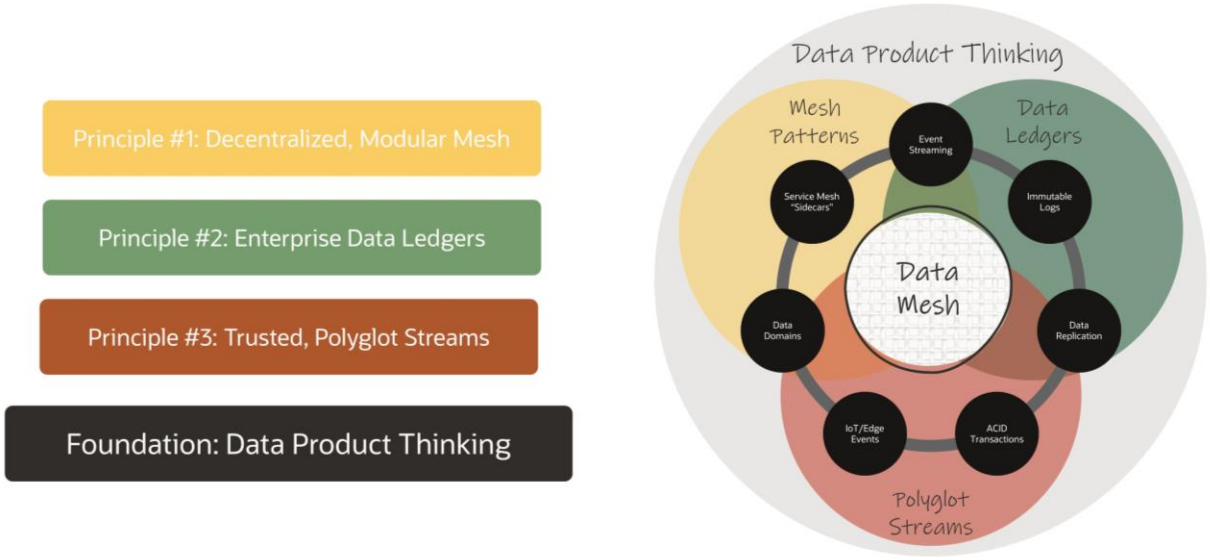


Figure 5: Key principles of a dynamic Data Fabric and a trusted Data Mesh

In this concept of a dynamic Data Fabric and trusted Data Mesh, there is a foundation 'way of thinking' and three technology principles. Data Product thinking is a foundational element because it is way of thinking that must imbue and permeate all aspects people, process, and technology that we bring to this solution domain. The first technology principle is that the frameworks and data services should be decentralized and highly modular by default. This decentralization is crucial to achieving CICD, Agile and sustainable multi-cloud data architecture. The second principle is that the data services should work with the data via data/event ledgers wherever possible. These ledgers allow us to

move forwards and backwards in 'data time' while also providing the highest possible fidelity into data events. The third principle is about supporting polyglot data streams, or streaming data that can take 'many shapes'. Enterprise data can have widely varying data formats and transaction semantics, so there may be many distributed ledger/streaming technologies that each specialize in a particular kind of polyglot streams. Taken together, data product thinking and these three technology principles provide a roadmap toward a new kind of data architecture.

Foundation: Data Product Thinking

The entire purpose of creating a Data Fabric or a Data Mesh is to bring more value to enterprise data – making it better, faster, cheaper to achieve business transformation objectives. A data product thinking approach is the most fundamental and foundational mindset shift that must be adopted. Without this mindset adjustment, it becomes far too easy to fall into the trap of adopting technology for technology's sake.

As regular consumers, each of us may buy products and services every day. From a morning coffee to much larger purchases like a computer or a car, we intrinsically understand the value of products in our daily lives – for the most part they are worth exactly what we are willing to pay for them. But what about data products, what does it mean for data to be a product?

Data Product Thinking is essentially the union of Design Thinking[xv] methodology and Data Product concepts that have sprung from the data science community. It defines a customer-centric approach for creating and curating data products and works to harmonize business and data consumer needs.

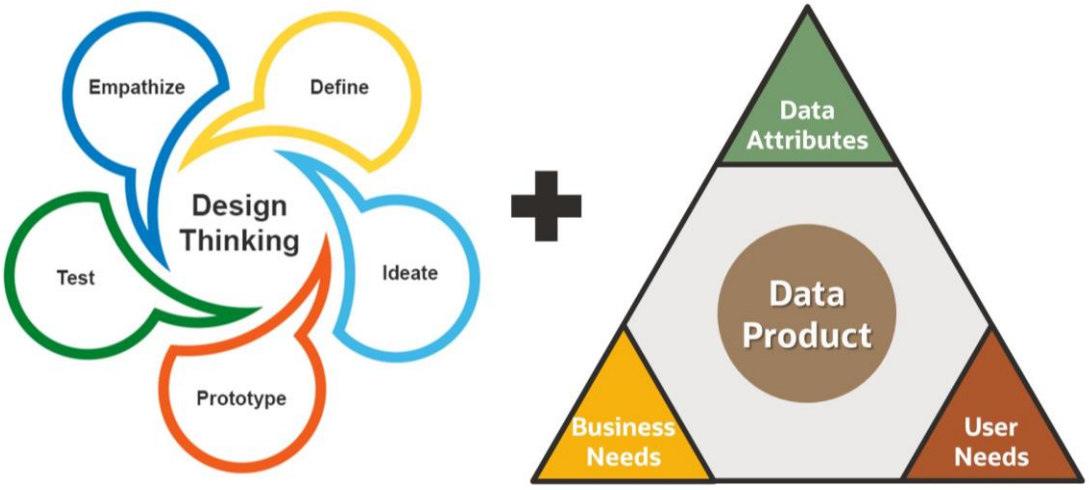


Figure 6: Value-focused Data Product Thinking is influenced by Design Thinking methodology and Data Product concepts from data science

Once we fully embrace data as an asset with explicit and implicit value, it becomes more natural to align data to KPIs (key performance indicators) and SLAs (service level agreements). We will no longer look at data simply as a byproduct of application workloads or as something that IT should relegate to the dustbins of cheap tape backups in a far-away bunker. Data product thinking means treating data as a proper asset of value and bringing a product management discipline to data management. Later in this document we will explore the top four attributes of managing data as a product.

Principle #1: Decentralized, Modular Mesh

Core to notion of a dynamic Data Fabric and any Data Mesh is a decentralized physical architecture of the data services. Producers of data can be anywhere, from the edge, the edge gateways, on-premise data centers, public cloud networks and SaaS applications. The essence of a dynamic Data Fabric means working with data events in real-time and therefore the Fabric/Mesh needs must be capable of existence in any part of the networks where data is

born. Like any de-centralized mesh (eg; a WiFi mesh), there is a controller (aka; control plane) but in a mesh architecture any node in the mesh can talk to and relay data to any other node in the mesh. Therefore, there is no need for centralized/hub-based routing. If you have a WiFi mesh in your home, that is how it works.

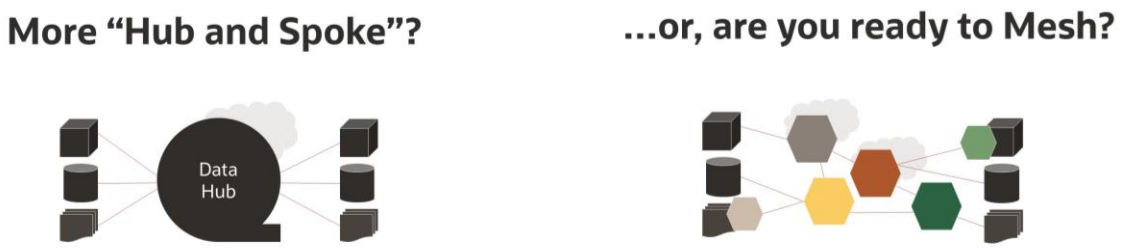


Figure 7: Conceptual topology difference between a Hub and a Mesh

This inherent ‘mesh’iness’ is instrumental to efficient data operations in multi-cloud ecosystem where streaming data can be routed via many different pathways without having to funnel through a singular, centralized hub every time.

WHY ISN’T THE MESH A MESH OF DATA ITSELF?

A successful enterprise data architecture will work with the data as it is, rather than how we wish it might be. The real-world truth about enterprise data is that there are decades worth of data formats in most typical large enterprises that just aren’t going away anytime soon. An IT philosophy of, ‘if it isn’t broken, then don’t fix it’ is widespread – meaning that legacy data is merely a fact of life.

Most data either have no schema or are completely de-normalized; and precious few data formats are self-describing or self-linking. Certain expressive data standards (such as the Resource Description Framework (RDF) or Web Ontology Language (OWL)) which are URI-linkable at Web scale are simply not adopted widely enough or in consistent enough ways to act as a unifying format.

Similarly, more than 40+ years of mostly unsuccessful industry vocabulary standards have essentially proven that vendors and enterprises alike usually prefer to have their own lingua franca, rather than conform to others. In some industries the underlying data models are considered core intellectual property (IP) to be protected and not to be shared.

Thus, the Data Fabric and Data Mesh are not some kind of meta-graph of data itself, they are a Fabric/Mesh of essential services (eg; data transformation, streaming, catalogs etc) which provide the capabilities to effectively move and manage any kind of data.

Principle #2: Enterprise Data Ledgers

The word ‘dynamic’ in the context of a dynamic Data Fabric is referring to the event-driven nature of this architecture approach. With data events as a central core feature of the architecture, there must be a ledger to keep track of the log events, messages and the position of the various readers who consume the events.

There is a decades-long history of data ledgers in enterprise software systems. In the 1960’s dedicated Transaction Processing Systems (TPSs) were created, TPSs are essentially an event ledger for application and data transactions. TPSs such as IBM’s CICS and Oracle Tuxedo continue to operate even now. Nearly all SQL databases are built around an internal online redo log, acting as a transaction ledger for strong consistency data events. Other more recent enterprise ledgers may be based on Apache Kafka, Pulsar or even blockchain technology such as Hyperledger.

Ledger based platforms are often designed to cover specialty use cases such as:

- **Application Architectures** – rise in popularity for microservices-based ledger solutions utilizing the event sourcing pattern, which depends on an ‘event store’ for cross-component communications
- **Infrastructure Log Ingestion** – event ledger for collecting telemetry (operational behavior) of enterprise infrastructure, tools like Apache Kafka, Pulsar, and AWS Kinesis etc are common examples
- **Enterprise-wide Data Services** – apart from simple log ingestion, the same ledgers such as Kafka and Pulsar are also being used for event-based data services more broadly in enterprise use cases
- **Multi-party Blockchain Ledger** – adopted for its transparency and immutability, blockchain technologies are being used for cross-organization (eg; B2B) exchanges to track transactions in a highly trusted way

- **Distributed Database Transactions** – globally distributed database transactions (eg; for 'sharded data' or for high availability) will replicate events using Data Replication tools like Oracle GoldenGate

This last category of Distributed Database Transactions is an incredibly important aspect for enterprise use cases. Since the vast majority of enterprise applications run on highly consistent relational databases, any shift towards decentralized data sharing necessitates ACID-capable transactions that maintain relational consistency across network boundaries. Tools such as Oracle GoldenGate are actually distributed transaction ledgers, built specifically to guarantee trusted, consistent data even as it may be replicated across thousands of kilometers and across polyglot data stores. This requirement ACID level consistency is noted in the research around OLEP systems.

Online Event Processing (OLEP)

The concept of a general-purpose Online Event Processing (OLEP) system for data-centric enterprise domains was initially explored in 2013 by one of Kafka's inventors Jay Kreps in a seminal paper, "The Log: What every software engineer should know about real-time data's unifying abstraction"[xvi]. This paper provides a foundation for thinking about an enterprise event ledger that is capable of capturing log events from a wide variety of enterprise data sources and making those events easily available to any data consumer. More recently, Martin Kleppmann et al. have attempted to push this concept further ("Online Event Processing: Achieving consistency where distributed transactions have failed"[xvii]), calling for ACID-like data consistency as part of a distributed OLEP system. The central concept in this line of thinking is to leverage event stores or messaging systems more broadly for enterprise-wide solutions, rather than just for tactical, point-solutions.

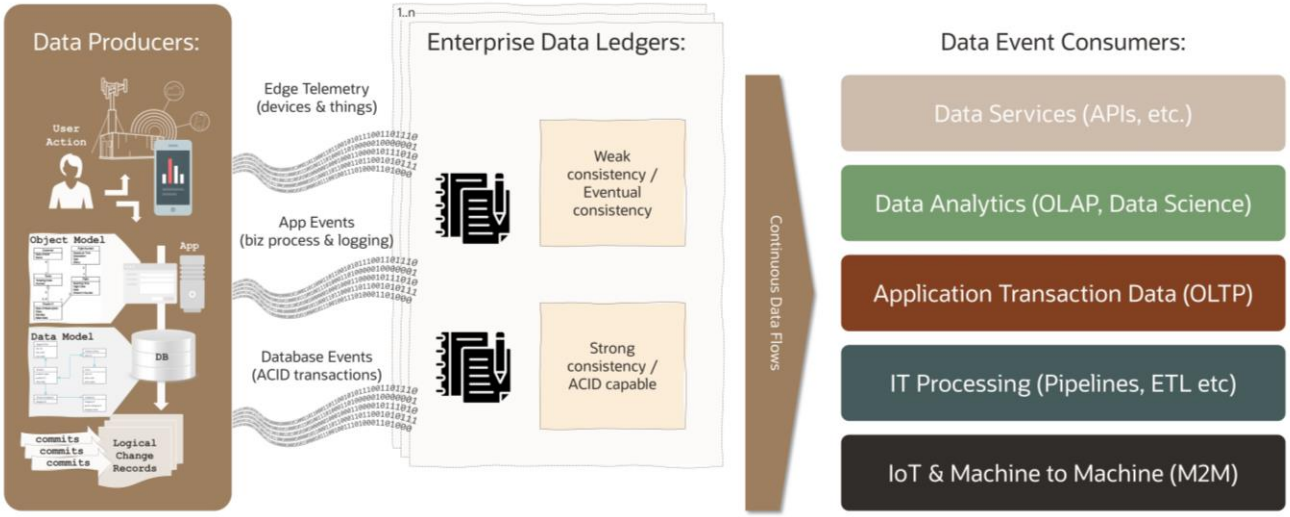


Figure 8: Enterprise event ledgers as a unifying source for data events of all kinds

An enterprise-wide capability for event ledgers does not necessarily imply that there needs to be only one ledger for an entire enterprise. The enterprise nature of the event ledger implies a set of enterprise class capabilities such as an ability to scale for very large amounts of events, an ability to cover a flexible set of transaction semantics, payload syntax, and serverless options when running in a public cloud or as part of a service mesh (eg; Kubernetes, Open Shift etc). In fact, in a dynamic Data Fabric or a trusted Data Mesh there are likely to be many different enterprise event ledgers, perhaps segmented by unique data domains or operated by different organizational business units. Shared catalogs or registries would serve to align and govern data across multiple ledgers.

Principle #3: Trusted, Polyglot Data Streams

The central feature of all data integration tools (eg; data replication, ETL engines, data federation etc) has always been grounded in a foundation of trust for database transactions. This trust in maintaining data consistency is a crucial area that has separated the $3+ billion-dollar enterprise data integration tools market[xviii] from other integration styles. Data Fabrics are the modern next-generation of data integration tools, so the foundation capabilities of trusted data

are an inherent part of Data Fabric. But a Data Mesh can exist distinct from a Data Fabric and therefore it is quite possible to build a Data Mesh without data consistency guarantees. In fact, some microservices-centered concepts of a Data Mesh place the trust and consistency of data into the hands of the application developer (to encode multi-phased commits, compensating transactions, and recovery logic) despite the risks that come with this approach.

The 'trusted Data Mesh' concept in this document is rooted on the principle of using reliable tooling that can provide data guarantees when necessary (for ACID-property data sets) and yet remain flexible enough to work equally well with data formats and transaction types that do not need to maintain very high levels of trust.

DATA TRUST CHARACTERISTICS	
Exact-once message processing	In a distributed system (messaging or databases), the computers that make up the system can fail independently of each other. Exact-once processing guarantees that if a message producer tries sending a message many times, it leads to the message being delivered exactly once to the end consumer.
Strict ordering of events	Application producers often persist data in 3rd normal form data structures. When data is saved, the precise ordering of the transactions are extremely important to the consistency of the data. A trivial example is ItemHeader and ItemDetail tables, with a Foreign-Key property between them the sequence of an update can result in consistency/commit failure if performed out of sequence.
Transaction atomicity (A)	As with the ordering of events, data transactions are often grouped together for a common "commit point". This allows for the database system to guarantee the atomicity of the transaction as a whole, even though the commits may consist of many different SQL statements. As atomic transactions are converted to other payloads and placed in messaging systems (aka: the enterprise event ledger) the systems must be capable of preserving the holistic atomicity of the transactions as they move through distributed systems.
Transaction consistency (C)	For purposes of this document our focus is on the consistency of the transaction itself when the data requires referential integrity. A trusted Data Mesh must be capable of preserving the consistency guarantees of relational payloads as they flow through the mesh. Row level and system level consistency of the transactions (possibly as large groups of SQL) should be preserved and guaranteed.
Isolation (I) and durability (D)	Concurrent transactions are guaranteed to be processed as if there is sequential isolation, in order to preserve the consistent state of the system even while incomplete transactions are in-flight. For maximum durability a Fabric or Mesh should provide near-zero RPO (recovery point objective). Durability is essentially a measure of acceptable data loss.
Rollbacks and recovery	Should any of the above trust types be violated, or if there is some sort of unplanned disaster within a partition or consumer, then the Fabric/Mesh frameworks should be capable of restoring the data to a consistent state.

Data integration tools and the Data Fabric are usually assumed to be trustworthy when it comes to working with ACID-type data workloads. But in a dynamic (distributed, streaming) Data Fabric or an Event Mesh, the maintenance of data trust becomes much more complicated due to the de-centralized architecture and the massive numbers of low-latency events being routed on the networks. Not all types of Data Mesh will preserve trusted, consistent data.

Therefore, a 'trusted polyglot data' Mesh means that the data management systems can provide a breadth of data trust characteristics that include exactly-once, ordered processing, atomicity, consistency, isolation, durability and recoverability. These trusted characteristics must cover the full pipeline of data processing starting with capturing events on the data producers, through any replication/messaging sub-systems, and onward through to the data processing pipelines where windowing, aggregation and data transformations may occur.

Dynamic Data Fabric, Deployed as a Trusted Mesh

This document's focus is on a particular type of Data Fabric – a dynamic, real-time and streaming centric Data Fabric. Likewise, the document focus aligns with a particular kind of Data Mesh – a trusted, strongly consistent enterprise Data Mesh. These areas of distinction spotlight Oracle's approach to Data Fabric as being more dynamic than most, and Oracle's approach to Data Mesh as being more trusted than others. A Data Mesh can exist outside a Data Fabric, (for example as part of a specialized IoT application) and similarly, a Data Fabric need not be real-time or dynamic.

For organizations that strive to achieve these attributes of a dynamic Data Fabric and a trusted Data Mesh together, there is a journey required to move from monolith to mesh:

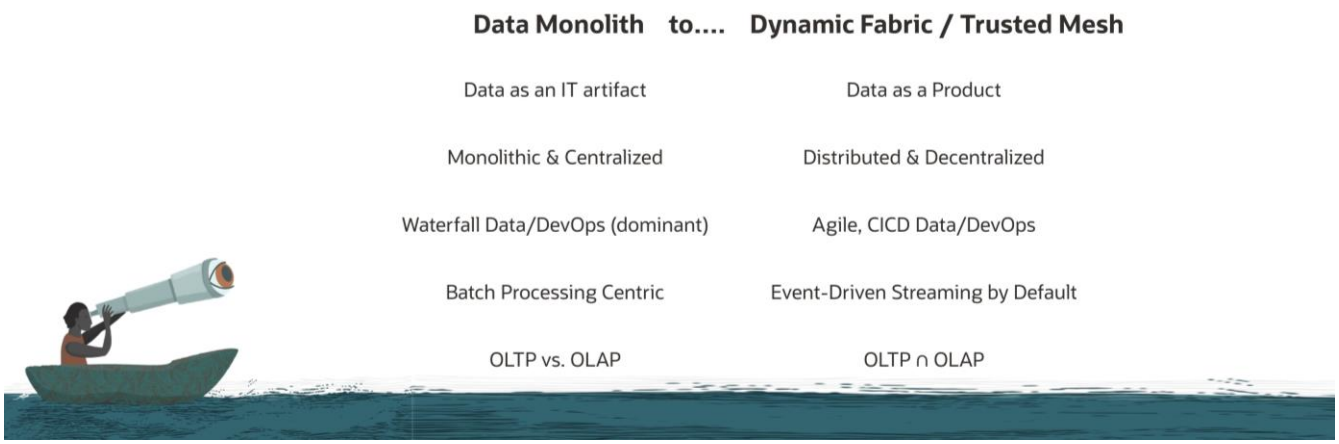


Figure 9: Benefits and opportunities of applying a dynamic Data Fabric and a trusted Data Mesh

Bold claims are made about the business value of Data Fabric and Data Mesh. Achieving the goals are not simple, there is no magical solution. There are many facets to adopting a holistic approach of dynamic Data Fabrics and trusted Data Mesh. For the remainder of this section, we will take a systematic look at key attributes of a dynamic Data Fabric and a trusted Data Mesh, including:

AGENDA FOR REMAINDER OF THIS SECTION:

- **Data Product Thinking** - as a holistic mindset towards managing data 'as a product'
- **Aligning Operational and Analytic Data Stores** – the importance of a data management approach that encompasses the whole enterprise data estate
- **Decomposition of the Data Monolith** – an introduction to decomposition strategies
- **Data Zones and Data Domains** – understanding the difference between a domain and a zone, within a distributed data architecture
- **What Makes a Mesh** – identify the essential characteristics of a decentralized mesh of services and data
- **Key Personas, Role of Self-Service** – exploring the people aspects of who uses tools of the Mesh/Fabric and what kinds of self-service can be expected
- **Event-driven, Streaming Data Pipelines** – how real-time data transformation affects business value
- **Repeatable Data Product Factories** – understanding why a focus on the Fabric/Mesh of data product production is so important to business value
- **DevOps, CI/CD and DataOps** – the relationship between operating the Fabric vs. users of the Fabric
- **Governance in a Data Mesh** – key features and capabilities that must be present in an advanced Fabric/Mesh

Data Product Thinking

The roots of data product thinking stem from an intersection of concepts familiar to Designers and Data Scientists. On the design side, there is a well-proven approach to user experience called Design Thinking[xix]. Design Thinking is a highly studied and repeatable methodology for producing innovations around products and services, it has been used by 1000's of organizations globally on everything from designing websites, applications and the physical products you might by at the store. On the data science side, the concept of a Data Product has regularly been used to

describe digital products fueled by data and machine learning. Most famously, a Harvard Business Review story in 2018 aimed to explain to a wide business audience, "How to Build Great Data Products."[xx]

In 2020, "Product Thinking 101"[xxi] was written; this paper is in many ways is a superset framing of the goals espoused in Design Thinking practice areas. Product Thinking applies many core Design Thinking methods but in the slightly wider context of a market domain, as product planning for a larger group of market consumers. In contrast, the Design Thinking methods (ideation, human-centered design, etc) often tie in very directly to individual user needs, goals, and solutions. In this way, Design Thinking is broadly complementary to Product Thinking.

