INTELLIGENT BUSINESS STRATEGIES

WHITE PAPER

Simplifying Data Mesh for Self-Service Analytics on an Open Data Lakehouse

By Mike Ferguson
Intelligent Business Strategies
June 2023

Research sponsored by:

dremio

Table of Contents

INTRODUCTION

There has been a frenzy of activity in data and analytics over the last decade

In the last decade we have seen a frenzy of activity when it comes to data and analytics. Many new technologies have emerged over that time including new data science and data engineering tools, new database products, and data management platforms. Also, the arrival of self-service data preparation, augmented business intelligence tools, and machine-learning (ML) automation has helped lower the skills bar needed to use these technologies. In addition, the number of new data sources that companies want to access has exploded as people look to add to what they already know to produce richer insights for better decision making.

Also, the number of new data sources has increased rapidly

Many companies have overspent because of stand-alone data and analytical initiatives

With so much going on, it's not surprising, that different departments in many companies have embraced all these technologies in their determination to deliver new value. However, different stand-alone data and analytical initiatives across large and medium-sized enterprises have resulted in piecemeal adoption of technologies resulting in a fractured approach to data and analytics development. It is common to see multiple different overlapping technologies in different parts of the enterprise. Also, re-invention, and data redundancy have occurred. As a result, many organizations now have multiple stand-alone analytical systems such as multiple data warehouses, multiple data lakes in use in data science, graph databases, and streaming analytics initiatives. There are also multiple overlapping and competing toolsets. All of this has led to platform complexity and inadvertent overspend.

Also, considerable data redundancy and multiple overlapping technologies exists across many analytical systems

The problem with this is that stand-alone departmental and line-of-business data and analytical development initiatives has meant that different teams are making use of different data engineering tools to clean and integrate data for different analytical use cases. In some cases, they are even sourcing and engineering the same data from the same data stores for these different analytical use cases. Also, stand-alone self-service analytics initiatives have led to many different self-service data preparation jobs, BI reports, dashboards, and machine-learning models being produced using different data preparation, BI and data science tools across the enterprise. All of this has led to managing tools and data platforms that do not integrate with each other and overspending. Also, very little of what has been created is published anywhere and so people who could benefit are often unaware of valuable data and insights that have already been created across the enterprise.

Organizations have made progress, but progress is slower than desired, integration is poor, and costs are running higher than they should be

In summary, while many organizations have been busy and progress has been made, what is being created under the banner of data and analytics is not joined up. The pace of development is slower than desired and the way in which artifacts like ML models, BI reports and dashboards are produced is somewhat untidy, inefficient, and complex to understand. This leads to cost that is higher than it should be.

As the pace of business quickens executives are demanding a strategy aligned industrialized development approach to shorten time to value

However, we are now in an era where the pace of business is quickening, and executives are demanding rapid development to compete in a data-driven digital economy. They want to move away from fractured development initiatives to *an industrialized development approach* that accelerates data engineering and enables the sharing of data, ML models, and business intelligence that are all being created to align with business strategy. To do that means more people are

needed but also, we need to organize them to build data and analytical products that can be shared, reused, and assembled more rapidly in a similar way to a manufacturing production line. A good example, of this is the emergence of a Data Mesh[1] and reusable data products.

Data Mesh is a divide and conquer approach to data engineering aimed at accelerating development

The general idea behind data mesh is to move to a decentralized, 'divide and conquer' approach to data engineering to speed up development. That means upskilling more people in different business domains around the enterprise to enable them to produce high-quality, reusable, compliant datasets known as data products. Data products are defined[2] as being discoverable, addressable, trustworthy, self-describing, interoperable and secure. They include the data itself, the pipeline (rules to clean and integrate data), the runtime specification to execute the pipeline and APIs. They can be physically stored or represented as virtual views of data in multiple sources that integrate data on-demand to produce the required data product.

A data concept model can be used to identify data products to produce

An easy way to think about data products is to think about data concepts in a data concept model. For example, data concepts in an insurance company would include customers, insurance brokers, claims assessors, insurance policy applications, quotes, policy agreements, premium payments, claims, etc.