Both Design Thinking and Product Thinking lean heavily on Clayton Christensen's concept of "job to be done" (JTBD)[xxii]. JTBD theory is an over-arching approach to customer-centric product design, for example the essence of this concept is well said as, "*When we buy a product, we essentially 'hire' something to get a job done. If it does the job well, when we are confronted with the same job, we hire that same product again. And if the product does a crummy job, we 'fire' it and look around for something else we might hire to solve the problem*"[xxiii]. This kind of product-centric thinking is an elemental aspect of data product thinking. For example, when we aim to design 'data products' we must continuously assess what job the data is fulfilling and how well it is performing at that job.

Central framing of the Data Product idea comes from the data science community. As initially conceived in the data science domain, a data product is most commonly defined as, "*the output of any data science activity.*" However, in the wider context of data product thinking, data products can also be thought of as, "*any governed data set, with KPIs and SLAs, that directly facilitate a business outcome.*" This document uses that broader definition, placing emphasis on KPI and SLA governance for productization goals. Representative examples data products can include:

REPRESENTATIVE KINDS OF DATA PRODUCTS:	
Analytics	• Reports and dashboards that drive outcomes and business decisions • May include data science regressions, historic reporting or real-time insights on data in-motion
Data Models	• Data domain objects (component models, perhaps serialized or represented as UML diagrams) • Data models (relational, graph, ontology, etc) and ML features (eg; columns/attributes etc)
Algorithms	• Business rules, as captured in design documents, data catalogs, RETE engines, ETL mappings etc. • Machine learning (ML) models, productionized or as captured in ML lifecycle catalogs • Data pipelines, including batch ETLs, ESB document transformations, real-time streaming integrations, etc.
Data Services / APIs	• Data payloads (serialized formats such as XML, JSON, Avro, Protobuf, proprietary tool formats etc) • API Management: REST APIs, including Pub/Sub semantics for message-based subscriptions etc. • Design-by-Contract based API frameworks, where API is the product to be consumed (by developers)

Importantly, not all business artifacts from the lists above would necessarily qualify as data products. An essential characteristic of a data product is that it '*…facilitate a business outcome…*' and not all data-centric artifacts directly facilitate outcomes. As a simple example, in a collection of dashboards and analytic reports, only a few of them may regularly and repeatedly used to drive business decisions. There is a lot of enterprise data out there, and most of it will not need to be managed and governed as a full-fledged data product.

Data Product Managers

Just as the term 'data product' was born from the data science community, so has the concept of a Data Product Manager. In the data science realm, it quickly became necessary to have skilled people on the team who could facilitate the intersections between business owners, the engineers and the data scientists. Like other engineering and product organizations, the data science teams needed these individuals to possess a strong mix of program management skills with the know-how for running product lifecycles and driving business outcomes.

The concept of a Data Product Manager applies to a broad set of potential data products. Data Product Managers bring a high level of data-literacy to ordinary product management skillsets. Like regular product managers, Data Product Managers apply Design Thinking/Product Thinking, develop business cases, create strategies, manage lifecycle of products, build roadmaps, present release plans, compile backlogs and work with a wide set of stakeholders. Additionally, Data Product Managers are skilled with and knowledgeable about the organization's data producers, data formats and meaning of data sets, as well as the kind of algorithms and visualizations that can be applied to data for downstream consumption by the consumers of data products.

Figure 10: Role of a Data Product Manager in relation to traditional Product Managers

As noted by the Rapido Labs, "*if a traditional Product Manager operates at the intersection of business, engineering, and user experience, a Data Product Manager sits at the nexus of data, engineering and business.*"[xxiv]

KPIs and SLAs for Data Products

Like any product, data products must be planned, managed and governed for a complete lifecycle. Data products will have Key Performance Indicators (KPIs) and Service Level Agreements (SLAs) that should be measured and adhered to. Each instantiation of a data product will naturally have its own unique attributes to be governed, but some example attributes of a data product include:

ATTRIBUTES OF A DATA PRODUCT	
Packaging, Documentation	• How is it consumed? Is it fully documented?
Purpose and Value	• Is the intended use well defined? • Is it value explicit or implicit? • Does it depreciate over time? at what rate?
Quality, Consistency, Service Levels	• Are KPIs explicit? What about the SLAs? • Will different consumers of the data have similar experiences? • Will consumers at different times (daily, weekly, monthly) experience it the same way?
Provenance, Lifecycle and Governance	• Fully managed from cradle-to-grave, repeatable lifecycle policies • Can we trace the origins of the data? • Is the derived data explainable? (eg; the output of algorithms)

The Data Mesh concept has become deeply intertwined with data product thinking. Although much of the origin of data product ideals sprang from the data science community, the term has been broadened to include a larger set of potential data products spanning the analytics, data services and conventional data modeling domains. Increasingly

there is demand for a new kind of data-centric product manager, as individuals who are deeply knowledgeable about their data domains and also the data formats and tools required to effectively work with that data. These data product managers have a unique task to curate and package high-value data for broad consumption across internal and external users. Likewise, for organizations that measure and track the value of their data, it will be the data product managers who manage KPIs and SLAs across the full end-to-end lifecycle of data.

Aligning Operational and Analytic Data Stores

Since the earliest days of computing, we have organized digitized data differently according to different workload needs and characteristics. Online Transaction Processing (OLTP) tools have generally been where we work with operational data – the applications that run the business and handle all varieties of transactions (insert, updates, deletes, etc). The Online Analytic Processing (OLAP) are optimized for complex read operations that may join lots of different data and run on very large data sets. OLAP systems are most closely associated to multi-dimensional data models, but in this document we are looking at Analytic data stores more generally - to include the wide variety of data structures used to support read-optimized analytics.

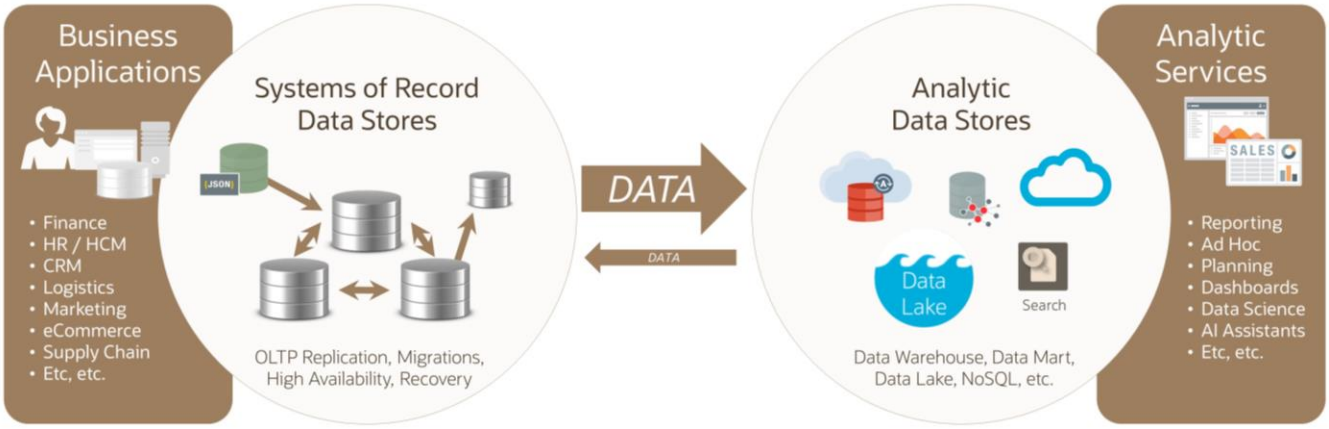


Figure 11: Traditional separation between OLTP and OLAP data domains

As depicted above, the OLTP and OLAP data domains have historically remained segregated with only very loose coupling between them. In most modern enterprise organizations, the operations team running the OLTP systems is typically in an entirely different IT organization from the engineers that design and operate the Analytic systems. Often times the only thing joining these data domains together are collection of ETL jobs and stored procedures.

Apart from a few niche vendors that claim to be able to run OLTP and OLAP workloads together on the exact same data store, a vast majority of the world's IT systems continue separate OLTP and OLAP workloads into distinct, optimized data stores. Likewise, while some distributed software applications are designed to support 'eventual consistency' data models, these kinds of applications remain a very small minority in practice. Workload optimized, strong-consistency data stores remain the basis for modern software applications. Converged database systems, such as Oracle Database, can run polyglot optimized data stores as different pluggable data store instances within the same overall DBMS. Many enterprises still create physical and network separation between the data. As such, the alignment between OLTP and OLAP data must be accomplished using data integration.

The old tried-and-true way of doing data integration has been with ETL hub-and-spoke engines. However, this type of batch-oriented (scheduler driven) data pipeline just exacerbates the split between OLTP and OLAP data sets. With this kind of deep separation of data, it becomes extraordinarily complex to align the data domains and provide rich data governance across the whole estate of data.

A dynamic Data Fabric and a trusted Data Mesh provides an alternate path: real-time streaming integration between OLTP and Analytic data domains. This can be simpler than you might imagine.

For many decades logical replication has been utilized as a solution set for distributed OLTP transactions. Tools like Oracle GoldenGate can replicate data transactions via wide area networks (WANs) while maintaining ACID properties of high-value data. In fact, Oracle GoldenGate's core use case remains high availability (HA), disaster recovery (DR) and it is the 'platinum tier' of Oracle's Maximum Availability Architecture (MAA)[xxv]. Thus, the foundation capabilities for trusted real-time transactions are already well established in the OLTP stack. Later in this document we will describe how to apply Data Mesh technology to the polyglot data sets often found in Analytic domains.

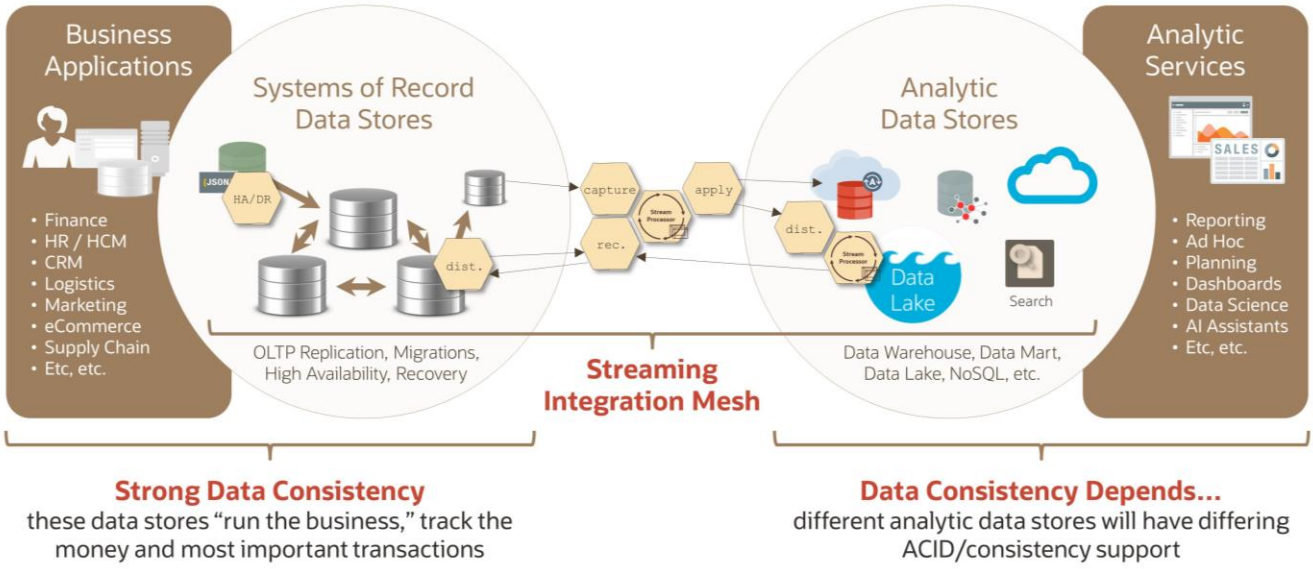


Figure 12: An integration mesh that can span the semantics of OLTP and OLAP data domains

As shown above, a streaming integration mesh can be applied to align OLTP and Analytic domains. Like the human body's circulatory system, a streaming integration mesh can continuously flow data in-motion to the OLTP and OLAP systems that need it. This is not a one-way flow, data events can continuously flow in many directions. Data flows can support polyglot data streams (relational or document payloads) while preserving the 'trust characteristics' described in Principle #3, Trusted Polyglot Data. These trust characteristics are absolutely essential for maintaining confidence that the meanings of the OLTP transactions are accurately conveyed into the Analytic data stores.

In summary, real-time streaming integration is what makes a Data Fabric 'dynamic' (as opposed to scheduler-driven batch processing) and support for high-consistency ACID transactions is what makes a Data Mesh 'trusted' (as opposed to data movement without guarantees). This real-time mesh can be used to keep data aligned and in sync between OLTP and Analytic data stores.

Enterprise Event Ledger

Earlier in this document, Principle #2: Enterprise Event Ledger noted how Jay Kreps are Martin Kleppmann are two of the central figures driving this concept of an enterprise event ledger. This is a big part of what has become a data architecture revolution around streaming data. At the heart of this new way of thinking about enterprise data is the concept that nearly all enterprise data platforms can be understood through their logs, that their real-time event logs are truly at the heart of understanding and working with business data.

Consider what it takes to understand the meaning of enterprise data. For most of the past 50 years of computing there have been two schools of thought on the meaning of computer transactions: one view held that the application business objects (in memory serializations) are the source of truth while the other view maintained that it is the data storage layer (physical storage) which held the ultimate source of truth. However, this new enterprise event ledger concept adds the time dimension and offers a third way – to really understand the semantics of enterprise data we must look at the sequence of event logs of these systems.

As a metaphor, consider a chess match and what information is needed to understand the match holistically. We cannot simply look at the chess board because that only tells us the position of the pieces at a point in time. Likewise, we cannot understand the chess match just by listening to the conversation that happens between the two players. To understand the full narrative of the truth we must understand the positions, player conversations, reasoning of the players, and we need to consider the full sequence of moves for the full duration of the match.

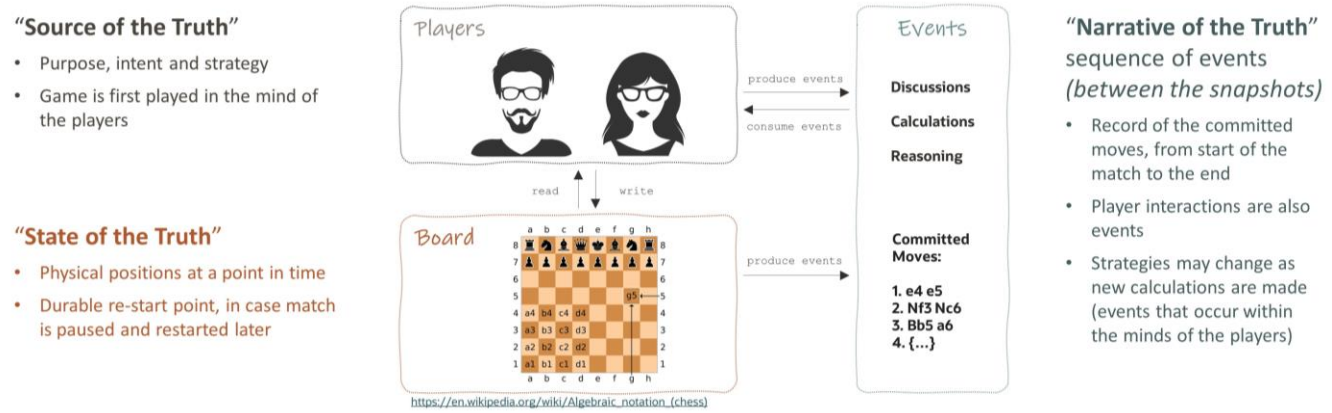


Figure 13: Metaphor, understanding a chess match

Similarly, to understand the meaning of enterprise data as it flows, we need to understand (i) the application source of truth through the log events of the application software, (ii) the data store log events as data's state changes, and (iii) we need to have a history of these log events over time so that we can situationally understand the data in any point in time. Done well, the power of this approach is that it allows us to understand the story of data as a dynamically changing narrative. The dynamic narrative of data is what provides us the context necessary to understand meaning. Without the dynamic narrative, we only see records and schema at a particular snapshot of time.

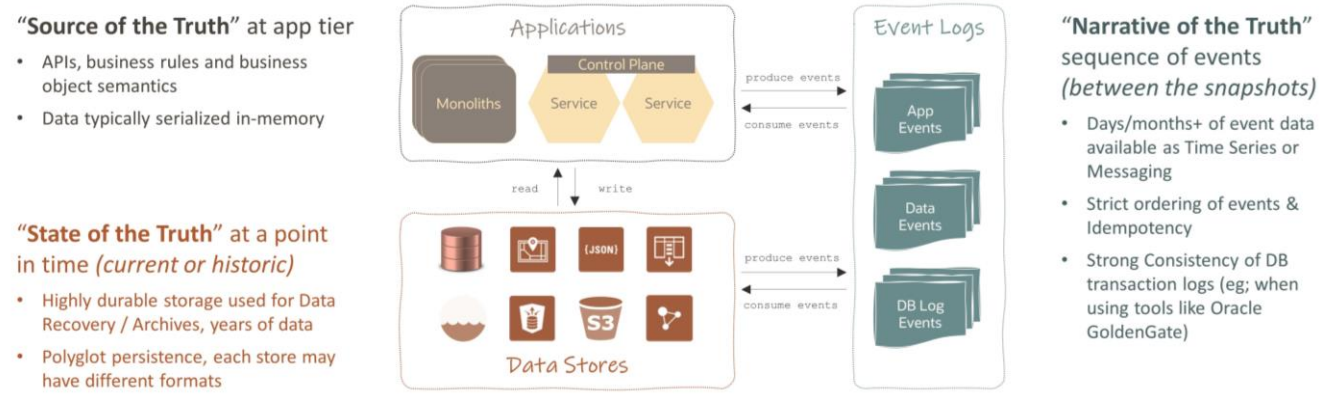


Figure 14: Enterprise event ledger as a narrative of the truth for enterprise systems

Business applications remain the ultimate source of truth for data since they embody the business logic and primary interfaces to the human users. The underlying durable data stores are the ultimate state of the truth since they are most typically the position of the data when used for high availability (HA) and disaster recovery (DR). But to fully understand the meaning of data in its holistic and fullest context, we need access to the story that the data is telling us, and this story comes from the narrative of the logs. This narrative of the truth is the sequence of events collected from the devices, applications, data stores etc. taken together as a whole.

PRESCRIPTIVELY, WHAT LEDGER TECHNOLOGY IS BEST?

There is no single best technology for an enterprise event ledger. As described previously, individual use cases for the event ledger concept are wide-ranging and could include a focus on event-sourcing for microservices, log collection for analytics or as general-purpose data services, among others. Accordingly, the exact technology selection for a given use case should be tailored to the business needs. For example, event ledgers could be crafted from technologies such as:

- **Microservices Event Stores – that include native features that align to common microservices patterns such as CQRS**
- **Time Series Databases – which are optimized for high-volume writes, typically from IoT devices**
- **Message Queues or Service Bus – proven queues for business process transactions and structured data payloads**
- **Event Streaming Platforms – such as Apache Kafka, Pulsar and various proprietary cloud messaging services**
- **Data Replication Tools – which are natively integrated with database event logs for maximum data consistency**
- **Blockchain – especially useful for multi-party event ledgers, where immutability and transparency are mandatory**

For general purpose, enterprise-wide solutions a combination of technologies may provide the best-fit. For example, combining a replication tool like Oracle GoldenGate (providing strong data consistency and data recoverability) with open-source framework like Apache Kafka (for horizontal scale-out and rich polyglot data support) is a common approach used by large enterprise IT organizations. For enterprises that use many different ledgers, a data catalog or schema registry may be used to govern data domains that span different ledgers or data zones.

The enterprise event ledger is not magical idea, but it does empower significant new capabilities that may seem far beyond what was previously possible using conventional integration techniques. First, the ability to move backwards in time at a very fine-grained level (eg; nano/second) was not really possible using older architectures. Second, the fidelity of data has never been higher (eg; seeing and working with user events, event-by-event) which is absolutely crucial for high-end data science initiatives. Finally, to be able to correlate polyglot events (unstructured and structured logs) across broadly distributed architectures in real-time (millisecond latencies) is a game-changer.

Decomposition of the Data Monolith

Functional decomposition has its roots in mathematics, as the process of taking complex ideas and deconstructing them into component parts[xxvi]. Similarly, in the realm of software, decomposition is the activity taking complex real-world domains and breaking the concepts into simpler parts that can be modeled in computer systems[xxvii]. For new applications, decomposition is usually undertaken at the design phase of a project. Techniques for object-oriented decomposition are widely understood and a foundation for most modern programming languages[xxviii].

In the context of decomposing monoliths, the idea originates with the microservices programming community as a method for refactoring legacy systems (typically monolithic software architectures) into smaller more modular parts. Much has been written about how to refactor legacy monoliths, and the topic is still considered more of an art than a science. Chris Richardson is a recognized leader in the microservices community and his talk[xxix] at an Oracle Code event in 2019 is considered a canonical resource on how to do functional decomposition of monolithic applications.

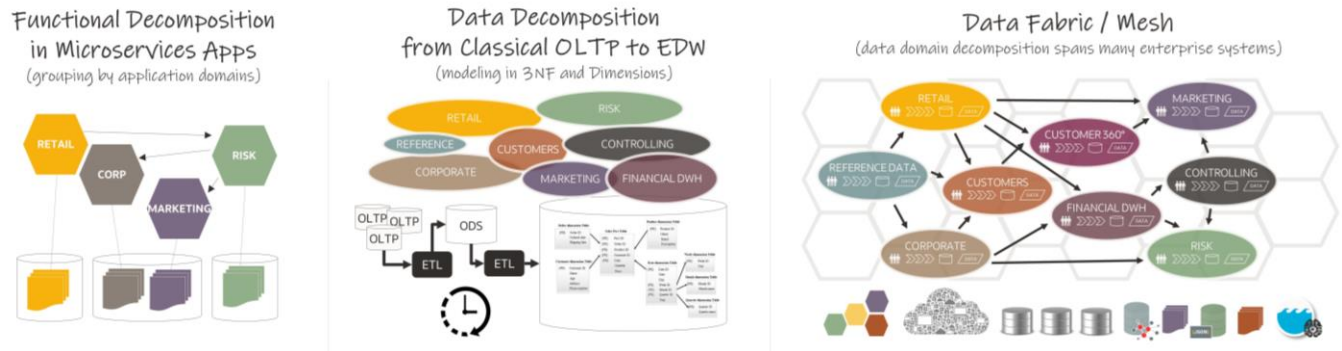


Figure 15: Contrasting different styles of decomposition

As part of a Data Mesh or a Data Fabric, the process of decomposition is more focused on the data domains. Functional decomposition of data domains is rooted in very longstanding traditions. Dating back to the early 1970's the methods and techniques for modeling data in third normal form (3NF)[xxx] are still by far the most common approach to data modeling inside of databases today. There are volumes of work on how to perform business domain decomposition into 3NF models. With 3NF focused on OLTP use cases, the corollary OLAP data models are typically modeled as multi-dimensional (aka, star schema) models. Again, looking back to the 1980's and the rise of the

enterprise data warehouse, there are a myriad of approaches (eg; Kimball, Data Vault, etc) for applying functional decomposition of data domains for read-optimized analytic data.

Decomposition for Data Fabric and Data Mesh are fundamentally different in scope because they are about decentralized data, produced by many different systems, packaged as many different data products, and consumed by many different consumers. A Fabric or a Mesh is not about a singleton application or data store modeled as a materialized entity. Instead, the data entities flowing via a Mesh or a Fabric can span many underlying systems and take on multiple forms, syntax, or shapes depending on the payloads or durable stores that materialize the data.

Another area where functional decomposition in a Fabric or a Mesh is different is in modeling behaviors. In application components the behaviors exist behind APIs and can be modeled with state-transition diagrams. In OLTP databases or data warehouses, behaviors are occasionally modeled as stored procedures resident within a database. Behaviors in a Fabric or a Mesh are typically going to be embodied by data pipelines. Data pipelines are implemented using a wide variety of technologies, but they are logically focused on concrete actions such as capturing events, moving and transforming data, then loading data into an event store or directly into a sink (target technology).

As noted in Principle #2: Enterprise Event Ledger, an enterprise may use many different event ledgers. Since data objects may span many different global systems, the related events and payloads may reside in multiple topics, queues or replication paths. Thus, these event ledgers (and their representations such as partitions, registries, topics and queues) should also be decomposed and managed alongside payloads, data models and pipelines.

From a pragmatic perspective, there are varying degrees of decomposition and refactoring that might be done. Depending on individual circumstances it could make sense to focus only on logical decomposition (ie; using enterprise data catalog tools) which provide benefits derived from organizing existing artifacts into categories of well-governed domain tags, taxonomy or ontologies. Alternately, a more extensive refactoring could be done to tear down and rewrite legacy monoliths such as applications, ETL tools, data warehouses. This more intensive refactoring could accomplish more aggressive business transformation objectives such as improving IT agility, reducing latency of data services, and improving the timeliness of data products and insights.

Data Domains and Data Zones

A data domain is a logical category defined by an explicit business problem area; these data domain categories are typically used to group data artifacts that belong together. In contrast, a data zone usually refers to a physical location where data may be collected and processed according to a particular lifecycle stage of the data.

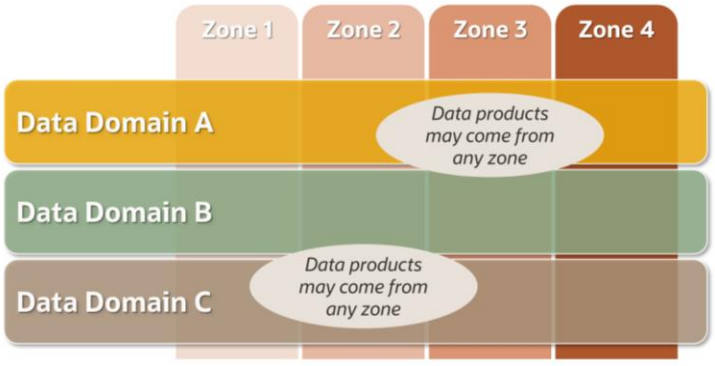


Figure 16: Data domains and data zones

Data domains may span multiple data zones. For example, a retailer may have multiple data domains such as inventory and purchasing, but the data entities in these domains may span multiple processing zones for transient data, raw data, prepared data, and master data. Each of the processing zones will usually correspond to a different physical location in the IT infrastructure and may be associated with a different phase of the data lifecycle or quality levels. Data domains (such as inventory or purchasing) are logical groupings that are used to organize and group data

of the same type, even when the formats of the data may be serialized differently (eg: Java aggregates, JSON/Avro payloads, Parquet or relational formats for analytics). Despite all the syntax and serialization changes across data zones the meaning of the data still belongs to the domain.

WHAT ABOUT DOMAIN DRIVEN DESIGN (DDD) CONCEPTS?

Within the microservices community few subjects are as polarizing as Domain Driven Design (DDD). In 2003, DDD was initially conceived as a collection of patterns for object-oriented application development in Eric Evan's book, "Domain Driven Design"[xxxi]. Later, in 2013 Vaughn Vernon's book "Implementing Domain Driven Design"[xxxii] updated DDD concepts based on years of hands-on implementations. Today, thought leaders in microservices still regularly debate the practices of DDD, and implementing the many DDD patterns in a prescriptive and repeatable way still befuddles many developers.

Focus of DDD remains definitively within the application development, object-oriented and microservices developer communities. There are many useful ideas and methods codified within DDD methods, but the specific patterns and the many workshops associated to DDD are about application development. In contrast, the scope of this document is about enterprise data domains, which inherently span many different enterprise applications and data stores. Therefore, in this document the term 'data domain' is not referring to DDD.

In the big data community, the notion of data zones has become commonplace. Before 'big data' the Kimball, Inmon and Data Vault data warehouse architectures also prescribed the use data zones. Within any data store that contains vast amounts of data, it is extremely useful to partition data sets into different physical zones that may differ by lifecycle phase, quality or security controls on the data. Unlike data lakes and data warehouses, in a dynamic Data Fabric or trusted Data Mesh, the data will be distributed and decentralized. Thus, the notion of a data zone may be associated to different storage engines, processing technologies, or physical cloud infrastructure.

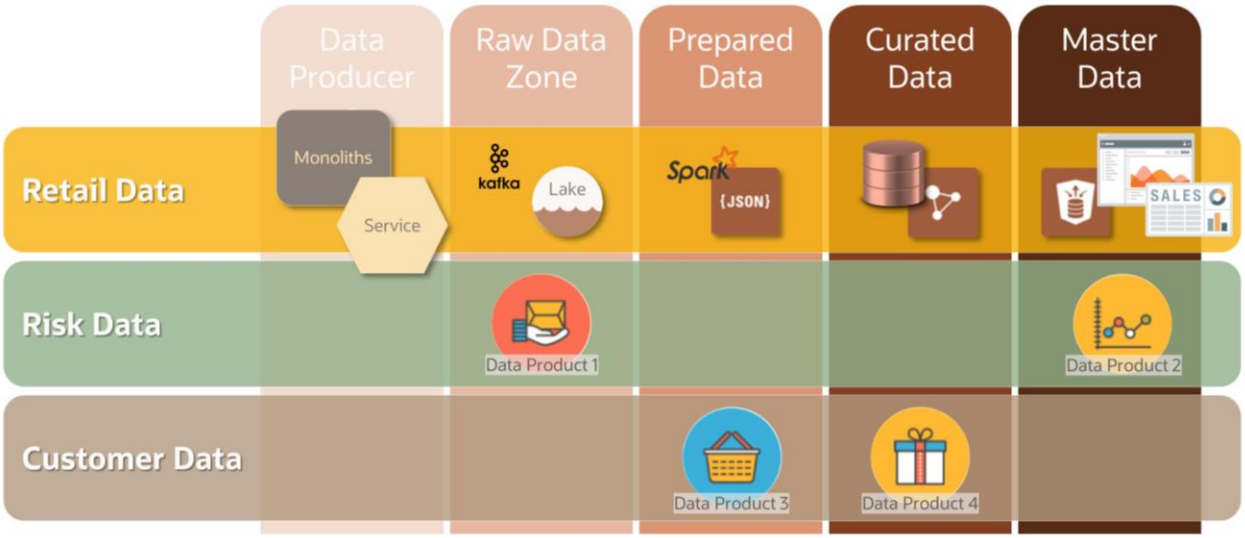


Figure 17: Example data domains, data zones, technologies and data products that can exist in any zone

Individual companies and projects will have unique requirements for the kinds of zones that are required. Some organizations will place top priority on security controls, while others may prioritize rapid development and iterations on their data products.

DATA ZONE	DESCRIPTION	EXAMPLE DATA PRODUCTS	EXAMPLE TECH
Transient Zone	Data that is staged and prepared for ingestion, may be deleted upon ingest	N/A, ungoverned	File systems, producer provided queues
Raw Data Zone	Collection point for events and data emitted directly from applications and data stores (eg; producers)	Raw data feeds are often quite interesting to data scientists, who may place data product governance controls directly on the raw feeds.	Apache Kafka or JMS for events, Object storage or HDFS for files/bulk data
Prepared Data	In this zone, basic quality filters will have been applied to filter and format data into conforming syntax / structures	Data products for IT data pipelines (eg; ETL or stream integrations), eg; full supplemental change data on source customer tables	Self-service CDC, replication tools, stream integration

DATA ZONE	DESCRIPTION	EXAMPLE DATA PRODUCTS	EXAMPLE TECH
Curated Data	Curation often implies models created by human activity, perhaps by a Data Steward or Data Analyst	Conventionally, an operational data store (ODS), or in an event-driven context a set of governed Topics with Avro/JSON payloads	Data catalogs, schema registry and pipelines for transformation
Master Data	As with conventional MDM systems, master data is a 'gold copy' of important data records (eg; customer, product, etc)	Master data records themselves are the data products, fully packaged and managed as discrete entities with their own lifecycle	Relational databases, graph engines, data pipelines, business rules
Secure Zones	There is an entire sub-discipline on this subject. Zones may include DMZ, Trusted or Restricted areas	Data products could include unencrypted financial transactions, account balances, hashed passwords, etc.	Security frameworks of applicable tools like Kafka, Hadoop and DBs

Although each organization may choose to organize zones differently based on needs, a common practice for any Data Fabric or Data Mesh will be to define data products as well-governed entities in the zones. A data product could be in any zone. Data consumers may be interested in consuming data at different stages of a lifecycle. For example, data scientists running regression analysis on user behaviors will want to see raw events directly from the data producers. While C-level dashboards and reports are usually built on curated/master data that is highly trusted.

In summary, the work of defining data domains and data zones within a dynamic Data Fabric or trusted Data Mesh should feel familiar to data architects, but there is a significant shift of scope to think about and perform this decomposition across data assets that are decentralized. For business domains, an intuitive solution will be to use data catalogs to govern data assets within one or more domains. For data zones in a decentralized architecture, there may be different tooling choices depending on data producers, data consumers and any required KPIs and SLAs.

What Makes a Mesh a Mesh

When we describe a dynamic Data Fabric, it is the low-latency streaming which makes the Data Fabric dynamic. When this document introduced the idea of a trusted Data Mesh, the ability to correctly handle ACID data is what makes the Data Mesh trusted. But what is it about the Data Mesh that makes it a Mesh?

In modern everyday living, there are number of areas where the term 'mesh' is already commonplace:

- **WiFi Mesh** – for home and business internet technology, the shift from single routers to a group of decentralized WiFi access points makes the internet faster and more reliable
- **Smart Home Mesh** – for connected smart home protocols like Z-Wave, Zigbee and others, the IoT devices you use in your home can connect to each other to repeat signals as well as the main hub/router
- **5G Mesh** – as telecommunications providers deploy 5G networks across the globe, a crucial attribute is the deployment of small local cells to maintain strong connections and very high speeds
- **Service Mesh** – for software application developers it has become mainstream to design applications that can run in containers, Kubernetes, OpenShift etc as part of a decentralized mesh of software components

Mesh Controllers / Control Plane

There are a few defining characteristics of what makes a mesh, a mesh. First, it is a networking concept. The central idea of a mesh is first and foremost about a connected network of things. Second, a mesh is decentralized. A primary distinction of any mesh network compared to a non-mesh network is that the mesh does not depend on connecting through a centralized hub. In a mesh network, nodes and devices can connect directly to each other and communications can be routed along without going via a hub ever time. Third, any mesh will have a type of control plane. All mesh networks have a controller or a hub in the network somewhere, but functional communications are not mandated to move through them at each network hop. Regardless of whether it is called a hub, a controller, or a control plane, every mesh will have this higher-level observer process that helps to manage the mesh nodes. In other words, an unmanaged mesh is not a mesh at all.

THE SIDECAR-PROXY PATTERN

Originally published in 1994, the "Gang of Four" book, "Design Patterns: Elements of Reusable Object-Oriented Software"[xxxiii] popularized object-oriented patterns including the Proxy pattern. The basic design principle with Proxy is to provide an interface to some other code object, typically with additional logic being applied within the proxy object. An early use of Proxy in distributed systems was as a "Remote Proxy" with the local interface hiding the complexity of communications back to a remote object.

Around 2016/17 the term Sidecar Proxy (occasionally referred to as Sidekick Pattern) became definitively associated with the Service Mesh movement (Istio, Linkered, etc) and subsequently has become a central design feature of modern Service Mesh like Kubernetes, OpenShift, Kong etc. It is also the definitive pattern used in all public cloud architectures from all the major providers.

The Sidecar is an interface proxy object which is coupled to distributed applications. A group of common functions (may vary by implementation) such as observability/telemetry, logging, configuration, network routing etc is managed by the Sidecar so that the base application software does not have to. The Sidecar is then governed by a common Controller (aka, Control Plane) that can take actions on all associated Sidecars – effectively controlling a vast number of 'data plane services' through the Sidecar Proxy.

The Sidecar Proxy is the fundamental base pattern for how modern Public Cloud infrastructure works, and how IT organizations can operate applications "as a Service" within their own on-premise infrastructure.

A Data Mesh should always have a controller whereas a Data Fabric might not. A single mesh will have a common controller that is connected to and governing the operational lifecycle of the mesh nodes. There could be many different meshes interacting with each other, but the governing scope of an instance of a mesh is defined by the set of nodes/pods/services that are managed by a particular controller. For example, a WiFi Mesh and a Smart Home mesh can run in the same house, and they can interact with each other, but each mesh ultimately has its own physical controller and is governed distinct from the other. In contrast, some notions of a unified Data Fabric are mainly logical in conception. Not all Data Fabrics will necessarily have a common systems controller. For example, some Data Fabrics may be unified only through the use of strong data governance methodologies (process controls) and metadata catalogs for shared lineage and views into different data domains. For the purposes of this document, the full definition of a Data Fabric is noted in the section, "A Dynamic Data Fabric for Enterprise Needs".

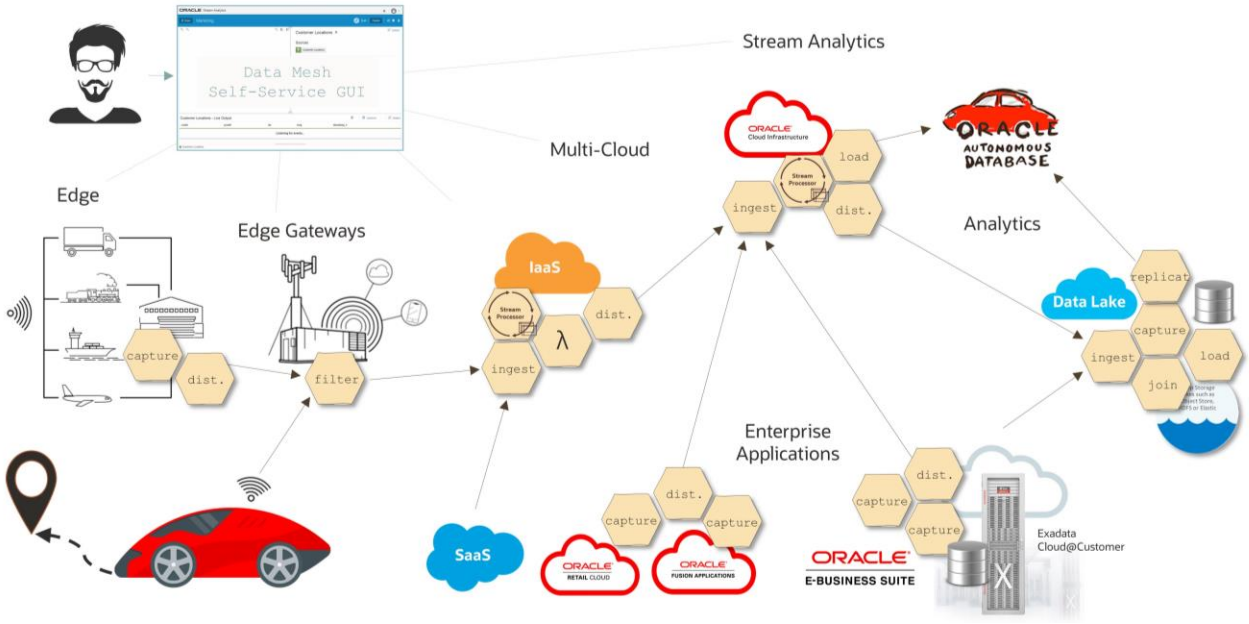


Figure 18: A Data Mesh will have common controllers but be capable of running in different physical, decentralized locations

Compared to conventional hub-and-spoke styles of data integration (eg; ETL and Data Federation) the topology of a Data Mesh is unique. Like all meshes, a Data Mesh is (1) a networking concept, (2) decentralized, and (3) utilizes a control plane, sidecar type pattern for the core services. Real-time data can be captured and routed across mesh nodes that physically reside in any connected infrastructure. There is no need for a single hub or a common service bus. In a large enterprise, there could be many different Data Mesh deployments each with their own control plane

services. In fact, with Oracle GoldenGate the Data Mesh could be controlled from a serverless infrastructure hosted on Oracle Cloud, on-premise infrastructure, or even inside of non-Oracle clouds. Regardless of where the GoldenGate control plane runs, it can distribute or receive events from any other deployment. This is the unique value proposition of a mesh type deployment, seamless routing of enterprise data events across any WAN-connected devices.

Key Personas, Role of Self-Service

Like all areas of software development, the shift to low-code and self-service capabilities are critical for improving Data Fabric access to non-technical people. Removing process-intensive IT projects from the basic tasks that most business people require is a win-win for both the business and IT. Data Fabric and Data Mesh are no different in the sense that self-service user interfaces are the future, it is about democratizing data access and eliminating the all-encompassing dependency on IT people and process.

Self-service data preparation is a well-established category of software. It is a part of all major analytics platforms and a common feature area for data integration tools. However, there are limits to what has been accomplished with self-service tools. Even with the incorporation of advanced machine learning (ML), natural language processing (NLP), and graph-based knowledge bases (KBs) the self-service technologies still require humans in the loop for curating data and establishing the accurate meaning (semantics) of the data. The widespread use of ML, NLP and graph is helping to automate and improve data curation, but they are not replacing humans yet. As such, a dynamic Data Fabric and trusted Data Mesh will still require human personnel to operate and govern the Fabric/Mesh.

KEY PERSONAS	
Data Operations / DBA	• Expert in the operational data platforms, such as OLTP DB engines & doc stores • Responsible for key KPIs and SLAs around uptime, availability and recoverability • Use self-service tooling for exposing changed data to Data Engineers
Data Engineers	• Developers of the data pipelines – movement and transformation of data • May use self-service tooling, or low-level pipeline technology that requires coding • Central execution of DevOps for data, updating and patching production pipelines
Data Product Managers	• A type of Data Analyst, aligned to the Business (direct or dotted-line) • Owners of the Data Products (see Data Product Thinking section) • Coordinates between IT/engineering, the Business, and Data technical experts
Infrastructure Admin	• Cloud Operations, networking and security within Public Clouds • Container Registry & Service Mesh, for POD administration and CICD lifecycles • Hardware and Networking, for on-premises teams running Fabric/Mesh components

Self-Service Data Products

In a transitional phase of self-service there will be some data products which can be managed by non-technical people, while other kinds of data products will naturally still require IT organizations to curate and produce the data products. Self-service data access will typically begin with a data catalog, where non-technical people can “shop for data” by selecting data assets into a virtual “shopping cart”. At check-out, the self-service process would prompt the non-technical human consumer how they would like their data served to them. For example, a user might select different formats (Excel, JSON, Avro, XML, etc) or different locations (data mart, data lake, etc). In a self-service process, data can be browsed by domain as well as by data zone.

In a real-time, streaming Data Mesh there may be a variety of data products – some of which are entirely self-service while others have been curated and prepared by IT. For example, in Oracle GoldenGate there is the notion of a Distribution Service which includes a microservices-based API for consuming changed data events. GoldenGate data

formatters can provide changed data in popular formats like XML, JSON, Avro etc. Thus, a streaming data product could be an API where downstream subscribers can always consume most recent data (or schema) changes. Other kinds of self-service data products could include data visualization/analytics on the data stream, where visualizations are created by non-technical people rather than big data developers.

Finally, it is worth pointing out that self-service tools benefit from Data Fabric and Data Mesh even when they are outside the frameworks. For example, if self-service tools were used to create a local data mart for reporting, the dynamic Data Fabric can be integrated to continuously feed updated data that IT has curated. This way, self-service tooling on personal devices can co-exist and benefit from a broader enterprise Data Fabric or trusted Data Mesh.

Continuous Transformation and Loading (CTL) Data Pipelines

Over the past 50 years of computing the task of data transformation has either been done as a scheduler-driven batch process or message-by-message as events. Batch processing is fast but infrequent (eg; daily) whereas event-processing is high frequency but traditionally much slower (in aggregate). Beginning around 2015 a new approach was emerging as a third way. This new approach became possible due to widespread easy access to streaming MPP (massively parallel processing) platforms and the dramatic reduction in relative compute costs (both in terms of Cloud/serverless as well as on-premise hardware). For the first time it became relatively economical to run supercomputer sized clusters dedicated to event-by-event data transformation in real-time.

Whereas ETL and E-LT approaches to data transformation occur in batch, on a scheduler, the CTL approach processes data as the data events occur. The computationally tricky part is dealing with statefulness of data. In ETL and E-LT approaches, there is a naturally occurring time window – the frequency of the batch job. If your ETL job runs every 24hrs, then by definition you are processing 24hrs worth of data at a time. All mappings and joins on data are bounded by the naturally occurring time window of the scheduler frequency. Further, the engine running the batch jobs are typically sized according to the largest possible amount of data that could be consumed in a batch.

DATA TRANSFORMATION TYPES		FREQUENCY
ETL	• Extract, Transform and Load (ETL) • ETL pipeline executed by scheduler/cron and then performs tasks • Data is processed in central engine, separate from any data stores	Batch / Micro-Batch
E-LT	• Extract – Load, Transform (E-LT) • ETL pipeline executed by scheduler/cron and then performs tasks • Data is staged into target data store, transformations occur within the data store	Batch / Micro-Batch
CTL	• Continuous Transformation and Loading (CTL) • Streaming pipeline is available as events arrive 24 x 7 x 365 • Data is processed within streaming engine • In some technologies the CTL process can selectively co-locate with batch processing (eg; Flink) as well as Data Science / SQL processing (eg; Spark)	Continuous / Event-by-Event

In contrast, a CTL pipeline must manage the statefulness of data within defined time windows (similar in concept to windowing functionality of CEP (complex event processing) engines. These time windows are used to drive transformation logic for the data joins and aggregations that may occur as part of the business logic. Sizing of these environments can be done by planning for the amount of data that must be held in-memory in the DAG (directed acyclic graph) during processing. Technology like Oracle GoldenGate for stream analytics includes user-configurable settings for managing a data cache, utilizing an in-memory data grid that spans the cluster. Effective management of statefulness for streaming data is what separates a modern CTL approach from event processors of the past.

Trusted CTL Data Processing

As noted at the beginning of this document, a key reason why data integration is its own enterprise software market segment (as opposed to other integration styles) is because of the assurances of trust in working with consistent and correct data. Data integration tools (like ETL, Replication, Data Federation) are built from the ground up to work with and respect the ACID-consistency boundaries of relational database transactions. This cannot be said of other integration styles, many big data frameworks, or of messaging systems such as Apache Kafka.

Continuous Transformation and Loading (CTL) data processing must fundamentally be a polyglot streaming approach that handles a variety of payload formats (eg; XML, JSON, Avro etc) and maintains the consistency and correctness semantics of relational databases and analytic formats (eg; Parquet, ORC etc). The responsibility for maintaining the correctness of the data should not be left entirely in the hands of the pipeline developers or the data consumers – the CTL system itself should enforce trust policies while processing business data.