Multiple teams of decentralized data producers create pipelines to produce reusable data products

The objective of a data mesh is to enable different teams of data producers to clean, transform and integrate data to create a data product for each of these. So, in this insurance example, you would have a data product holding all claims data, another data product holding all premium payments, and another holding all customers, etc. These data products are 'building blocks' that are made available for sharing and reuse in multiple analytical systems.

Data products can be consumed and used in multiple analytical workloads

This is a very different approach to building pipelines to populate a monolithic data model in a single analytical system like a data warehouse. In a data mesh approach, work is divided up among different teams of business domain-oriented data engineers, who are experts in specific kinds of data (e.g., claims data), and who use self-service common data management software to build pipelines that transform and integrate data to produce specific data products. These data products can then be assembled and used in multiple analytical workloads to drive business outcomes.

There are several requirements that need to be met to make this possible

In this fast-moving digital economy, the question is, how do you implement this? How can you move away from the fractured approaches of the last decade and industrialize the development of data and analytics to rapidly build a data and AI-driven enterprise? What are the requirements that need to be met to make this possible? How does data mesh and data products fit in and what else is needed?

This paper seeks to answer these questions by defining the requirements needed to make it possible. It then looks at how one vendor, Dremio, can help organizations reduce the cost of operating, accelerate data engineering, and shorten time to value.

---

[1] Data Mesh – Delivering Data Value at Scale, Zhamak Dehghani, O'Reilly ISBN: 9781492092391

[2] How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh, Zhamak Dehghani, May 2019

REQUIREMENTS TO ACCELERATE THE DEVELOPMENT OF DATA PRODUCTS

Before getting into requirements, let's first look at what Data Mesh is.

WHAT IS DATA MESH?

Data Mesh is a decentralized approach to self-service data engineering to produce reusable data products available to share across the enterprise

Data Mesh is a decentralized data engineering approach created by Zhamak Deghani, in 2019. There are four principles of Data Mesh:

1. Domain oriented decentralized data ownership and architecture
2. Data as a product
3. Self-serve data infrastructure as a platform (common platform)
4. Federated computational data governance

In this approach, decentralized teams of domain-oriented subject matter experts most familiar with data used in their business domain, build pipelines to produce reusable, data products (datasets) that they own and make available for others to use across the enterprise. To simplify things, self-service development is done on common data infrastructure software. Data governance is federated since enterprise-wide data privacy policies must be applied to sensitive data in all data products while data owners control access to data products they create.

Some data governance policies must apply enterprise-wide, and some are controlled locally in the domains

The following requirements need to be met if companies are to implement a data mesh and accelerate data product development in a data-driven enterprise.

ORGANIZATIONAL REQUIREMENTS TO ACCELERATE DEVELOPMENT

Getting the right organizational setup in place is critical to success

The first major requirement to move away from siloed data engineering and analytics development initiatives is to address the organization problem. While it is accepted that different parts of the business need to deliver value in their respective areas, it has to be part of a bigger plan. We have to consider this as a jigsaw puzzle where different teams are building different pieces that align and come together to achieve common business goals. Therefore, development needs to be coordinated as opposed to everyone working on isolated stand-alone projects. That requires a federated organization setup with a central program office that acts as a 'base camp' for all data and analytical projects. This is shown in Figure 1 and is typically by the office of the Chief Data Officer.

A federated setup is needed with a CDO run central program office coordinating development activity across multiple decentralized teams and aligning it to business strategy

Development of data products should all be aligned with strategic business priorities to achieve high-value business outcomes