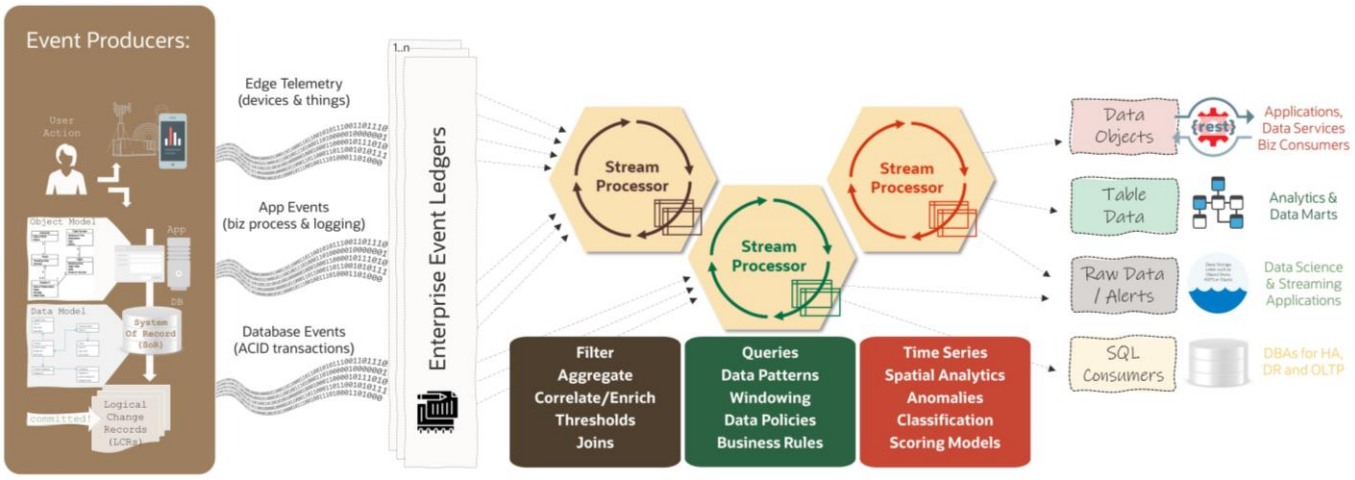


Figure 19: CTL processing components receive events from the ledgers, process data in real-time and then output trusted data to consumers

As business organizations gradually shift from the old batch processing world into a modern streaming data world, they will achieve massive benefits that come with being able to iterate faster on data products. This CTL shift to a decentralized data processing model is also more aligned with a multi-cloud architecture, where the stream processors could be running in any of the major public clouds or on premises while low-latency data events move continuously between them. It is this enterprise-wide, more repeatable vision which ultimately leads us to the industrialization of streaming data product factories.

Repeatable Data Product Factories

> ***"We realized that the true problem, the true difficulty, and where the greatest potential is – is building the machine that makes the machine. In other words, it's building the factory. I'm really thinking of the factory like a product" -Elon Musk***

Everyday products in our personal lives come from a variety of creators. Some products are individually hand-crafted by skilled artisans. Other products are mass-manufactured in high tech factories sometimes operated by robots. A modern Data Fabric or trusted Data Mesh will provide the means to industrialize and automate the means of producing data products. Rather than depending on one-off, bespoke data artisans to create and manage products, the factory and automation within the factory is where the true magic happens.

Like Henry Ford before him, the focus of Elon Musk is on the factories that produce the products. He says in a 2016 shareholder meeting, "*We realized that the true problem, the true difficulty, and where the greatest potential is – is building the machine that makes the machine. In other words, it's building the factory. I'm really thinking of the factory like a product*"[xxxiv]. The factory is the product. Elon Musk applies this thinking to Tesla and SpaceX, and the same thinking should be applied to factories that make products from data.

For smaller companies or tactical projects, the data products produced can surely be created by one-off, bespoke data artisans (eg; highly skilled developers, data scientists, or data analysts). But for larger enterprise IT organizations that are supporting 1,000's of business workers across continents, investing in an automated data factory concept could be the difference between success and failure in the modern digital age.

To move past the simple, bespoke data products that are hand-crafted by developers it is necessary to conceive of the data pipelines as the means of production and automation for creating data products. More concretely, data products come to life as data is captured from source producers, joined with other data, cleansed and prepared, and ultimately written into some kind of digital payload. CTL (continuous transformation and loading) data pipelines can be hand-crafted by data artisans working in frameworks like Apache Flink or Spark, but this is no more scalable than the olde tyme cobbler making shoes with a wooden hammer.

Tooling that automates CTL pipelines will need to have the following attributes:

- **Self-Service** – utilizing no-code graphical user experience that is friendly for a data analyst to use
- **Metadata Driven** – layer of indirection between the specification and the code, runners, or the engine
- **Autonomic** – capable of self-optimization (performance/scale) and self-healing (data drift etc)
- **Serverless** – either in a public cloud, or as an IT-managed Service Mesh deployment
- **Trusted, Polyglot** – capable of preserving ACID consistency as well as handling schemaless payloads

The purpose of aspiring to a highly advanced data product factory is to achieve consistency and repeatability in the production of business data products. As more an more industries are transformed by the digital economy, the opportunities for business to use data productively are multiplying each year. All products and services are going to be radically transformed by the availability of data and data products. It is the advanced data product factory which will enable the business/IT to achieve positive results consistently and repeatedly with their data assets.

Fourth Industrial Revolution and Smart Factories

> **"In the new world, it is not the big fish which eats the small fish, it's the fast fish which eats the slow fish." – Klaus Schwab**

In 2015, Klaus Schwab, the founder of the World Economic Forum, coined the term 'Fourth Industrial Revolution'[xxxv] and along with it the central idea that we are entering a new phase of mass industrialization. In essence the idea is that there have previously been three waves of industrialization (1. steam power, 2. electrification, 3. digitization) and that we are now entering into a fourth phase: cyber-physical. The cyber-physical stage is marked by breakthroughs in extreme automation of our digital systems (AI, ML, neural nets, etc) and increasing connectedness to the physical world (5G mesh, robotics, nanotech, biotech, internet of things, etc). A central and recurring theme in the Industry 4.0 thinking is about smart factories and agile supply chain and logistics.

Just as the production capabilities of physical products are undergoing a massive transformation, the means of producing data products must also move into this era of Industry 4.0 – embracing automation, robotics, and the use of AI, ML and graph technology as enablers for smart factories for data products.

Global economic leaders have repeatedly elevated the discussion of smart factories at global events like Davos[xxxvi] for the World Economic Forum, but why? Quite simply there is a broad realization that digital business transformation is reshaping the global economy. The key insight is that businesses which are capable of moving faster with greater agility are going to be the economic winners of the future. As Klaus Schwab said, "*In the new world, it is not the big fish which eats the small fish, it's the fast fish which eats the slow fish.*" In the high tech world, we know this as the Innovators Dilemma[xxxvii], where new startups wielding disruptive new technology regularly outmaneuver established incumbents.

Agility is no longer just a buzzword. In many industries agility is a life-and-death matter for a corporation. Implementing a smart data factory could soon become mandatory for survival of businesses which depend on data

for revenue, strategy or as a competitive advantage. What used to be a luxury (investing in a world class data product team) will soon be table-stakes just to get in the game for many global industries.

Manufacturing Agility with DevOps, CI/CD, and DataOps

Agility in IT means different things to different people. It could mean being able to hot-swap drives without disrupting online applications, promoting code to a production data pipeline without creating any downtime, or migrating entire applications to decentralized cloud computing. In truth, for modern data product management the concept of agility must apply at several layers of IT – a data product supply chain is only as agile as its least-agile link.

It is such a mainstream part of software development, it is hard to believe that the term 'DevOps' first appeared in 2008[xxxviii]. Before the pervasiveness of DevOps it completely normal for code promotion (from Development to Production) to take days or event weeks of IT work. Everything from the organizational structure to the tools used to work with infrastructure required human intervention and complex processes for communications. In the cloud-driven software development ecosystem circa 2020 these old inefficiencies have been inverted and replaced with DevOps automation toolchains designed to support CI/CD (continuous integration, continuous delivery).

In the software development domain DevOps has been strongly associated with the popularity of microservices design patterns. Microservices are designed to make it easier to continuously evolve an application without disrupting online operations. In the data management domain DevOps practices have been slower to take hold. DevOps surveys have repeatedly indicated data management challenges in keeping pace with rapid delivery schedules[xxxix]. In part this is due to most data management systems still being managed as monoliths, and also because data and schema are by-definition tightly coupled to the application logics. Regardless of whether data is modeled as 3NF in a database, or a JSON/Avro documents in Kafka, the application code will always have some impedance mismatch[xl] with whatever durable store is used.

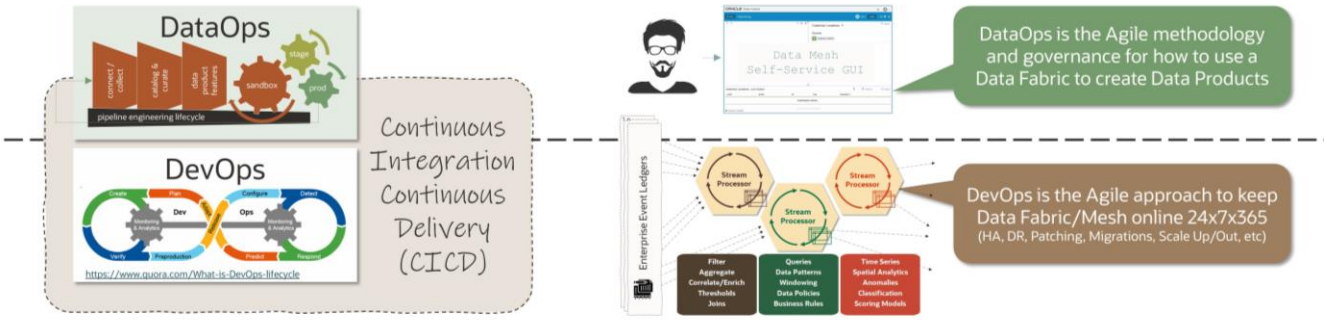


Figure 20: DataOps and DevOps are distinct but critical parts of an agile approach to Data Fabric and Data Mesh

Creating an agile CI/CD approach for a dynamic Data Fabric requires navigating the challenges of doing effective DevOps for data management. DevOps engineers will have to be concerned with keeping the Data Fabric/Data Mesh online for continuous operations. Operational concerns like high availability (HA), disaster recover (DR), patching, migrations (software or infrastructure), as well has vertical and horizontal scalability all need to be accounted for. These DevOps concerns are supportive, but orthogonal to the agile processes applied in DataOps control flows.

WHAT ABOUT 'ProdOps'?

Beginning in late 2018 the term 'ProdOps' starts to appear more frequently in CI/CD domain discussions. For this document the term has not been included in discussion of Data Fabric or Data Mesh because the definition of ProdOps remains unclear. In some communities, ProdOps stands for 'Production Operations' and refers an IT organizational role in a Production unit that performs a set of tasks that support CI/CD and DevOps. In other communities, ProdOps stands for 'Product Operations' and is associated with the full lifecycle of products that would include sales, marketing, product management, engineering, quality assurance and support engineers. Both definitions are interesting in the context of data product thinking and data mesh capabilities, but since there is still no consensus definition it has been omitted as a major topic in this document.

DataOps will principally be concerned with governing and managing the activities within the smart data factory itself. For example, connecting to data producers, creating and versioning the lifecycle of data pipelines, creating effective reuse of common business rules and ML algorithms, and managing the dependencies across many data products. Generally speaking, DataOps is tool-independent, but any tooling for a dynamic Data Fabric or trusted Data Mesh should support highly automated DataOps controls, especially for data governance.

Security and Governance in a Data Mesh

Data security is a perennial "top 3" concern for CIOs[xli], and the widespread adoption of AI and decentralized multi-cloud infrastructure is keeping the subject critical for all. The impact of tighter governmental regulations on data, such as for GDPR, Sarbanes-Oxley (SoX), Basel I & II, and HIPAA, place mandatory controls on how data must be secured and governed. These controls may affect countries differently depending on where data is physically stored and under which legal jurisdiction the operating company is under. Multi-national organizations that wish to comingle data coming from different applications and operating locations already face a complex web of governance requirements on the data. These regulatory mandates are no small matter, organizations can be fined[xlii] substantial amounts for non-compliance.

Security and data governance is first and foremost a matter of the physical persistence tier. As noted in the introduction of this document, a dynamic Data Fabric or a trusted Data Mesh is not a replacement for data stores. Operational databases, data marts, data warehouses and data lakes will continue to provide data persistence within an enterprise data estate. As such, these data stores will continue to be the most critical access point for managing secure, governed access to sensitive data resources. Likewise, any compartmentalized data access policies (restricting which users may see certain data) must fundamentally be handled as part of the data store tier. Cryptographic controls and obfuscation of the data at rest are a data store function not replaced by any Fabric or Mesh concepts.

In distributed, multi-cloud architecture data stores may physically reside in different clouds (possibly in different political/regulatory boundaries) under administration by 3rd party vendors. Fully managed OLTP database engines, Data Warehouse databases, or object-storage based cloud data lakes have physical storage and other infrastructure that is owned and managed by the cloud vendors, not the owners of the data. Each cloud vendor or data store technology will have different contractual policies or technical safeguards in place to protect unauthorized access to data at rest or data in use. As a holistic approach to multi-cloud cybersecurity, a useful place to start is with the 2018 International Standards Organization (ISO) report, *"ISO/IEC 27103:2018 — Information technology, Security techniques, Cybersecurity and ISO and IEC Standards"*[xliii]. This ISO report describes a multi-layer "Defense in Depth"[xliv] approach to security and makes recommendations across five general aspects (identify, protect, detect, respond and recover) of security.

In several regards the Data Mesh and Data Fabric concepts complicate the data security picture. For example, if an organization could simply centralize all their data into a single repository, with a single access control list, a single policy enforcement point, and in a single physical location with a single political regulatory jurisdiction – this would dramatically simplify the operational security and data governance of that data. But Mesh and Fabric are all about empowering a decentralized data architecture. There is an inherent trade-off that comes with a business decision to operate data services within infrastructure owned by other legal entities, or to provide business operations outside the political boundaries of a home country. For most large organizations these business decisions have already been made (to decentralize the data) and now the question is really about how to effectively and responsibly manage data that is widely decentralized.

Unique security and governance aspects of the Data Mesh and Data Fabric are mainly around providing effective security and governance of data that is in motion, as it must flow between data stores and between clouds. From a security standpoint, the networking approach is crucial for highly secure decentralized data movement in a Mesh. From a governance standpoint, decentralized data creates additional challenges for data lineage, explainability, data consistency and validation.

Decentralized Authentication and Authorization

When Data Mesh and Data Fabric services are deployed across many networks the secure interoperation of the services is typically a function of network layers 6 and 7 of the Open Systems Interconnection (OSI) layered networking architecture[xlv]. For example, Web Sockets (WSS) are generally considered a Layer 7 service and when used with mutual Transport Layer Security (mTLS) for two-way encrypted authentication, services can achieve very high trust levels for service-to-service communications. Mutual authentication schemes (using trusted certificates) are effective[xlvi] at protecting against attacks such as:

- **Man in the Middle** – third parties trying to eavesdrop on your communications
- **Replay Attacks** – older messages are replayed to try and fool the server
- **Spoofing** – bad actor using false data (from a trusted source domain) to pose as another trusted user
- **Impersonation** – bad actor using look-alike / similar data in order to gain the trust of another user

Most Service Mesh frameworks (which are a foundation layer for dynamic Data Fabric and trusted Data Mesh) can use mTLS based authentication. Similarly, operating from services from public clouds usually provides an mTLS foundation for authentication.

Further up into the application layers it is important to provide a delegated way to authenticate users and authorize access to particular resources, APIs and data. Standards such as OAuth 2.0[xlvii] and OpenID Connect[xlviii] provide a canonical archetype for how to securely authorize using token-based claims for access. However, the OAuth 2.0 and XACML[xlix] (policy language for authentication) are typically run from centralized gateways or hubs, and oftentimes the individual cloud providers may have their own unique policy languages. This makes running a decentralized Data Mesh or Data Fabric quite challenging since the identity providers and access policies will differ from one network to the next.

Decentralized authorizations must be able to scale out to modern Mesh-type solutions. One way to do that is for the Mesh controllers to interoperate natively with the authorization frameworks of the various cloud or on-premises tools. Another way is to implement and support a decentralized policy-based controller that can run as a part of the Mesh itself. The Open Policy Agent (OPA) project[l] is one such option.

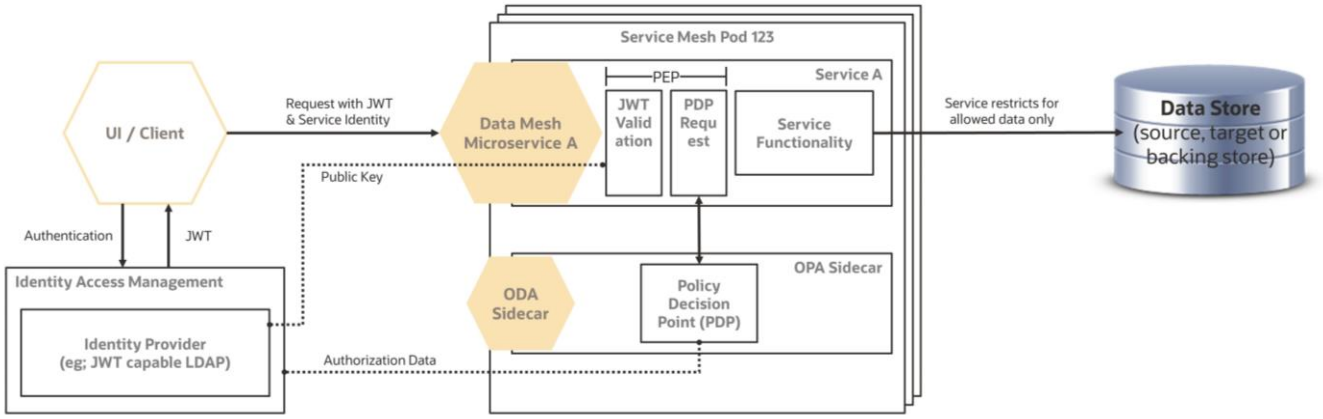


Figure 21: Open Policy Agent sidecar for distributed PEP and PDP within microservices

With the OPA architecture each microservice (eg; a Data Mesh Microservice) is its own policy enforcement point (PEP). The decision making, however, is handled via an OPA sidecar proxy. Policies evaluated by OPA sidecar are local to each sidecar, and they can be manipulated by an OPA Client (all communications are REST APIs). Additionally, the policies themselves (written as a JSON notation called 'Rego') may be stored in an LDAP store and refreshed via JSON Web Tokens (JWT) upon authentication, or replicated periodically.

A multi-cloud Mesh architecture can take advantage of the ODA sidecar proxy pattern as a means to distribute the PEP (policy enforcement point) and PDP (policy decision point) across individual microservices in the Data Mesh, while preserving the centralization of PIPs (policy information points). This way, there is no single hub-and-spoke required for individual decisions, nor is it necessary to translate policies into the myriad of different IAM (identity access manager) PDPs that tend to be unique per cloud provider. Distributed Mesh and Fabric services can share policies across a wide array of networks.

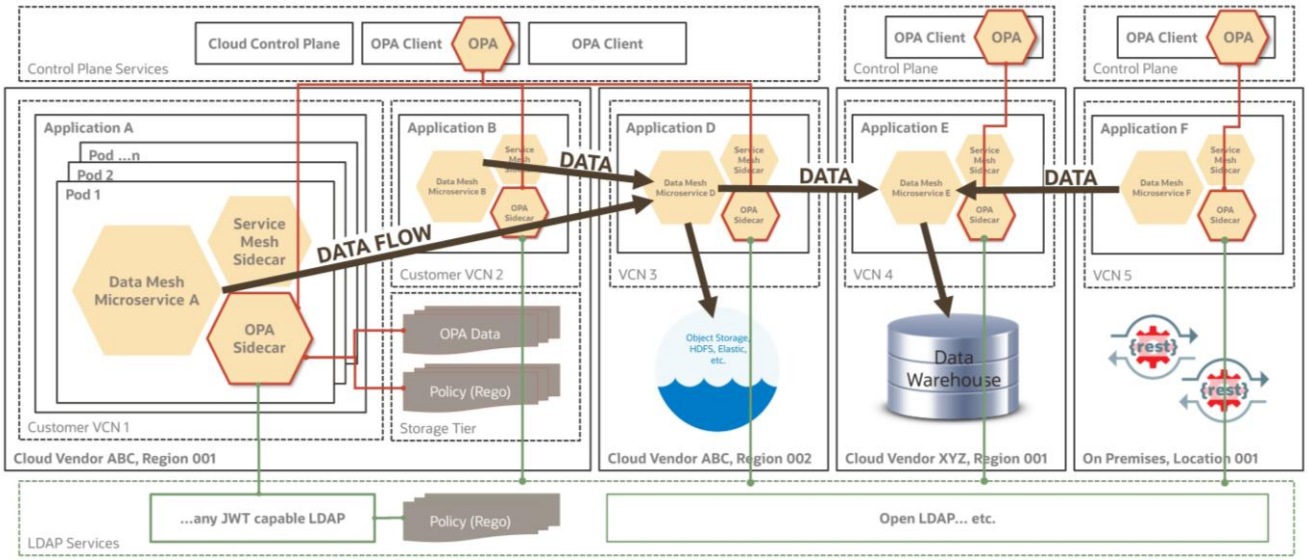


Figure 22: Open Policy Agent sidecar for distributed, multi-cloud policy decision-making with a shared policy language

The combination of WSS, mTLS, OAuth 2.0, and Open Policy Agents can enable highly secure, trusted services that run as a dynamic Data Fabric or a Data Mesh. Highly decentralized data services can interoperate with each other on top of lower layer protocols like TCP and HTTP, while the decentralized aspects of authorization are taken care of by OPA-like patterns for distributed policy decisions. The OPA policy language (Rego) is a highly flexible JSON notation that may also be used for data-tier access policies, this will be particularly useful when filtering data sets (payloads) at the API layer of multi-service REST endpoints. However, there is still a need for fine-grained data access controls at the persistence tier (eg; data stores) and within messaging subsystems (eg; Kafka, JMS, etc).

Granular Data Access Controls

In many ways the OPA approach to data access control is the simplest to govern, since it provides for an API-driven approach, with uniform policies, and distributable PEPs and PDPs. However, most enterprise domains won't have the luxury of a modern agent-based framework for the vast majority of their legacy data tools. Most data centric tools, such as databases, messaging frameworks and big data lakes will ordinarily have their own proprietary PEPs and PDPs built into the tool. Most reasonably modern data stores and data integration tools will integrate with LDAP, but oftentimes interoperability is limited to simpler requirements like authentication and single-sign-on (SSO). More difficult requirements for fine-grained data access controls are typically deferred to the data store itself.

Other access controls can include:

- **Role-based Access Controls (RBAC)** – the Data Mesh / Data Fabric tool will have its own RBAC model for different user roles, supporting varying access levels to platform features
- **API Access Controls** – if a policy agent approach is not taken, an API Gateway could be used as an external access control to Data Mesh / Data Fabric client-facing REST APIs. API Gateways are typically complementary to core Service Mesh security, focusing on "north-south" traffic rather than "east-west".
- **Database, Data Lake & Messaging Access** – individual data resources will have their own built-in RBAC controls, typically the Data Mesh / Data Fabric tools are just ordinary clients from the data resource

perspective. Fine grained access to specific columns, properties, and schema elements may be managed by policies that are local to the data resource itself.

- **Catalogs and Classifications** – a well governed Data Mesh / Data Fabric framework will have a local data catalog that is capable of classifying assets governed from the catalog, the RBAC controls for the platform should be capable of restricting access based on these catalog classifications and metadata.
- **Cloud Compartments** – public clouds will typically have some kind of compartmentalization, these compartments can be used by the identity sub-systems to provide additional layers of RBAC controls to resources associated to a particular cloud tenant.

As with any distributed software environment, the "Defense in Depth" approach to access controls requires a multi-layered model.

Other Security Considerations

There are many other security concerns important to decentralized Data Fabric / Data Mesh deployments. Although this document is not intended to provide a deep-dive into security, it is important to mention the following:

- **Encryption of the Ledger** – regardless of the type of ledger (Event Sourcing, Event Streaming, Data Replication, Blockchain etc) the underlying storage tier should have the ability to be fully encrypted on disk
- **Immutability of the Ledger** – for Blockchain type of ledgers should have protection and controls that require quorum for commits and immutability of the committed transactions
- **Wallets, Key Stores and Secret Stores** – a multi-cloud Data Fabric / Data Mesh should be capable of using a variety of wallet, secret store and key store APIs on any of the infrastructure it may be deployed into
- **Credential Management using Principals** – in a distributed mesh, it is often necessary to grant access to another actor "on behalf of" a given user. Some tools will need to be able to reference, pass and receive principals so that credentials may be provided to external data stores without the calling tool being able to discover any sensitive information about the user credentials
- **Network Encryption** – encryption can be applied at multiple network layers using different protocols. A typical Data Mesh / Data Fabric deployment may run on a private dedicated encrypted line, or use IPSec VPN clients for communication. The payloads themselves may run on HTTPS or Websockets over SSL/mTLS.
- **Reverse Proxy** – oftentimes important to simplify port routing, a security reverse proxy can sometimes also be employed to inspect client invocations for malicious requests
- **High-Side / Low-Side Guards** – isolated high security networks, often in the national security domains, will often use 'guard' software to inspect payloads as they move between them
- **Target Initiated Paths –** for distributed replication of the transaction ledger across Firewalls or Demilitarized zone (DMZ), the ability of the recipient service to initiate an outbound connection, when inbound connections are not possible to initiate externally (more trusted location creates link to less trusted location)

Security can be a complex topic and it is imperative to do well when fully embracing a decentralized data architecture. An enterprise tool kit for a dynamic Data Fabric or trusted Data Mesh should provide sophisticated and verifiable for security controls – as well as high levels of data governance.

Data Governance

In most regards, the governance controls in a dynamic Data Fabric or trusted Data Mesh should mirror those used conventionally as part of monolithic data management architecture. The business demands for regulatory and compliance do not change just because the underlying technologies have. Regardless of which tools IT uses, or how smart their data product factories are, the top-level requirements for governance are still there.

To the degree that data governance practices are different with dynamic Data Fabrics and trusted Data Mesh, it is mainly due to the unique characteristics of these newer approaches. Foundationally, the nature of data product

thinking means bringing a high degree of KPI and SLA governance to managing data products. The first principle of a decentralized and modular mesh of data services means that data governance techniques must inherently work with data assets that are fundamentally decentralized rather than co-located. The second principle of using event ledgers requires governance policies to apply to data events in-motion as well as the more conventional governance that applies to static data. The third principle of trusted, polyglot data implies that governance policies should trace and track the verifiability of transaction semantics to assure that data flows are not lossy (with the data semantics).

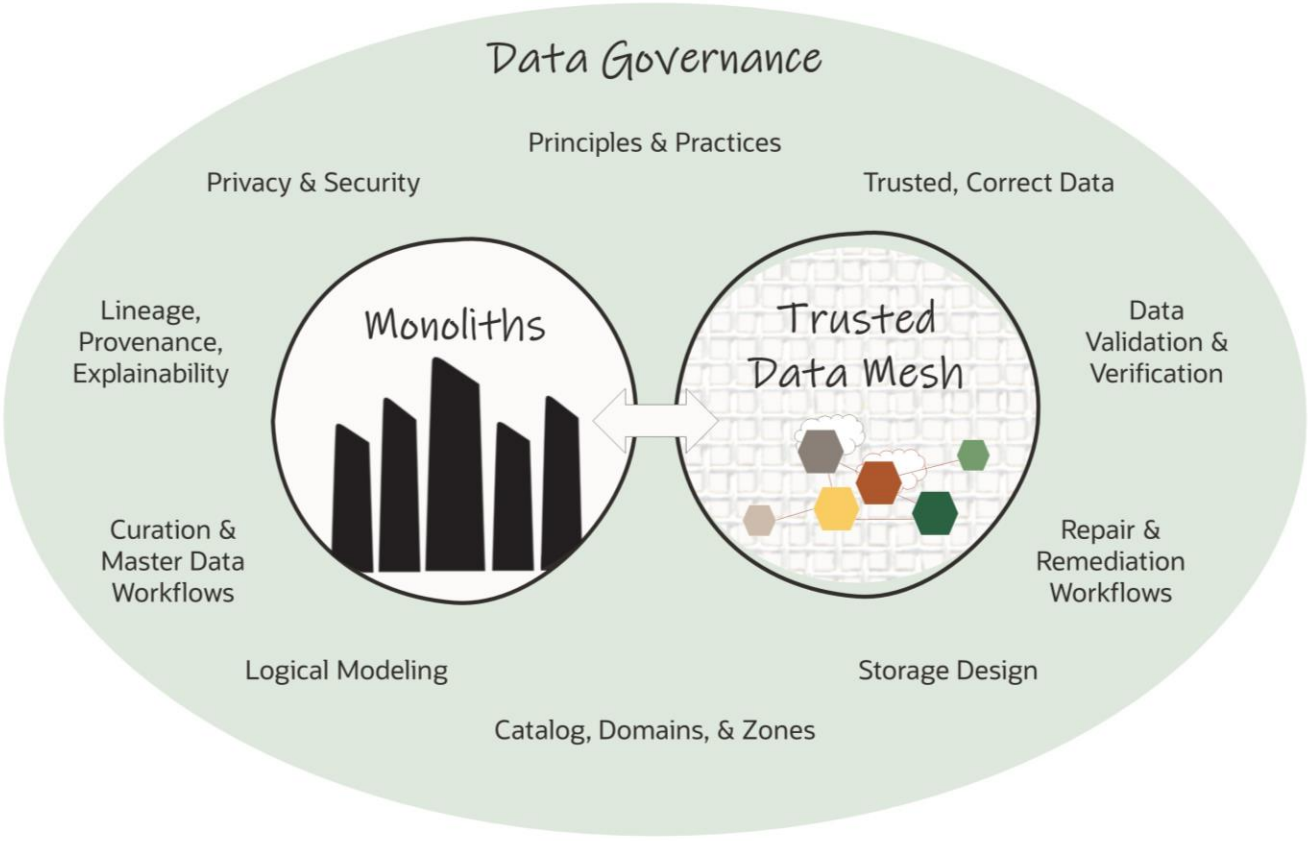


Figure 23: Data governance for a trusted Data Mesh should align with a governed Data Fabric and legacy data monoliths

In most large enterprises, the legacy data management monoliths will coexist with newer dynamic Data Fabric and Data Mesh technologies – therefore, the governance frameworks will need to span both the new and old data architectures. Data governance itself is a discipline, not just a collection of tools. Without people and process controls, any software tool for data governance would inevitably fail. A few of the most important areas of data governance to be applied into dynamic Data Fabric and trusted Data Mesh include:

DATA GOVERNANCE AREA OF CONCERN		UNIQUE CONSIDERATIONS
Principles and Practices	Human factors including the organizational reporting hierarchies, interfaces between lines-of-business, corporate mandates and executive sponsorship, and best practices that are imbued in the culture of supporting Data Stewards	Shift to a real-time data architecture means impact of governance failures will be felt sooner – a rapid response team should be aligned with DevOps/DataOps
Data Catalog, Domains and Zones	Most large organizations have many different data catalogs to scan and harvest metadata from business systems and analytic tools. These will naturally have reach into different business data domains and different physical data processing zones.	Adopting data product thinking should drive organizations to align catalogs to data products and data domains. Consider a 'catalog of catalogs' to cross domains.

Trusted, Correct Data	Data pipelines that span different enterprise tools with differing processing semantics can cause a loss of data consistency, orphaned records, or duplicate and incorrect data.	A smart data factory concept should incorporate controls that assure correct processing of ACID/polyglot data sets
Data Validation and Verification	In any distributed system it is possible to have data drift over time. Drift may include changing schema definitions or missing records – due to human or machine factors. Validation and verification controls should be able to confirm that data copies match the sources of truth	In a decentralized data architecture, data verification is more than just counting and comparing records. Since datatypes and transformations can occur, it is critical to compare values and schema as well.
Lineage, Provenance and Explainability	As data moves across networks and data pipelines, it can be transformed multiple times and joined with data values that originated in other systems. Lineage provides the exact trace of data flows, provenance tells us which is the ultimate source of truth for the data, and explainability provides a detailed explanation of how derived data was created (eg; inferred data values, automated joins etc)	Real-time, event-driven systems will typically require traceability at the record-level, including external events to APIs etc. (this is different than conventional batch or static lineage that examines 'sets' of data or system-to-system interfaces.
Curation and Master Data Workflows	The concept of a governed data record originated with MDM (master data management) systems and includes coverage of reference data as well. Examples of this curated and tightly governed data include: customer records, product records, citizen data, and reference data such as legal codes, country codes, and analytic reporting dimensions	Data product thinking is a natural extension into this area, the management of data domains and packaged/governed data products is conceptually similar to how master data have conventionally been governed.
Data Privacy and Security	Closely associated with international regulations (eg; Basel/BCBS 239, right to be forgotten etc) aspects of data privacy include simplistic concerns like assuring governed records are deleted correctly to more complex topics like using private data within machine learning algorithms that produce derived data. Data security is more conventional but equally important on the access controls, masking and encryption protocols for data.	In a decentralized data architecture, the aspects of dynamic data masking become more important (eliminating data on disk), and for polyglot data the policy controls for access may become 'multi-level' as data is comingled within data pipelines.
Remediation Workflows for Data Repair	Inevitably in any computer system data is at risk of corruption, human factors or machine factors can create corruption. Workflows for remediation will provide multi-level controls for teams of Data Stewards and Data Product Managers to repair data at the source, in a pipeline, or as output from a data product.	Data product thinking may change the approach to remediation, for example elevating the workflow SLAs to be governed directly by a Data Product Manager (rather than conventionally a DBA or Data Engineer)

ETHICAL CONSIDERATIONS IN DATA GOVERNANCE

Since the earliest days of Artificial Intelligence (AI) there has been a strong subculture dedicated to examining ethical use issues around technology as it gets smarter and more autonomous. Emerging in the 1990's, data governance for enterprise data has been largely focused on the pragmatic concerns surrounding corporate and regulatory compliance. But as data is becoming more crucial to business transformation, "Industry 4.0" and cyber-physical systems, there is a growing chorus of experts calling for more focus on the ethical considerations of data governance[li]. Unlike the strictly compliance-related aspects of data governance, ethical considerations can arise from questions about what organizations do (or don't do) with the data, how data is collected and how we use it. These ethical considerations can affect judgements about informed consent, anonymity, privacy and transparency. A modern approach to data governance may empower tools to capture and communicate these kinds of ethical considerations into the lifecycle of data stewards.

Regardless of the relative maturity of a data management architecture, data governance policies should be applied as an integral part of the system. In highly mature dynamic Data Fabric and trusted Data Mesh architectures, the role of data governance must be a first principle and not an afterthought. While no single IT tool can encompass all data governance needs, the tools remain an integral foundation of data governance in the Fabric/Mesh – which is one reason why the Oracle GoldenGate platform is an industry leader in trusted data use cases.

Oracle GoldenGate: Trusted Bridge to a Data Mesh

Oracle GoldenGate technology is unique in the industry because it brings long-standing trusted data replication capabilities right alongside modern decentralized microservices, polyglot payload capabilities, and stream processing for CTL (continuous transformation and loading) or time-series analytics.

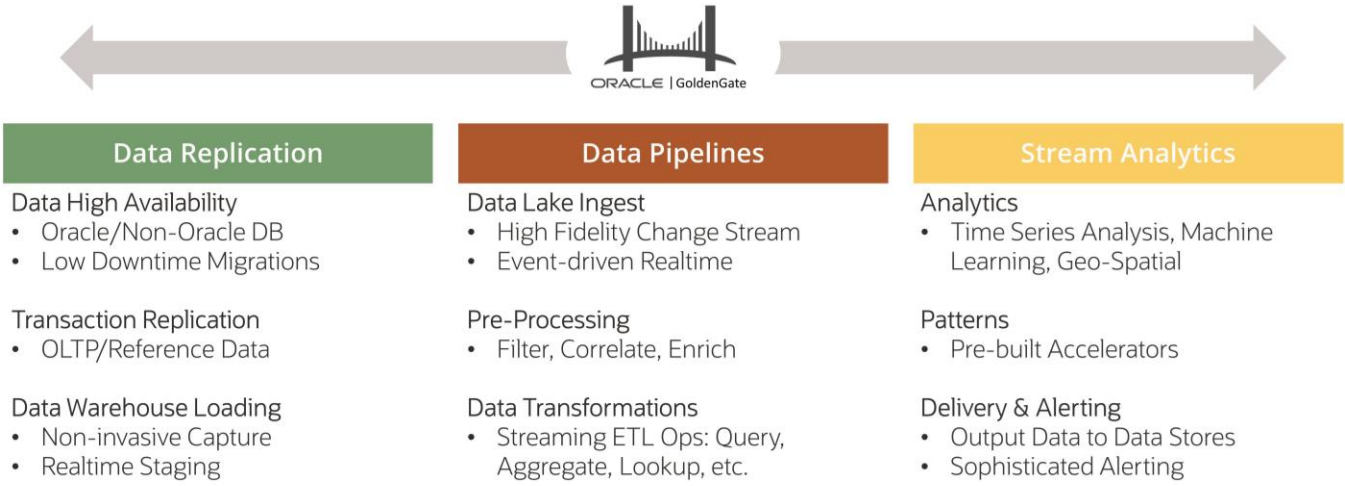


Figure 24: GoldenGate platform capabilities

For organizations that prioritize trusted data in real-time, Oracle GoldenGate is an ideal solution. For over 20 years, GoldenGate has been an established leader for trusted Data Fabric architectures, and the newest versions of GoldenGate are laying the foundation for trusted Data Mesh capabilities built around data product thinking, decentralized architectures, event-driven pipelines and polyglot data payloads. The GoldenGate platform is capable of change data capture (CDC), real-time trusted data replication, data ingestion, data pipelines for continuous transformation and loading (CTL) and a wide variety of analytics on streaming data.

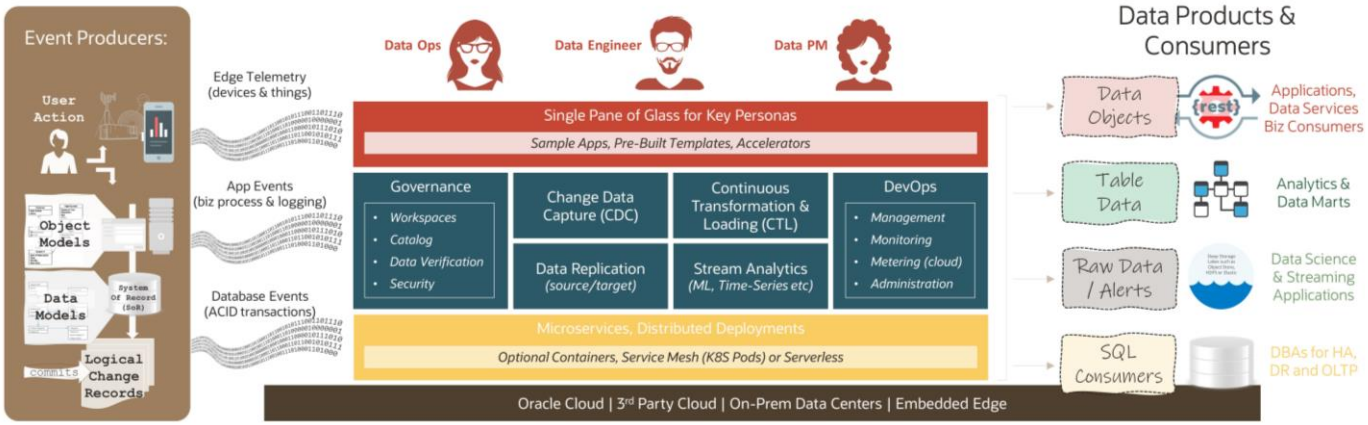


Figure 25: GoldenGate platform, logical architecture and key components

In the context of a dynamic Data Fabric and a trusted Data Mesh, the GoldenGate platform aligns to provide the following core capabilities:

- **High-Value Data Products** – provided by the GoldenGate platform itself
- **Enable External Data Products & Data Mesh Patterns** – GoldenGate as an enabling technology
- **Real-time Data, Microservices and Trusted Transaction Ledger** – architecture ready for the future
- **Self-Service, Low-Code and Strong Governance** – designed to be accessible to non-technical users

- **World-class Stream Processing, Open Core** – unique value, not just repackaging open-source tech

The combination of these capabilities provide a powerful bridge to help organizations transition from older monolithic data architectures towards next-generation dynamic Data Fabric and trusted Data Mesh designs.

Data Products Provided by GoldenGate

The GoldenGate platform can provide data products itself, as well as support other data products as an enabling technology. As described in the Data Product Thinking section of this document, data products can come in a variety of forms and structures. All data products (i) directly facilitate a business outcome through the use of data and (ii) are governed with formal KPIs and SLAs. Data products that are directly provided by the Oracle GoldenGate streaming data platform include:

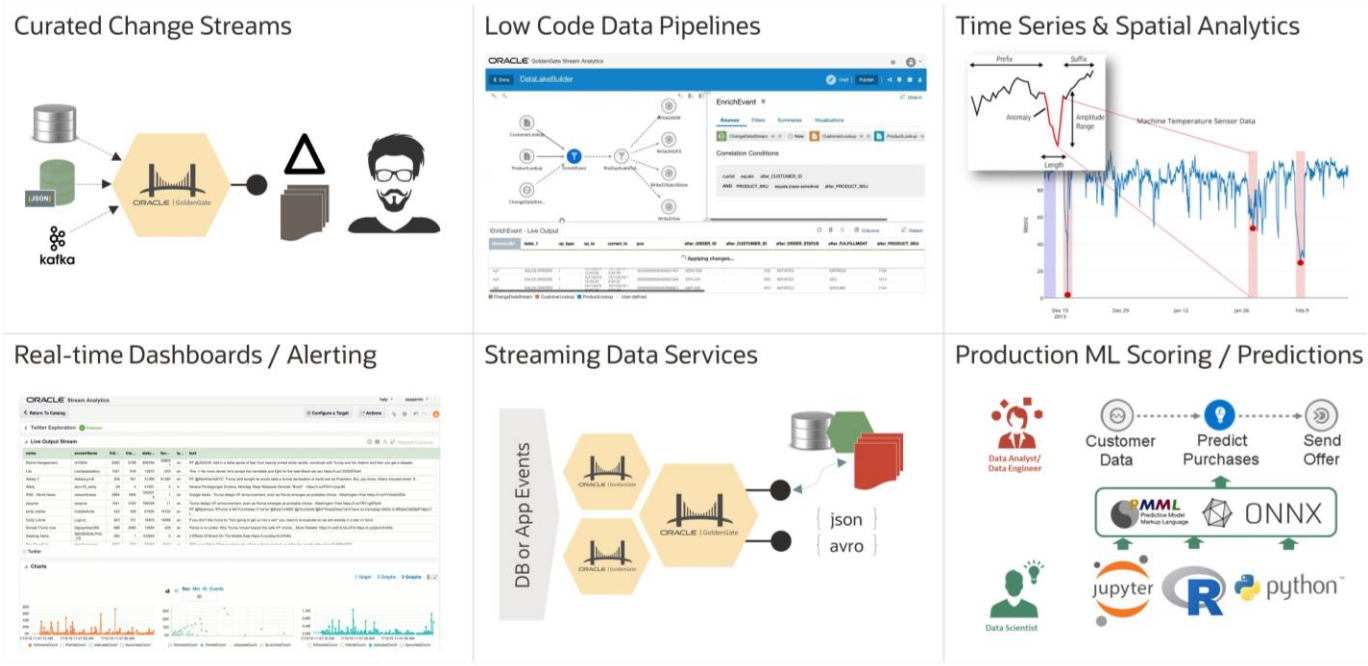


Figure 26: Data products provided directly from the Oracle GoldenGate streaming data platform

Curated Change Streams

Like the chess example given in the Enterprise Event Ledger section of this document, the change stream of data is like having access to the history of moves during a chess match – the change stream tells the full narrative of the truth and is essential for understanding the semantics of data. GoldenGate is able to detect changed data in a wide variety of relational data stores, non-relational data stores and messaging systems. Internal to GoldenGate there is a trusted event ledger that stores a local record of events and can be used in cases of catastrophic failures as a reliable recovery point for data. One of GoldenGate's microservices in the platform provides a 'distribution path' for change streams, these distribution paths are a curated collection of data objects. Registered consumers of the distribution paths will receive a continuous flow of change events for any objects in the path.

Data Pipelines

Data pipelines are the foundation of a 'data product factory' and a crucial enabler of an ETL or CTL data flow. In a real-time stream, the data pipeline accounts for the windowing functions, data transformations and analytic functions as well. Not all data pipelines are managed as full-fledged data products, but many might be. For example, if a data pipeline is consumed by external users, perhaps for a real-time analytic view or for change streams on master data objects which must be aggregated across many pipelines, then it could be a tightly governed data pipeline with established KPIs and SLAs.

GoldenGate data pipelines support the following capabilities:

WINDOWING FUNCTIONS	DATA TRANSFORMATION OPERATIONS	ANALYTIC PATTERNS
• Global Windows • Fixed Windows • Sliding / Tumbling Windows • Session Windows • Custom Windows • Custom Merging Windows • Group Timestamp Windows	• Filters • Aggregations • Joins • Expressions • Correlations • Queries / Functions • Custom Functions	• Time-series • Geospatial • Classification • Machine Learning (PMML, ONNX) • Apache Spark (native) • Rule Engines

Time Series Analysis

GoldenGate Stream Analytics supports ordered event processing down to the nano-second scale. Built-in patterns provide a no-code library that self-service data product managers can quickly bring into a data pipeline. Many of these pre-built patterns provide time-series function and analytics. Often, the time-series analytics are productized directly within the pipeline itself (eg; for alerting), but the output of time-series data can also flow into real-time visualizations suitable for consumption directly by business data consumers.