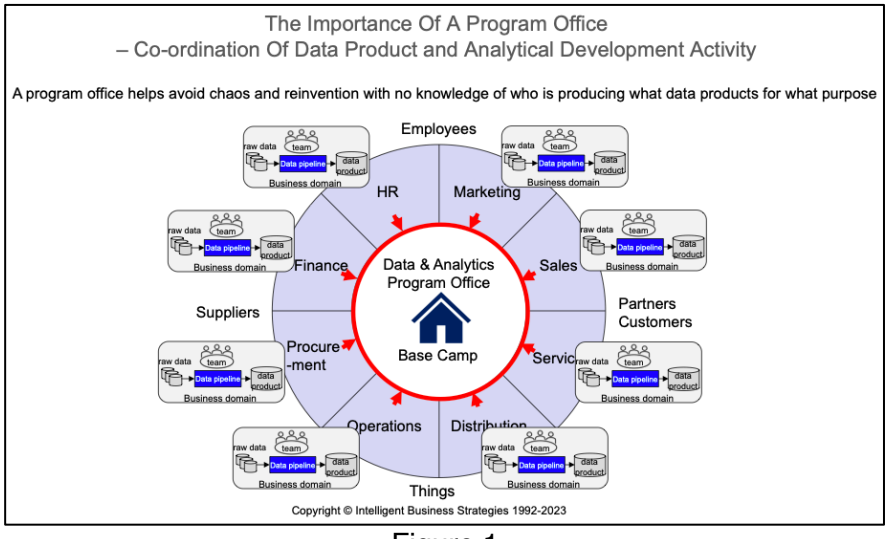


Figure 1

A center of enablement embeds data engineering experts in decentralized domain-oriented teams

A centralized program office supporting decentralized data engineering teams offers several benefits.

- It provides a focal point for the company on data and analytics. It ensures that each data and analytical project is aligned to one or more business objectives in the company business strategy.
- Prioritization of data product, BI and ML model development can be dictated by and aligned with strategic business priorities to help achieve specific high-priority business outcomes.
- It enables collaborative planning and coordination of all data product, BI (reports, dashboards), and ML model development projects across multiple domains while avoiding reinvention and siloed engineering.
- It provides teams with an understanding of the 'big picture' on who is building what right now, what is still outstanding, how everything fits together and is progressing towards helping achieve specific business outcomes.

Their job is to upskill citizen data engineers while ensuring adoption of best practices

A centralized program office can also function as a center of enablement from where professional IT data engineering experts can be embedded in business domain-oriented teams of data producers to help upskill citizen data engineers and guide them to adopt best practices. This federated organizational setup is critical to ensuring progressive incremental development of a data-driven enterprise.

STANDARDIZATION REQUIREMENTS TO ACCELERATE DEVELOPMENT

Standardization is needed to industrialize data product development and avoid unnecessary complexity

The second major set of requirements in accelerating data product development is to lay the groundwork for rapid and consistent development of data products and to avoid unnecessary complexity across teams. Standardization (see Figure 2) should improve development productivity and enable data and metadata sharing across teams of domain-oriented data producers without getting in their way. It should also make everything easier to maintain over the longer term.

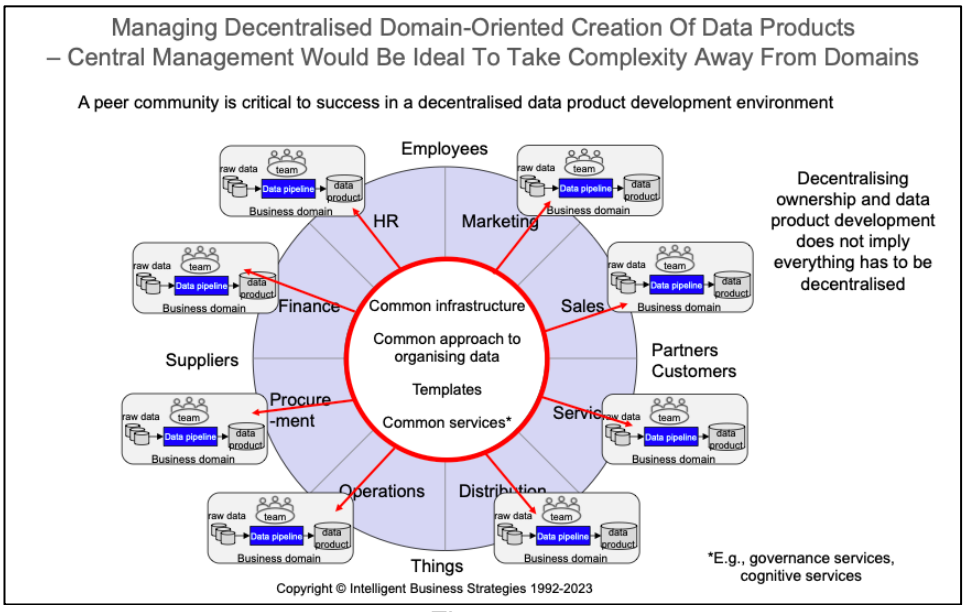


Figure 2

All domain-oriented teams of data producers should use a common data platform technology