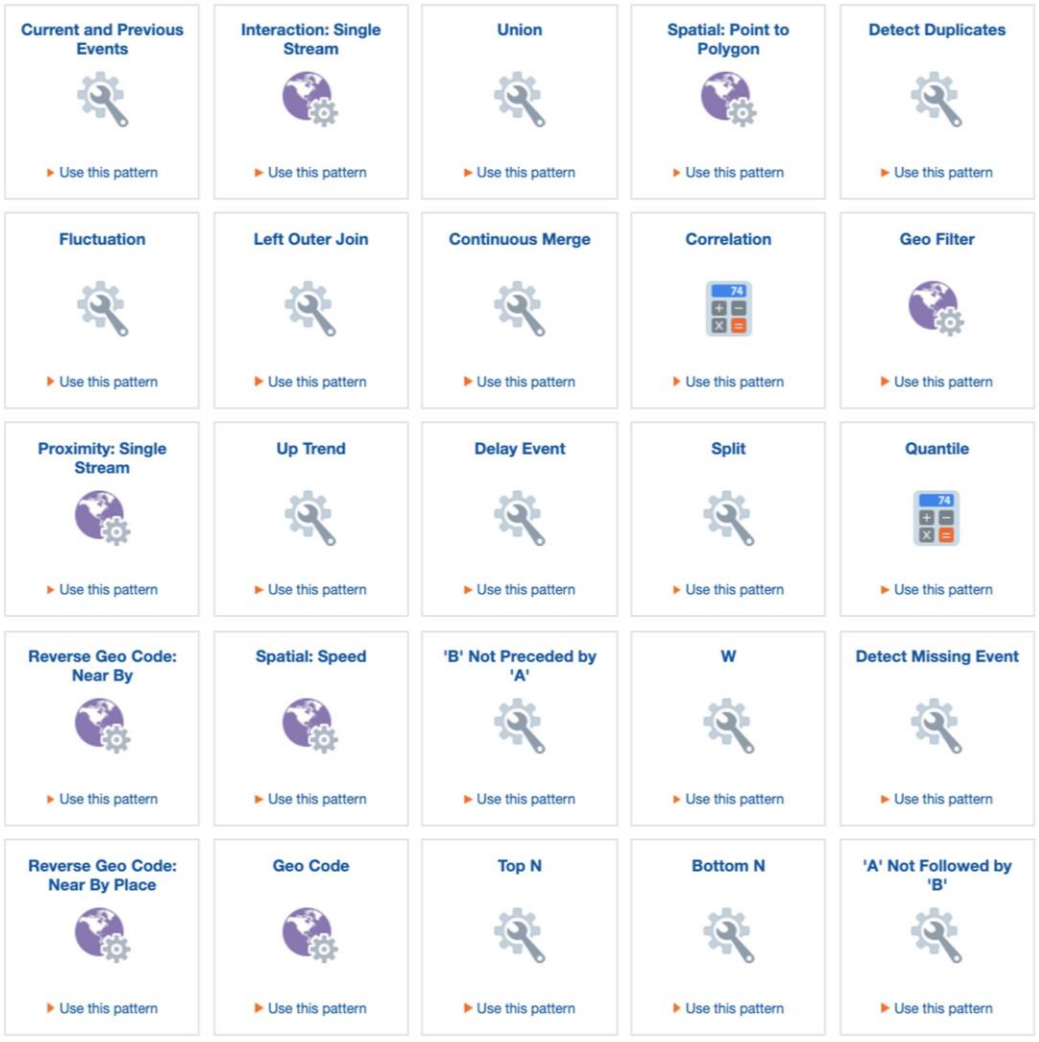


Figure 27: Time series analysis patterns browser inside GoldenGate Stream Analytics

Real-time Dashboards & Alerting

Data products in the pipeline include any data artifact which direct facilitates a business outcome and is governed under a specific KPI or SLA. This definition includes alerting pipelines that can notify business users via email, SMS/texting, a Kafka topic, or other gateway that might be used for alerting events. Likewise, this will include direct support for live dashboards. GoldenGate Stream Analytics supports a tightly integrated Data Catalog with native artifact types for data cubes and more than ten types of data visualization.

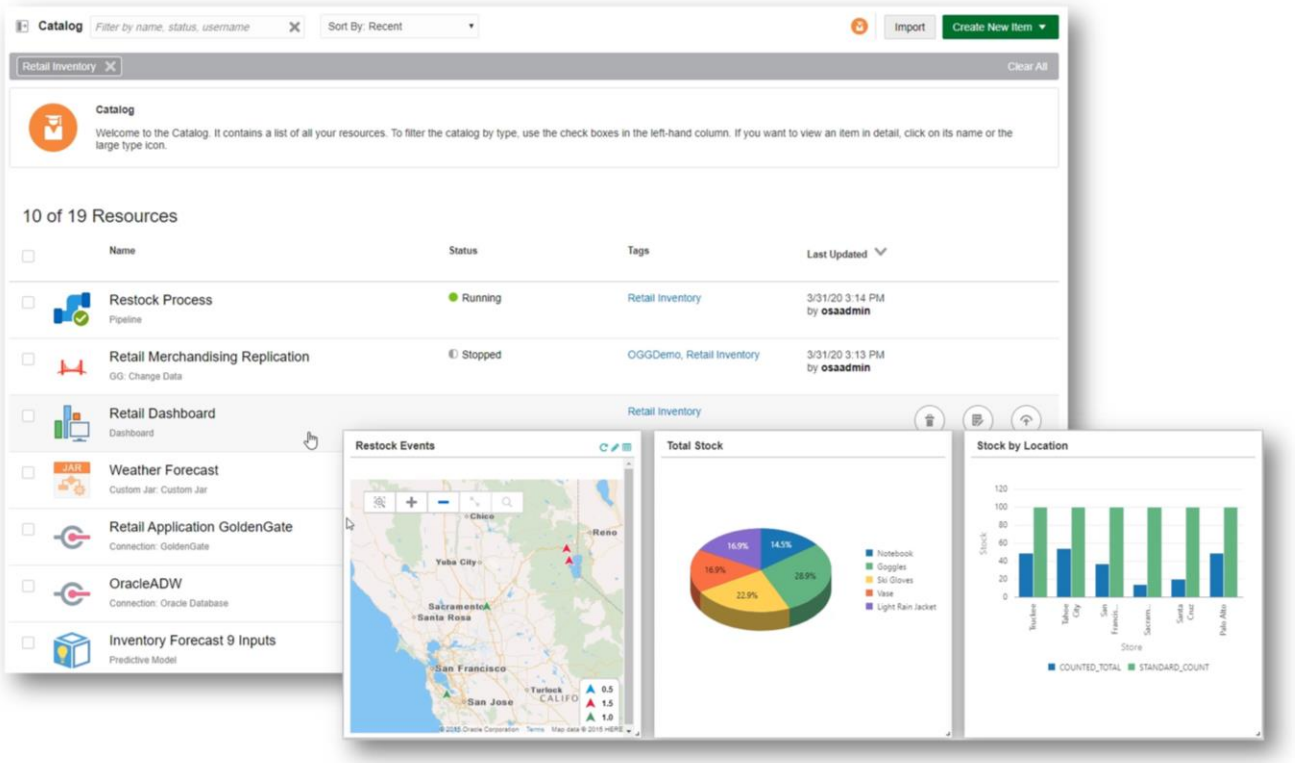


Figure 28: Data products can include dashboards, alerts and cubes – managed directly from an integrated catalog

Streaming Data Services

For many IT data lake implementations, the 'raw data zone' is a crucial staging area for downstream ETL and analytics. In a physical product supply chain, this can be thought of a similar to a distribution center (aka, fulfillment center), where products are staged prior to delivery into retail outlets or directly to consumers. In the physical world there are third party logistic providers that handle products from many producers and may co-locate services for many retail outlets. In the digital data product domain, Oracle GoldenGate is a productized service that is essentially handing the logistics of moving, routing and staging data products. The GoldenGate platform can continuously stream full data or changed data, as part of a real-time stream, a micro-batch, or larger batch processing (using time windows). These services are data products and they are often governed by a strict set of KPIs and SLAs.

GOLDENGATE SOURCES	GOLDENGATE TARGETS
• **Databases: Amazon Aurora PostgreSQL, Amazon Aurora MySQL, Amazon RDS for MariaDB, Amazon RDS for MySQL, Amazon RDS for Oracle, Amazon RDS for PostgreSQL, Amazon RDS for SQL Server, Azure Database for MySQL, HPE NonStop, IBM DB2 for I, IBM DB2 for z/OS, IBM DB2 LUW, MariaDB, Microsoft SQL Server, MySQL Database Server, MySQL Database Server, Oracle Autonomous Database, Oracle Database, PostgreSQL, SAP Sybase ASE**	• **Any supported Source may also be a GoldenGate Target** • **Native: Amazon Kinesis, Amazon Redshift, Amazon S3, Apache Cassandra, Apache Hadoop, Azure Data Lake Storage, Azure Event Hub, Azure Synapse Analytics, Cloudera Data Platform (CDP), Cloudera Hadoop (CDH), Confluent Platform, Datastax Enterprise Cassandra, Elasticsearch, Google BigQuery, Greenplum, Hortonworks Hadoop (HDP), Java Message Service**

GOLDENGATE SOURCES	GOLDENGATE TARGETS
• Non-Relational: Cassandra, MongoDB, Kafka Connect, Java Message Service (JMS) • Applications: Oracle Fusion ERP, Oracle EBS, Oracle Retail Cloud, Oracle Argus Cloud, Oracle Transportation Management • Via Oracle Integration Cloud and Kafka Connect: Salesforce, Marketo, Workday, MQTT, Jira, Oracle NetSuite, Oracle Commerce Cloud, Oracle HCM, Oracle Responsys, Oracle Engagement Cloud, LinkedIn, SAP Success Factors, Twillio, Twitter, and 120+ others	(JMS), Java DB Connector (JDBC), MapR Hadoop, MongoDB, Netezza, OCI Autonomous Data Warehouse, OCI Object Storage, OCI Streaming Service, Oracle Cloud Object Storage, Oracle NoSQL, Oracle TimesTen, SAP Hana (JDBC), Snowflake, Teradata, Vertica (File/JDBC) • Via Apache Kafka: any • Via Oracle Integration Cloud: any • Via Oracle IoT Cloud: any

Note: always check with your Oracle representative for latest set of supported technologies

GoldenGate is a trusted data provider that maintains dependable ACID consistency for relational transactions and also a polyglot streaming data service that can format data products in a variety of syntax including: Native formats of targets, HDFS Sequence Files, Delimited Text (both Row & Operation modes), JSON (both Row & Operation modes), Avro (both Row & Operation modes), XML, Parquet and ORC (Optimized Row Columnar).

Production ML Scoring/Predictions

Oracle GoldenGate is an ideal platform to put machine learning (ML) models into production. While some ML models are mainly to support regression analysis on historic data, other ML models are designed to support scoring and recommendation capabilities on real-time data transactions. The GoldenGate platform allow for ML models to be integrated directly into real-time data streams so that individual events can be scored according to pre-defined models. For example, ML models in a streaming pipeline could include: next-best-action recommendations for customers, scoring algorithms that identify customer affinity, or associative ML for identifying potential fraud.

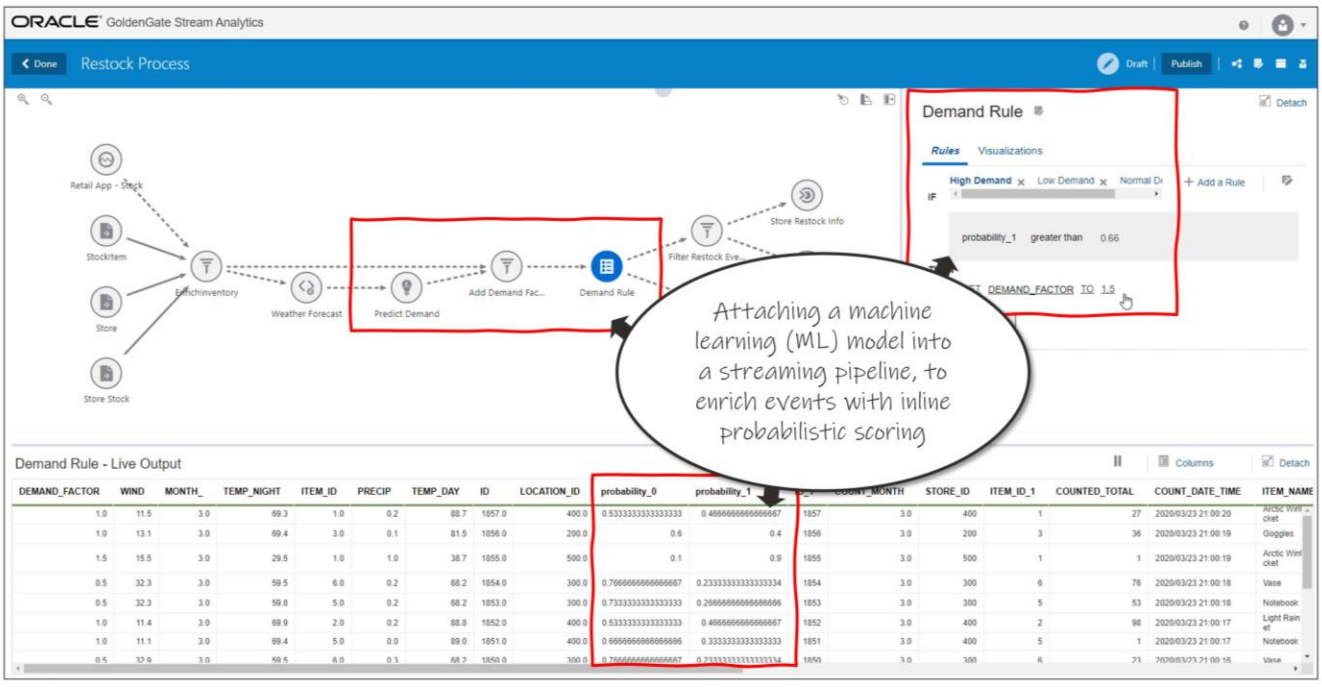


Figure 29: Data products can be governed ML models used to drive predictive scoring in real-time

Data pipelines that run ML models could be used for advanced filtering (eg; to reduce unneeded events produced by Internet of Things (IoT) devices) at the Edge or on an IoT Gateway. Event ledgers that produce events could be deployed in a microservices Data Mesh (eg; as GoldenGate distribution paths, Apache Kafka topics or even Java Messaging Service messages). More centralized deployments could simply run in any public cloud.

Microservices, Cloud-Native Architecture

Oracle GoldenGate was the first data replication technology to make the transition to a microservices architecture. In 2017 at the Oracle Open World event in San Francisco, Oracle began to preview the next-generation architecture. Unlike other vendors who may simply place REST APIs on top of existing monolithic tools, the GoldenGate architecture was completely refactored to eliminate monolithic dependencies and embrace a fully-encapsulated microservices mesh style architecture.

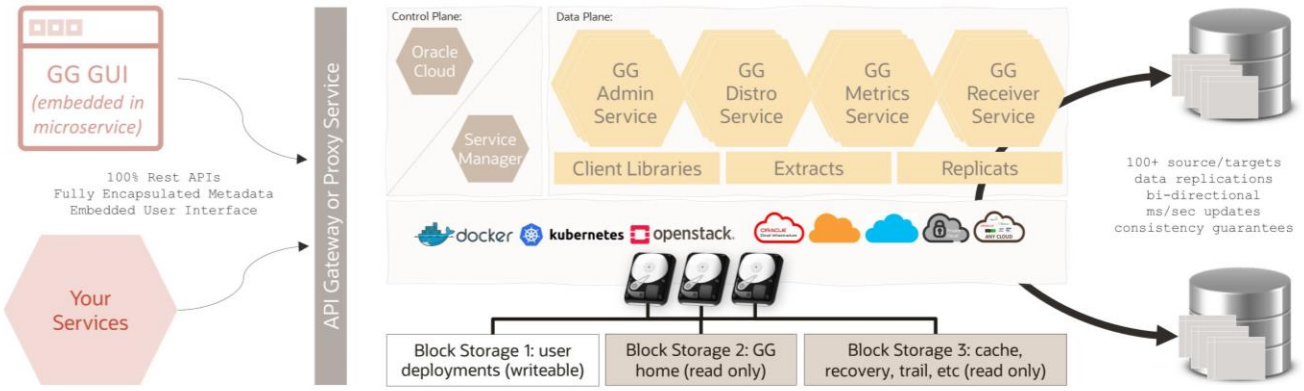


Figure 30: GoldenGate microservices components

Within GoldenGate on-premise deployments, the GoldenGate Service Manager acts as a control plane for other GoldenGate microservices deployments that can be associated with it. GoldenGate Administration microservice is where Data Ops/DBAs manage Extracts/Replicats. The Distribution microservice is where change data paths can be published for secure distribution across the WAN/LAN. The Receiver microservice will receive those distribution paths and the Metrics service provides self-service access to all telemetry and metrics.

When running inside the Oracle Cloud as a native, fully-managed service the control plane is the Oracle Cloud itself. This fully-managed GoldenGate cloud service comes with many benefits including pay only for what you use, auto-scale with workloads, automatic backup, recovery and automated patching.

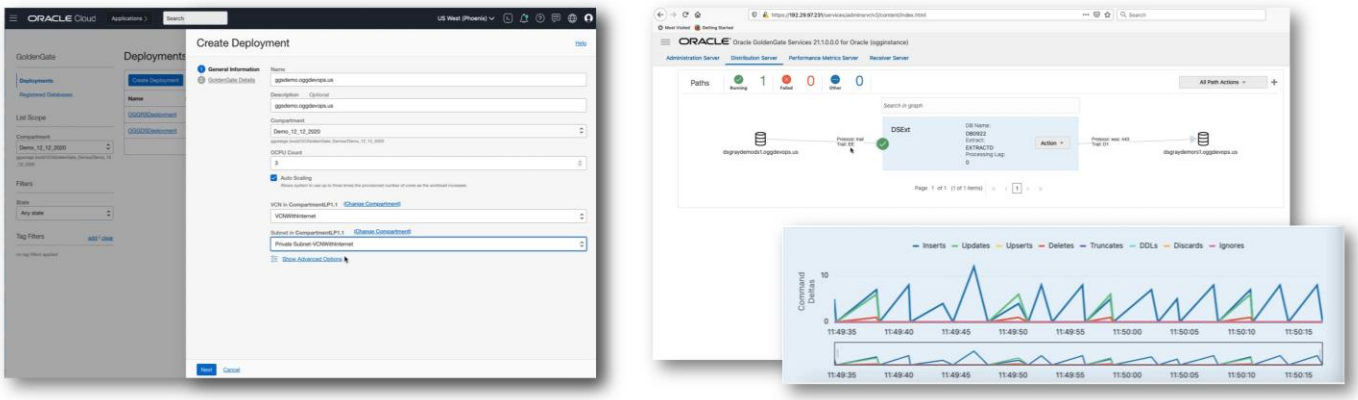


Figure 31: GoldenGate fully-managed from the Oracle Cloud and a graphical user experience from multi-cloud or on-premise

Real-time, Event-driven Data Movement

GoldenGate can capture events from source systems directly, regardless of whether GoldenGate is installed on the same host, on the same network, or event across the planet in an altogether different data center. These events are detected as soon as the source systems send them (it is a real-time solution, not a polling-based solution) and GoldenGate will immediately send the events to any registered target systems (there is no micro-batching unless the developer specifies it). Additionally, GoldenGate can be deployed as a mesh of microservices across the WAN by

utilizing the Distribution and Receiver microservices. These microservices are extremely useful since they provide network and protocol decoupling from the source/target client frameworks. Some sources/targets can be latency sensitive, causing anomalies when streaming events must flow over longer distances or unreliable networks. The Distribution and Receiver services provide more tolerant, flexible and secure routing using secure sockets and certificate-based mutual TLS.

GoldenGate Transaction Ledger, the Trail Protocol

Like Apache Kafka, Oracle GoldenGate is driven by its internal transaction ledger. In GoldenGate this is called the 'trail'. When on disk (eg; used for recovery purposes) it is called the Trail File, and when in memory being transmitted via the network it is functionally a Trail Protocol. GoldenGate Trail is a canonical format, in that regardless of what source data events are being consumed, they are re-formatted into the Trail format while GoldenGate is processing. This allows GoldenGate to provide very high trust levels since the semantics of the Trail format are tightly integrated to GoldenGate and have been used for more than 20 years as a high availability (HA) and disaster recovery (DR) format for many different kinds of databases and data types.

GoldenGate maintains referential integrity of database payloads, as it is tied in closely with the commit logs of the source databases themselves. The semantics of each database's commit strategies can be slightly different, for example they way MongoDB supports ACID transaction is different from Oracle Database or PostgreSQL. GoldenGate's integrations are designed to bring the highest levels of trust so that data consumers and data products are correct and trusted. Internal to GoldenGate Trail, every transaction carries a unique change number (SCN/CSN) which is used to group transactions correctly as well as to be sure that they are applied in the correct order.

Downstream from GoldenGate, it can integrate with many different kinds of non-relational targets (eg; cloud object storage, Apache Kafka, ElasticSearch, etc.) and developers can utilize the GoldenGate Trail metadata to enrich their custom data pipelines. The GoldenGate data pipeline technologies are natively integrated with Distribution microservices and Trail metadata so that CTL pipeline processing can also be trusted with correct and consistent data.

Data Fabric and Data Mesh Patterns Enabled by GoldenGate

Modern enterprise data architecture will have many requirements for different kinds of data and will make use of many different vendor technologies to provide end-consumer data products. The Oracle GoldenGate platform can be used to enable solutions with all types of data products aligned to different vendors and cloud providers.

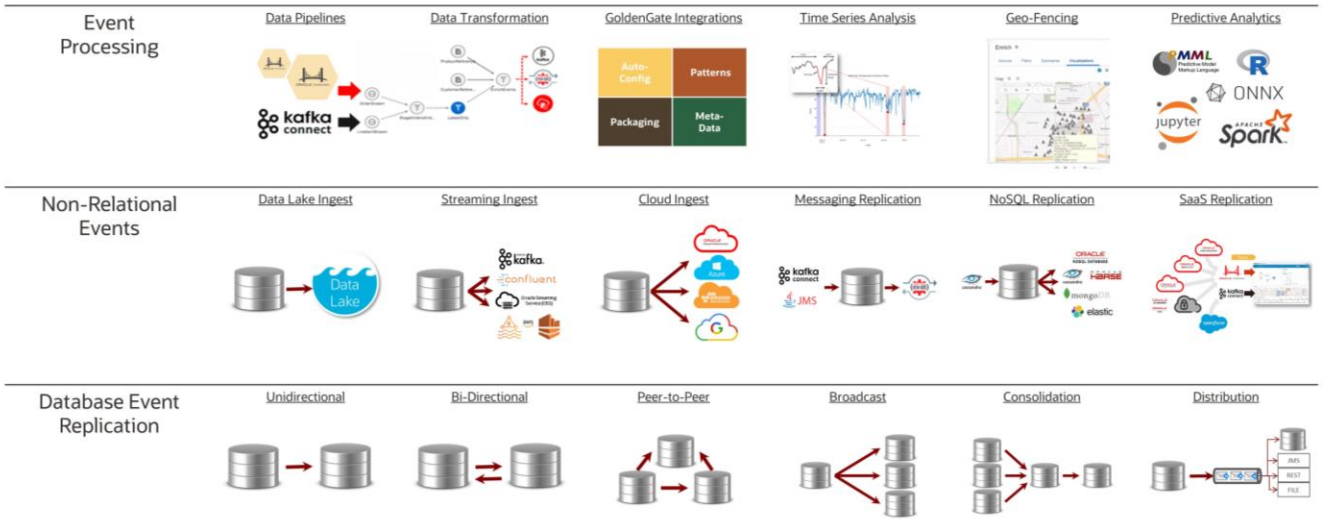


Figure 32: Enabling a wide variety of data product solutions with the Oracle GoldenGate platform

Database event replication is the historic DNA of GoldenGate and there are many 1,000's of customers using GoldenGate for this kind of Data Fabric. Common use cases for database synchronization, multi-active high

availability (HA), and real-time ingestion to data warehouses are the foundation use cases for GoldenGate Data Fabric. Customers using the technology for database event replication that have moved to the GoldenGate microservices edition are already prepared for the shift towards a trusted Data Mesh.

Non-relational data replication provides the backbone for polyglot streaming services, capturing and moving data payloads like XML, JSON and Avro. Core use cases in this area can be straight-forward, for example to ingest data events into data lakes, cloud storage and messaging platforms like Apache Kafka. Other use cases are more complicated and take advantage of the wider platform to capture events from NoSQL stores, Kafka or directly from certified SaaS cloud applications. Trusted, polyglot event streams are at the heart of what GoldenGate enables.

GoldenGate's own built-in event processing engine is powerful and fast, but sometimes customer's have existing investments or may make other choices. The core GoldenGate replication frameworks can be directly integrated with non-Oracle event processing and analytics tools from open source, other vendors or public cloud providers.

Microservices Transaction Outbox, CQRS and Event Sourcing

One of the perennial struggles that microservices application developers have is with the data[lii]. Certain microservices dogma says that developers should bring all the data logic into the component tier of the microservice, the line of thinking is to minimize coupling with the underlying databases[liii]. Because of this approach, many design gurus essentially push the coordination and consistency of data transactions directly into the hands of the application developer. There are a myriad of design patterns to support this approach, and most all of them require the application to revert to an 'eventually consistent' data policy – since there is no common data store and consistency/rollbacks are handled by the developers themselves.

Transaction Outbox is design pattern has emerged to simplify communications and provide greater data consistency. In its original form[liv] the requires developers to write two updates in their application, and possibly an entirely separate 'message relay' service. But many have noticed that this pattern can easily be combined with changed data capture (CDC) and replication tools like Oracle GoldenGate to really simplify the pattern. By utilizing CDC, the base table and any filtered columns becomes the 'outbox' natively, and the message relay is replication framework of choice. In this way, microservices events (eg; commit events on the data store) can be propagated to a broker, a data lake or directly to any other microservices.

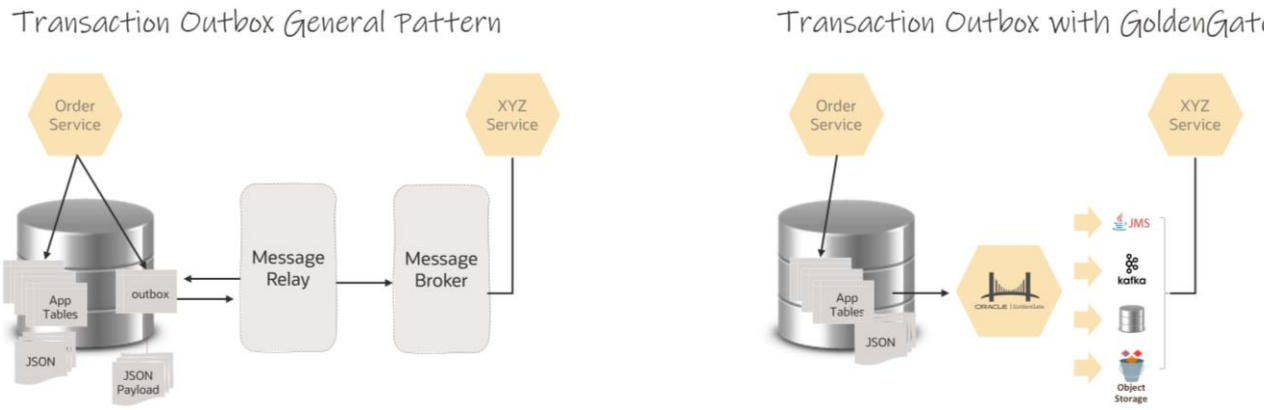


Figure 33: Implementing a Transaction Outbox pattern with Oracle GoldenGate

Using CDC and data replication with microservices is a significant reduction in complexity and a far superior way to achieve reliable consistency and data correctness in distributed architectures. Without the use of a change detection tool, the microservices applications would ordinarily have to implement some kind of polling mechanism. There are many disadvantages to polling, including the fact that the data store will constantly be saturated with queries that are looking for changes. Another big disadvantage of polling is that it is not real-time, it would be more of a micro-batch solution with changes being collected every few minutes (or whatever the polling frequency is set to). In comparison,

CDC tools are driven by the commit events and there is typically very little overhead on the data store itself (in fact, there are often techniques to eliminate any additional overhead on the database store).

Naysayers to the Transaction Outbox approach using CDC tools may not like that there is a tight-coupling between the CDC tool and the database, but this is central to the benefits of being able guarantee that fully consistent data is captured and replicated – this trade-off is well worth tight coupling at the CDC layer.

Extending the Transaction Outbox pattern to other common microservice data tier patterns such as CQRS (command query responsibility segregation)[lv] and Event Sourcing[lvi] can yield similar benefits – reduced complexity for the developer and superior data consistency guarantees.

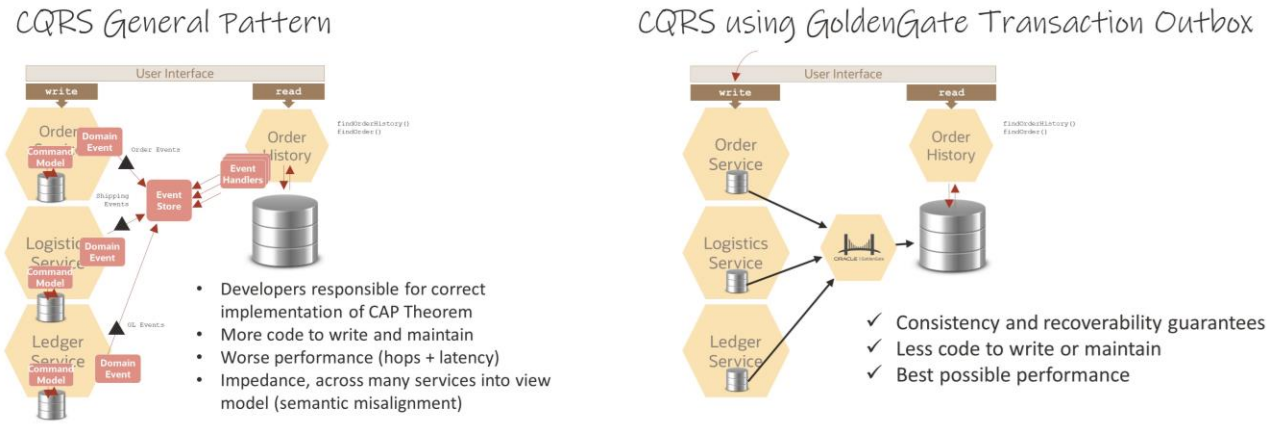


Figure 34: Implementing a CQRS pattern with Oracle GoldenGate Transaction Outbox

Event Sourcing

The subject of event sourcing is complex and many books have been written on the overall pattern, but in general there are four distinct ways that event sourcing is used in microservices design:

Figure 35: Differing ways that Event Sourcing patterns are used with microservices architecture

As you can see, there are a variety of ways that microservices developers use event sourcing. But, if we step back and look at event sourcing from a wide architectural view, it mirrors the same kinds of patterns we explored earlier in this document as part of the Enterprise Event Ledger section. In that context, microservices are just one specialized kind of event producer among many different kinds of enterprise systems.

An enterprise Data Fabric is by definition scoped to cover all enterprise data producers. In a dynamic Data Fabric, the microservices application tier should be a fully supported and first-class event producer supported within the fabric. This document has noted in several sections that there does not need to be, and perhaps there cannot possibly be, a single event ledger than can do all things. Thus, from time-to-time there might be unique requirements from a particular microservices application that requires a specialized event store that is optimized for microservices applications (many of which natively implement CQRS patterns, for example).

However, a general-purpose dynamic Data Fabric or trusted Data Mesh will accommodate different ledger technologies and should encourage the use of patterns that support trusted, consistent and correct data. That is why

the Transaction Outbox pattern with Oracle GoldenGate is a powerful option for microservices – it can align the microservices technology stack with an enterprise Data Fabric, simplify the development of the core microservices application, and greatly improve the system-level data consistency of distributed data transactions. The fact that Oracle GoldenGate itself is a set of microservices enables the same DevOps and CI/CD best practices to apply to GoldenGate components, thereby providing the best of both worlds: data consistency with high agility.

World-class Stream Processing

As a type of enterprise software, stream processing has been around for many decades. But only since 2015 has the widespread availability of open-source big data platforms made event stream processing practical for mainstream IT organizations. The use cases for stream processing are broad and varied.

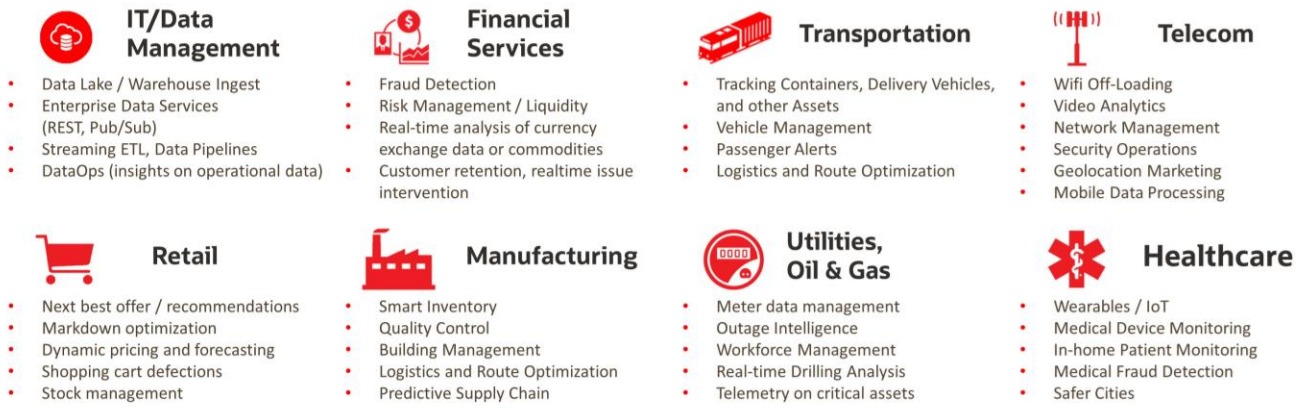


Figure 36: Common use cases for event stream processing

However, the impact of event stream processing is quite likely to be felt more universally than these use case specific examples. More broadly, streaming will forever alter the multi-billion dollar market landscapes of Cloud, Big Data, and Data Management overall. It is the underlying enabling technology of streaming that is already shaping the service mesh and serverless cloud products. Likewise, the disciplines of big data and data management overall a sharply pivoting towards empowering developers to work directly with the data streams and event logs. Thinking ahead just five years it is difficult to imagine Cloud, Big Data, and Data Management solutions without event streaming at the heart of it all.

Stanford Collaboration with Continuous Query Language (CQL)

The Oracle journey with event streaming goes all the way back to 1977, even the earliest versions of the Oracle Database were centrally defined by transaction events, known as the 'online redo log'. But databases typically treat the event logs as internal artifacts of the engine, they are not designed for direct end-user interactions. Fast forward to mid-2000's and Oracle began collaboration with Stanford University and ultimately created a SQL-based proposal (Continuous Query Language – CQL) for querying event streams. This paper[lvii] laid out the base principles of a declarative programming model as well as a number of computational semantics needed for working with streaming data and different types of windowing functions.

Complex Event Processing (CEP)

One key result of the Stanford collaboration was the emergence of the first Oracle Complex Event Processing (CEP) engine. The Oracle CEP engine implemented the concepts defined by CQL declarative language and materialized the engine into Oracle's middleware technology stack of that era – the BEA WebLogic application server and Coherence data grid for Java. Using this technology stack, the Oracle CEP engine was able to scale more than a million events per second on a single 2010 era Exalogic node. More impressively, the core engine and CQL semantics continued to evolve and support a very functionally rich set of data processing semantics for time series, geospatial events, and complex multi-stream correlations.

Shift to Open-Source Platforms

By 2015 Oracle noticed the shift happening with event stream processing in big data and decided to align the CEP engine with Apache open-source frameworks and ultimately to Oracle GoldenGate.

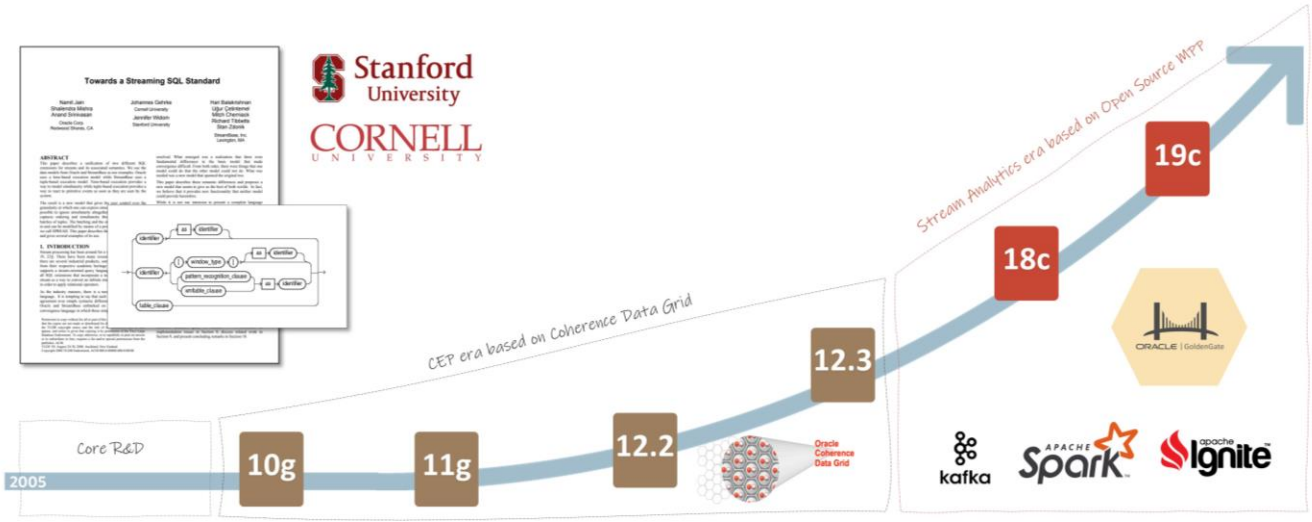


Figure 37: Three major eras of development for Oracle stream processing core engine

Early investigations to refactor the Oracle CEP engine looked at a variety of Apache technology. Apache Storm was a powerful and proven (Twitter-scale) option, but already felt stale in 2015/16. In this timeframe Apache Flink was an obscure project rooted in the German university system. Apache Spark was already a roaring success and although the streaming frameworks of Spark were (and still are) weak, it didn't matter since the Oracle CEP engine was to provide the actual power of the streaming semantics. Ultimately, Oracle decided to leverage the MPP foundations of Spark as a host for the CEP engine (implementing the underlying CQL semantics first explored with Stanford University). Today, the GoldenGate Stream Analytics pipelines can run in any Apache Spark frameworks and the core engine implements its own powerful semantics for stream processing.

		Common open-source alternatives...		
[best] ●●●⊙○ [worst]	GoldenGate Streaming	Spark Streaming	Apache Flink / Beam	Confluent KSQL
User Experience				
Low Code Development *(with built-in patterns/accelerators)*	●	○	⊙	⊙
Interactive/Live Edits *(browser based, view changes immediately)*	●	○	●	⊙
Built-in Live Dashboards *(event-driven charts/graphs)*	●	⊙	⊙	○
Core Streaming Semantics				
What is Being Computed *(transforms, joins, flatten, statefulness etc)*	●	◉	◉	◉
Time Windows *(global, fixed, sliding, tumbling, custom etc)*	●	◉	●	●
When in Processing Time *(triggers – event, time, count, timers, etc)*	●	⊙	◉	⊙
How do Refinements Relate *(discarding, accumulating, retracting)*	◉	◉	◉	⊙
Analytics				
Robust CEP Capabilities *(complex event correlations, native time clock)*	●	○	⊙	⊙
Geo-Fencing & Spatial *(lat/long, built in maps, custom map tiles, etc)*	●	○	○	○
Machine Learning *(native scala, ONNX, PMML, python support etc)*	◉	◉	◉	⊙
Time Series Analysis *(built-in interval patterns, thresholding etc)*	◉	⊙	⊙	⊙
Other Features				
Backpressure *(dynamic ingest per pipeline)*	Automatic	Custom	Custom	Custom
State Management *(automation across streams & native cache)*	Data Grid	Custom	RocksDB	RocksDB
Data Consistency *(OLTP Change Events, Inserts/Updates/Deletes)*	Automatic	Custom	Custom	Custom
GoldenGate Stream Type *(aware of SCN/CSN, transactions, order, etc)*	Native	Custom	Custom	Custom

Figure 38: High level comparison between GoldenGate Stream Analytics and some common open-source streaming platforms

Around the same time the CEP engine was refactored to run inside Apache Spark, the other macro-trend sweeping the IT world was low-code development. Thus, the decision was made to further abstract developers away from the underlying programming (Java/Scala and CQL) and to provide a graphical user interface (runs in a web browser) that supports point-and-click development of stream processing pipelines and analytics. At this time, Oracle also began supporting a rich set of pattern accelerators designed to help jump-start the developer experience. Today, the Oracle GoldenGate Stream Analytics is undoubtedly one of the simplest to use streaming tools, it comes with pre-built sample applications and can be launched from Oracle Cloud in less than 5 minutes.

Oracle is committed to a world-class streaming platform designed to be easy to use, trusted for enterprise data and with the most powerful underlying core engine for processing complex streaming data at scale. For our Oracle Cloud customers, the end-user experience should be as seamless and serverless as possible, with just the no-code GUI used for developing streaming applications. On premise customers will have the option to deploy streaming inside open source Apache frameworks and alongside the highly modular GoldenGate microservices.

Dynamic Data Fabric and Trusted Data Mesh with Oracle Cloud

As noted in the first section of this document, the broad concept of a Data Fabric encompasses other Oracle technologies and services. Across the wider Data Fabric ecosystem, Oracle is an independently recognized[lviii] leader and the full portfolio includes:

CLOUD-NATIVE, COMMON PLATFORM DATA FABRIC	BEST-OF-BREED DATA FABRIC FOR MULTI-CLOUD & ON-PREMISE
• **Self-Service ETL for Analytics & Autonomous DB** • **OCI Data Catalog, OCI Data Integration, OCI Data Flow** • **OCI GoldenGate and Stream Analytics for OCI** • **Integration Cloud and Oracle Cloud SQL**	• **Oracle Data Integrator (w/ETL, Quality, Messaging)** • **Oracle GoldenGate and Stream Analytics** • **Oracle Big Data SQL (Data Federation)** • **Oracle Data Visualization (Data Preparation)**

The main scope of this document has centered around a 'dynamic' Data Fabric which is principally focused on real-time, event-driven and streaming data use cases. Since the Data Mesh definition is already narrowed to focus on event pipelines, the additional area of emphasis that Oracle brings is on 'trusted data' which includes several aspects of data governance as well as transaction guarantees for ACID-level data consistency.

All the technologies discussed thus far in the document are widely available to run on customer provided infrastructure, non-Oracle public clouds and service mesh technology like Docker and Kubernetes. But now we can look at the services and capabilities that are native to the Oracle Cloud Infrastructure (OCI), and what it would take to operate a full-scale Data Fabric or Data Mesh in any of the 25+ OCI data centers around the globe.

DATA FABRIC PRODUCTS, SERVICES, AND CAPABILITIES IN ORACLE CLOUD INFRASTRUCTURE (OCI)	
Data Preparation	Innovative AI-based engine that utilizes Natural Language Processing (NLP) and a graph knowledge-base (linked open data) for machine-assisted data wrangling, packaged with Oracle Analytics
Autonomous Data Tools	A collection of tightly integrated capabilities that are bundled with Autonomous Data Warehouse, this includes the award-winning Oracle Data Integrator E-LT tool for batch data transformations
OCI Data Integration	Serverless ETL capabilities with low-code user interface and advanced workload optimizer
OCI Data Flow	Serverless Apache Spark execution, tightly integrated with OCI Data Integration
OCI Streaming Service	Native event messaging framework, provides Kafka-like capabilities across Oracle Cloud
OCI GoldenGate	Fully managed GoldenGate native to Oracle Cloud, provides auto-scale capabilities and many other advanced automations that are unique to OCI native services

GoldenGate Stream Analytics	Complete capabilities of GoldenGate Stream Analytics available with per-hour OCI cloud credits, pre-packaged with all necessary engine components and flexible enough to run workloads in any Spark nodes
Integration Cloud	Industry leading application integration and business process automation, this is an ideal component of the Data Fabric for integrating with SaaS and Social Media data events of all kinds
IoT Cloud	Industrial IoT solution for smart manufacturing, predictive maintenance and logistics, this is an idea component of the Data Fabric for integrating with edge devices, IoT gateways and managing 'digital twins'
OCI Data Catalog	An integral OCI service for overall governance and the management of business data domains

Each of the services mentioned above share the same OCI control plane and a long list of platform service capabilities that enable the Oracle Cloud to function as a holistic platform.