Build data products once and share them everywhere

Data products representing master data should be used enterprise-wide

In terms of standardization requirements, it should be possible to:

- Have all domain-oriented teams of data product producers make use of a common data platform to build and manage data products
- Engineer data to build both virtual and physical data products

You can connect directly to multiple data sources and integrate data to create virtual or physical data products

Alternatively, source data can be ingested into a managed lakehouse first before building data products

A common approach is needed to produce data products

Templates and common services help speed up development

Data products should be shared and governed in a standard way

- Produce common master data products (e.g., customers, products, employees, suppliers, materials, assets etc.,) once and share them everywhere to avoid them being reinvented unnecessarily
- Leave source data where it is and define rules and mappings to transform and integrate that data to create virtual data products (virtual views of integrated data) and physical data products (persisted datasets)
- Provide the option to ingest source data into a managed lakehouse (preferably one that offers open table formats, ACID[3] support, schema evolution, columnar file formats, data partitioning, indexing, fine grained access control and support for deploying data as code) to ensure consistent data and then enable different teams of domain-oriented data producers to engineer that data for self-service analytics
- Use a common approach to managing and governing source data ingestion, engineering data and publishing of all data products so that data is processed and managed in a consistent way no matter which teams are developing data products
- Enabling the sharing of standard templates and common services across multiple teams to simplify and expedite development while also ensuring adoption of best practices
- Adopting a common federated approach to governing access to published data products made available for sharing
- Enabling the sharing of data products in a standard way

The use of a common data platform enables each team of data producers to create their own workspace where they can find, connect to raw data, and then transform and integrate it in isolation to create data products. In addition, they would also be able to share business-ready data and metadata across teams in a collaborative environment. Figure 3 shows how the above requirements can help standardize the setup to establish a common approach across multiple domain-oriented teams of data producers.

A common data platform enables teams to build business-ready data products in isolation and easily share data and metadata with others

A standard setup is needed for all teams

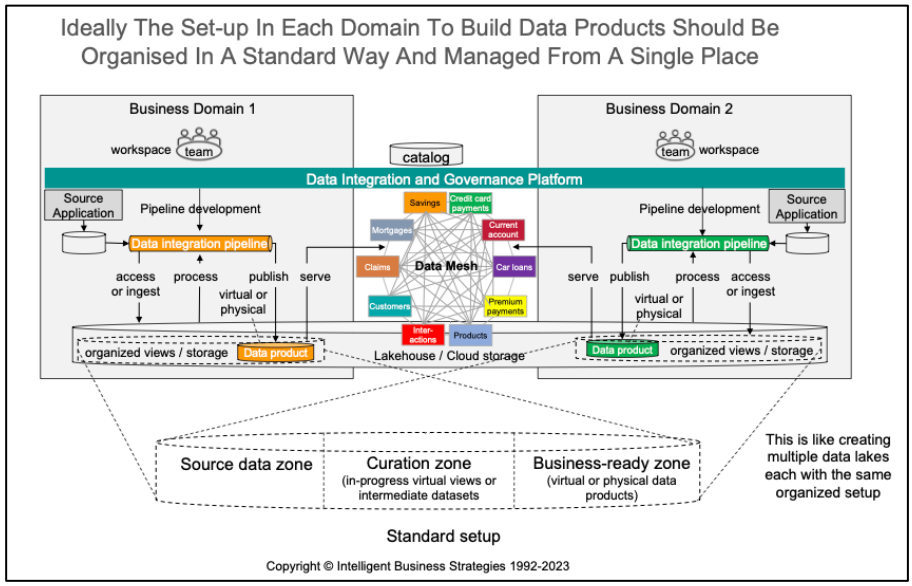


Figure 3

[3] Atomicity, Consistency, Isolation and Durability

In this way data products can be produced by connecting directly to data sources and transforming and integrating data.

When using a data lake, data ingestion needs to be managed and source data checked before it is made available for use

However, if an organization wishes to ingest data into a central data lake (e.g., cloud object storage), it should also be possible to manage data ingestion in a common way so that new data is not made available to teams of data producers until it is quality checked. Managed data ingestion would also enable data ingestion to be tracked and prevent the same data from being ingested more than once. This is particularly important when it comes to external data that has been purchased. Once ingested and made available, all teams could then access a common ingestion zone and have their own workspaces (working environment) to create new data products or new versions of data products without impacting on any other data producers or data consumers.

Data producers need to have their own working environment to build data products

DATA PRODUCT DEVELOPMENT REQUIREMENTS

Incremental development of semantically linked master, transaction and aggregate data products is needed

Each team of domain-oriented data producers should be able to connect to, query, transform and integrate data from one or more sources to create a data product that can be easily shared across the enterprise. Examples of master data products might be customers, products, and suppliers. Examples of transaction data products would be orders, shipments, payments, returns etc.

In this way, multiple decentralized teams of data engineers can use a common data platform to incrementally create a set of semantically linked, reusable data products that can be published and made available to use in multiple analytical workloads. To create a data product, it should be possible for each team to use the common data platform to:

All data products should use common business data names and enterprise-wide identifiers

- Define common business data names that describe all attributes in the data product to be created. It is preferable that this should be documented as business metadata in a business glossary
- Ensure that the business data names defined for the data product include an enterprise-wide identifier
- Define and create a data product schema using the common business data names as column names in virtual or physical table structures

Automatic discovery and cataloging of source data

- Connect to and automatically discover data in domain data sources to identify the data needed to build a data product
- Map the physical data names of the required data in the data sources to the business data names used in the data product to be created

Map required source data to data product business data names

- Access identified source data where it resides or ingest the source data into a central or domain-oriented data lake using a managed service. If ingestion occurs, then the ingested data will be the source data used to produce the desired data product

Engineer data ensuring any sensitive data is protected to create a data product

- Engineer the identified source data in domain-oriented workspaces, to produce the required virtual or physical data product
- Ensure any personally identifiable data contained in the data product is masked during data integration so the data product created is compliant with enterprise-wide data privacy legislation. This ensures data governance is designed into the data engineering process.

Assign a data product owner and publish it in a data marketplace to make it available for sharing

- Appoint and tag the data product with an owner
- Grant the owner privileges to enable them to approve/reject access requests to the data product

- Publish domain-oriented data products in a shared workspace or marketplace for others to request access to and use

DATA PRODUCT VERSION CONTROL REQUIREMENTS

With so many new data sources, the chances of new data emerging that would enrich what you already have are very high. Therefore, the demand for change is inevitable which means data product version control is needed. Also, firms have long struggled with the impact of change in traditional data warehouse architectures and the domino effect it causes resulting in long change times.

To address this, it should also be possible to support:

Version control is also needed

- Data product lifecycle management. This would enable new data sources to be added, schema changes tracked, and new enriched versions of data products created that can offer more value.
- Build data integration pipelines to produce new versions of data products in isolation without the need to copy data and without impacting on data consumers in production while new versions are being developed

SELF-SERVICE ANALYTICS SEMANTIC LAYER REQUIREMENTS

It should be possible to use the common data platform to:

Access to data products needs to be governed

- Set policies to govern data sharing using role and attribute-based access control
- Enable consumers to search for, find and request access to data relevant products subject to approval from data owners and acceptance of data sharing terms and conditions

They also need to be easy to find and should be provisioned without the need to copy data

- Provision data products without the need to copy data as copies introduce security risks, are hard to govern, and data consumers question the validity of the data
- Enable consumers to select and combine relevant semantically linked data products without the need to copy data to build:
  - Star schemas and a semantic layer for use in self-service analytics
  - Curated data for data scientists to develop machine-learning models

Consumers should be able to select the data products they need and for different analytical use cases

For example, to analyze sales in a