High Level Blueprints

When operating in the Oracle Cloud there are blueprint reference architectures available to guide customers in choosing the correct services for particular use cases. These architectures also provide useful best practices and may also offer Terraform automation to help customers materialize the solution within OCI. The following high-level blueprints are shown here to illustrate the Data Fabric discussion. To browse all the most recent reference architectures, visit the OCI architecture center (https://www.oracle.com/cloud/architecture-center.html)

Dynamic Data Fabric, Trusted Data Mesh for Operational Data

Described earlier in this document as part of the "Aligning Operational and Analytic Data Stores" section, operational data use case for Data Fabric or trusted Data Mesh could include:

- **Online Database Migrations** – for no disruption to online applications
- **Database High Availability (HA) and Disaster Recovery (DR)** – for maximum uptime across regions
- **Reference / Master Data Synchronization** – lookup tables and golden records kept perfectly in sync
- **Polyglot Data Integration** – sync data across disparate OLTP databases or Microservices (CQRS, etc)

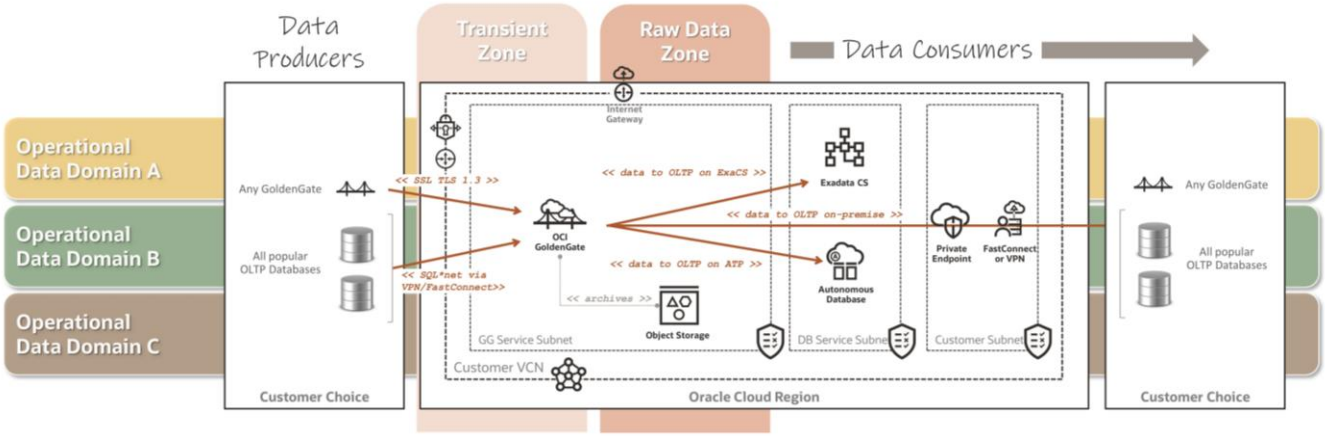


Figure 39: OCI GoldenGate in a mesh with on-premise GoldenGate microservices and operational data stores

The blueprint above is showing the OCI GoldenGate service (in its own dedicated service subnet) integrating with several different sources and targets. OCI GoldenGate can directly integrate with any other GoldenGate deployment (running on premise, or another cloud) and 100's of different combinations of source technologies. This integration allows real-time database transactions to continuously flow through the OCI GoldenGate deployment into downstream data consumers such as Exadata Cloud Service, Autonomous Database, or any number of supported external data stores, big data, NoSQL engines or messaging platforms. In this depiction, the use cases are operational

in nature and there would most likely only hours/days worth of log ledger recorded in durable storage located in the Transient Zone. In the Raw Data Zone, the OCI GoldenGate service can archive Trail data for customers who require long-term storage for governance, audit or compliance. GoldenGate can replicate data transactions from any Operational Data Domain, and depending on the best practices defined by 'data product thinking' guidelines it is possible to assign specific GoldenGate deployments to named data domains – thereby functional partitioning the services by domain. (in Oracle Cloud, one way to do this is with OCI Compartments[lix])

Dynamic Data Fabric, Trusted Data Mesh for Analytic Data

Much of this document focuses on supporting analytic use cases for Data Fabric or trusted Data Mesh, a brief recap of key use cases include:

- **Streaming Analytics / Applications** – often for decision support and automation (eg; next best action, etc)
- **Data Warehouse and Marts** – the streaming use cases provide data ingest and CTL services
- **Data Lake and Data Science** – high fidelity system events are pure gold for most data scientists
- **CTL and Data Integration** – continuous transformation and loading for any data management use case

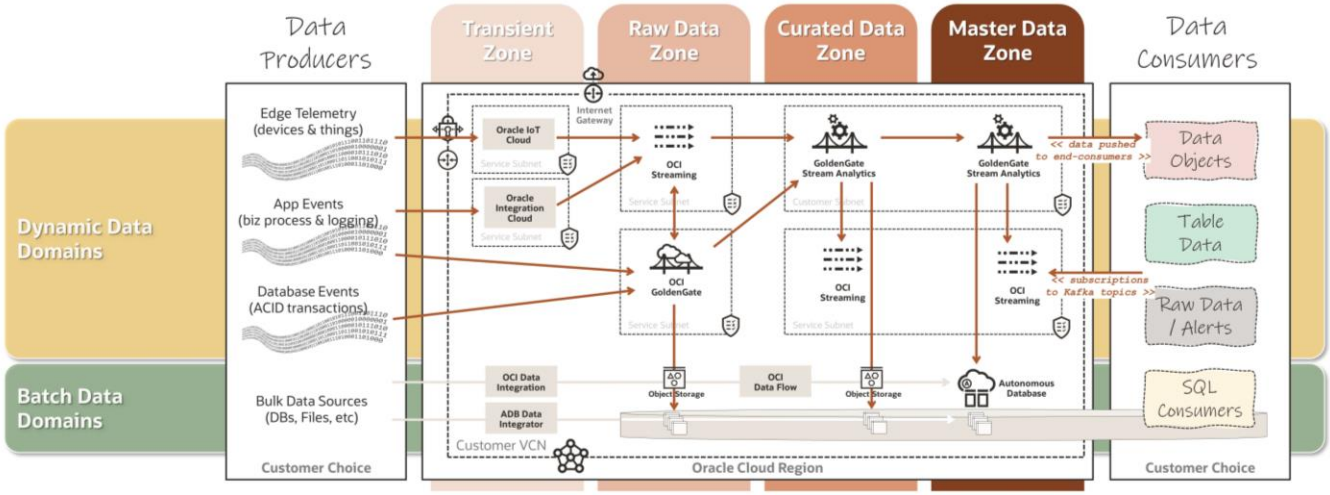


Figure 40: OCI Components in a trusted Data Mesh, focus is on streaming and real-time services (batch services shown for reference)

The blueprint above is showing a variety of data producers (on the left) that can producing events in real-time. Oracle Cloud has several options for Transient Zone (ingestion) including Oracle IoT Cloud for IoT events, Oracle Integration Cloud for SaaS Application and social media events. The Raw Data Zone can be configured to store days/months of data (acting as a enterprise event ledger). Here we see OCI Streaming for Kafka-like messaging and OCI GoldenGate for trusted data transactions. Both OCI Streaming and OCI GoldenGate are integrated with GoldenGate Stream Analytics, which can be used to prepare and transform data into Curated Data Zone and the Master Data Zone. Data in any of the zones can be persisted in Oracle Database, OCI Object Storage or OCI Streaming and the choice of which to use will depend on the use case. For streaming use cases, the canonical events (eg; JSON payloads that have been curated or mastered) may typically be placed in governed OCI Streaming topics. Downstream data consumers can selectively subscribe to OCI Streaming topics, or GoldenGate can directly push data into a wide variety of target systems – making the whole event chain push-driven.

In the depiction above, focus is clearly on the dynamic, real-time streaming components but some batch domain processing is shown for completeness. OCI Data Integration can batch data in from a variety of source data stores, landing the data and applying a range of batch data transformations. Optionally, OCI Data Integration can use OCI Data Flow service to execute Apache Spark jobs in batch. For database centric use cases, the Autonomous Database (ADB) directly supports the use of Oracle Data Integrator, which can be used to batch data directly into ADB and then apply a series of push-down, E-LT data transformations within the database itself.

There are many possible variations and interaction models which cannot be shown in a single picture. An enterprise Data Fabric or Data Mesh use case will likely have some streaming events being sent into batch processing flows. Likewise, some batch processing (eg; database loads) can trigger downstream real-time data pipelines. All of the OCI tools and services are designed to run from the same control plane, security policies and networking infrastructure so that all these Data Fabric/Mesh combinations are achievable.

Oracle Cloud Data Catalog

When running a Data Fabric or Data Mesh from Oracle Cloud, it will be the OCI Data Catalog that brings your data domains together for simplified governance. Depending on your 'data product thinking' heuristics, your team could aim to use a single data catalog to manage all data domains. Or, more likely, your team may need to align different data catalogs per-domain, or possibly support many different data catalogs coming from different vendors or cloud providers. For all data domains running in Oracle Cloud, or integrated with Oracle Cloud, the OCI Data Catalog is an ideal tool for business domain governance.

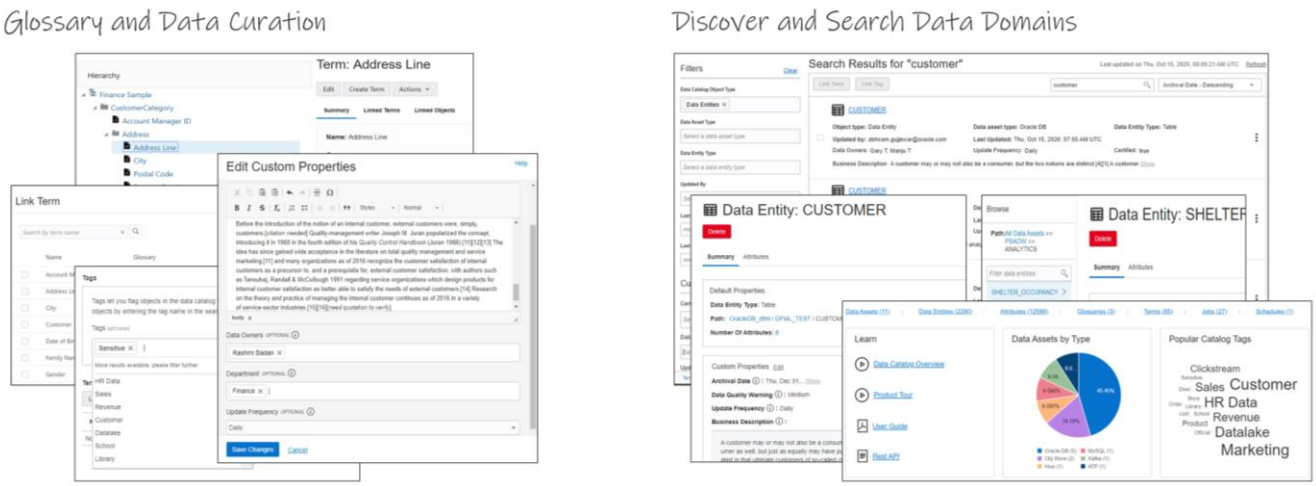


Figure 41: Using the OCI Data Catalog to align business data domains, taxonomy and data zones

The Oracle Cloud overall provides a rich set of services and capabilities that can help any organization on a journey to create a dynamic Data Fabric and a trusted Data Mesh. Native services share a common control plane and are pre-integrated in several operationally significant ways. The overall approach across all OCI data fabric services is to provide low-code or no-code capabilities, along with the many benefits of a serverless infrastructure where customers only pay for what they use (no idle resources).

Conclusion – Get Ready for Change

For far too long the world of data management has artificially segmented OLTP from Analytic workloads, the old ways of batch processing and brittle waterfall-style data management methodologies are due for systemic disruption. Like the application development software ecosystem before it, the data management software ecosystem must adapt and adopt newer more business-driven thinking and agile operations to thrive in the decades to come.

Now we have reached an inflection point with new technology enablers like service mesh, software defined networking, polyglot streaming for CTL, and commodity scale-out platforms. These foundation improvements can empower a new kind of dynamic Fabric for enterprise data. The most challenging part, as always, will be to change hearts, minds, and the culture of what is possible. We will need to invert old assumptions to apply data product thinking, unify OLTP and analytic data domains, reject batch-first thinking and embrace a shift to decentralized architecture patterns.

Throughout this document we have described a dynamic Data Fabric and a trusted Data Mesh. This intersection of capabilities is a new paradigm for enterprise IT data architecture. Several meaningful differences from the past are distinctively noted in this new conception:

- **Foundation: Data Product Thinking** – an organizationally disciplined way of managing data assets
- **Principle #1: Decentralized, Modular Mesh** – rejecting the data integration monoliths of the past
- **Principle #2: Enterprise Data Ledgers** – driving integrations from data logs rather than static data
- **Principle #3: Trusted, Polyglot Data** – support all payload types, preserve ACID consistency on the stream

The intersection of these foundation principles taken as a whole is what differentiates the approach with others that have come before. For example, the market domain for Data Fabric is very large (several billion dollars) and includes a variety of techniques and approaches to data integration and data management – most of which do not apply event-driven decentralized architectures for data. Likewise, the Data Mesh technologies as initially conceived are most closely aligned with microservices event-sourcing patterns and do not prioritize data pipelines or trusted, ACID-level data consistency.

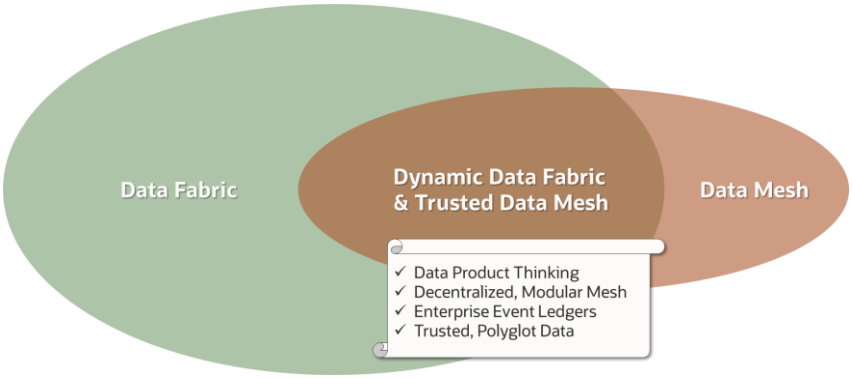


Figure 42: Dynamic Data Fabric & Trusted Data Mesh

This document is describing is the union of the two concepts – a Data Fabric (enterprise-grade, and trusted) that has the architecture attributes of a Mesh (decentralized, event-driven). Data product thinking is a characteristic that has been long associated with both the Mesh and the Fabric (albeit by other names, such as data monetization, data stewardship/governance, and dataops), but here data product thinking is elevated to a central tenet of the entire approach – placing business value of data at the very heart of the data management philosophy.

Oracle has long been a leader in data management (#1 database vendor[lx] and cloud database leader according to Gartner[lxi]) and data integration (12 year 'Leader' in Gartner Magic Quadrant[lxii]) and is now also taking the lead in these next generation dynamic Data Fabric and trusted Data Mesh categories as well. Customers who invest in Oracle data management and data integration technologies like the GoldenGate platform have a clear path forward to the future. In fact, the wider vision of a dynamic Data Fabric and trusted Data Mesh is precisely what is guiding the ongoing investments into the overall Oracle GoldenGate platform for streaming data.

The Oracle GoldenGate team is committed to providing tools and technology that help our customers create value from their data, and this aligns with the overall Oracle mission, "*to help people see data in new ways, discover insights, unlock endless possibilities*". Today, the Oracle GoldenGate platform helps customers to create innovative data products such as:

- **Curated Change Streams** – data product owners can publish changed data APIs for external consumers
- **Low Code Data Pipelines** – governed, visual editor for functionally rich data flows and transformations
- **Time Series and Spatial Analysis** – easy to use patterns for analyzing any data in motion
- **Real-time Dashboards and Alerting** – quickly create charts and visualizations that are continuously updated
- **Streaming Data Services** – push real-time changed data to databases, NoSQL or any data lake platforms

- **Production ML Scoring and Alerting** – take refined scoring models and put them into a real-time stream

The GoldenGate platform is also widely used to enable data-driven solutions and data products that are consumed through other technologies, such as:

- **Synchronous OLTP Databases** – for low-downtime migrations, high availability (HA) and recovery
- **Multi-Active Databases** – for geographic data sharding and cross-regional data replications
- **Classical Data Warehouse** – as an enabler for 'trickle feeds' or CTL (continuous transformation and loading)
- **Data Lake Ingest** – non-invasive and real-time data staging into Cloud or big data technologies
- **Microservices Outbox** – pattern for maintaining data consistency while replicating application events
- **Stream Processor** – as an embedded streaming solution for edge, gateways, or streaming applications

Oracle's laser-focus on high value data products and business outcomes from GoldenGate leads the industry as a true multi-cloud solution for modern decentralized data architectures that will define the future.

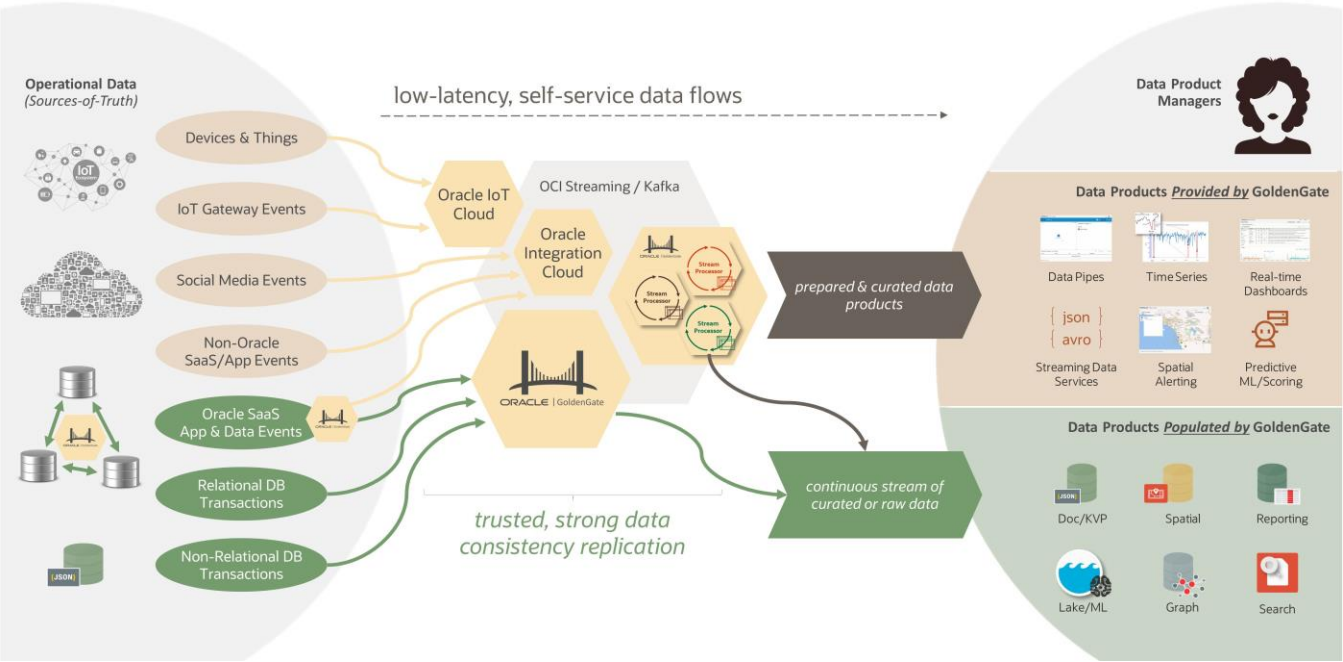


Figure 43: Modern, dynamic Data Mesh and trusted Data Fabric with GoldenGate in a mesh with Oracle IoT Cloud and Oracle Integration Cloud

GoldenGate heritage around trusted real-time data and its newest microservices/cloud architecture makes it the ideal foundation for next generation dynamic Data Fabric and trusted Data Mesh. If your data architecture is inspired by data product thinking and aims to be (i) a decentralized, modular mesh, while (ii) leveraging event ledgers for integration, on (iii) trusted, polyglot data streams – then you should take a close look at Oracle GoldenGate platform and the whole stack of services in the Oracle Data Fabric portfolio.

References

[i] Forrester Blogs, Information Fabric Delivers Next Generation of Data Virtualization, Noel Yuhanna, August 2013 https://go.forrester.com/blogs/13-08-15-information_fabric_30_delivers_the_next_generation_of_data_virtualization/

[ii] Gartner, Emerging Technologies: Data Fabric Is the Future of Data Management, Sharat Menon, Ehtisham Zaidi, Mark Beyer, 4 December 2020 https://www.gartner.com/en/documents/3994025

[iii] Forrester, The Forrester Wave™: Enterprise Data Fabric, Q2 2020, Noel Yuhanna, June 2020 https://www.forrester.com/report/The+Forrester+Wave+Enterprise+Data+Fabric+Q2+2020/-/E-RES157288

[iv] Forrester, The Forrester Wave™: Enterprise Data Fabric, Q2 2020, Noel Yuhanna, June 2020 https://www.forrester.com/report/The+Forrester+Wave+Enterprise+Data+Fabric+Q2+2020/-/E-RES157288

[v] https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2818.1994.tb03441.x

[vi] https://gcn.com/articles/2000/06/07/heres-all-you-need-to-know-on-tcpip.aspx

[vii] https://cacm.acm.org/magazines/2009/12/52840-a-smart-cyberinfrastructure-for-research/fulltext

[viii] https://communitywiki.org/wiki/DataMesh

[ix] Gartner, Maverick* Research: Revolutionizing Data Management and Integration With Data Mesh Networks, Mark Beyer, Guido De Simoni, Ehtisham Zaidi, 4 Nov 2016 https://www.gartner.com/en/documents/3500835/maverick-research-revolutionizing-data-management-and-in

[x] https://martinfowler.com/articles/data-monolith-to-mesh.html

[xi] https://en.wikipedia.org/wiki/Paradigm_shift

[xii] https://en.wikipedia.org/wiki/Cruft

[xiii] https://www.gartner.com/smarterwithgartner/use-a-hybrid-integration-approach-to-empower-digital-transformation/

[xiv] *Data derived from real world discussions with customers who have adopted the methodologies and tools described in this document*

[xv] https://en.wikipedia.org/wiki/Design_thinking

[xvi] https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

[xvii] https://queue.acm.org/detail.cfm?id=3321612

[xviii] Gartner, Market Share Analysis: Data Integration Tools, Worldwide, 2019, Sharat Menon, 13 July 2020 https://www.gartner.com/en/documents/3987502/market-share-analysis-data-integration-tools-worldwide-2

[xix] https://en.wikipedia.org/wiki/Design_thinking

[xx] https://hbr.org/2018/10/how-to-build-great-data-products

[xxi] https://uxplanet.org/product-thinking-101-1d71a0784f60

[xxii] https://hbr.org/2016/09/know-your-customers-jobs-to-be-done

[xxiii] https://hbswk.hbs.edu/item/clay-christensen-the-theory-of-jobs-to-be-done

[xxiv] https://medium.com/rapido-labs/the-data-pm-and-his-experiments-8f38bcf4ae9b

[xxv] https://www.oracle.com/a/tech/docs/maximum-availability-wp-19c.pdf

[xxvi] https://en.wikipedia.org/wiki/Functional_decomposition

[xxvii] https://en.wikipedia.org/wiki/Decomposition_(computer_science)

[xxviii] https://en.wikipedia.org/wiki/Object-oriented_programming

[xxix] https://chrisrichardson.net/post/refactoring/2019/10/09/refactoring-to-microservices.html

[xxx] https://en.wikipedia.org/wiki/Third_normal_form

[xxxi] https://www.oreilly.com/library/view/domain-driven-design-tackling/0321125215/

[xxxii] https://www.informit.com/articles/article.aspx?p=2023702

[xxxiii] https://en.wikipedia.org/wiki/Design_Patterns

[xxxiv] https://medium.com/@mark.tesla2/elon-musk-and-first-principles-thinking-e0035730d7

[xxxv] https://www.foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution

[xxxvi] https://www.businessinsider.com/what-is-davos-world-economic-forum-conference-2020-1

[xxxvii] https://en.wikipedia.org/wiki/The_Innovator%27s_Dilemma

[xxxviii] https://en.wikipedia.org/wiki/DevOps

[xxxix] https://www.devopsdigest.com/the-state-of-database-deployments-in-application-delivery-2019

[xl] https://en.wikipedia.org/wiki/Object%E2%80%93relational_impedance_mismatch

[xli] https://www.ieee.org/about/news/2020/survey-chief-information-officers-and-chief-technology-officers.html

[xlii] https://dataprivacymanager.net/5-biggest-gdpr-fines-so-far-2020/

[xliii] https://www.iso.org/standard/72437.html

[xliv] https://en.wikipedia.org/wiki/Defense_in_depth_(computing)

[xlv] https://en.wikipedia.org/wiki/OSI_model#Layer_7:_Application_Layer

[xlvi] https://en.wikipedia.org/wiki/Mutual_authentication

[xlvii] https://oauth.net/2/

[xlviii] https://openid.net/connect/

[xlix] https://en.wikipedia.org/wiki/XACML

[l] https://www.openpolicyagent.org/

[li] https://www.datanami.com/2021/01/21/governance-privacy-and-ethics-at-the-forefront-of-data-in-2021/

[lii] https://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/

[liii] https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/data-sovereignty-per-microservice

[liv] https://microservices.io/patterns/data/transactional-outbox.html

[lv] https://microservices.io/patterns/data/cqrs.html

[lvi] https://microservices.io/patterns/data/event-sourcing.html

[lvii] https://dl.acm.org/doi/10.14778/1454159.1454179

[lviii] https://www.forrester.com/report/The+Forrester+Wave+Enterprise+Data+Fabric+Q2+2020/-/E-RES157288

[lix] https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingcompartments.htm

[lx] https://db-engines.com/en/ranking_trend

[lxi] https://www.oracle.com/news/announcement/oracle-recognized-in-gartner-cloud-database-market-reports-121420.html

[lxii] https://blogs.oracle.com/dataintegration/gartner-mqdataintegration2020

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle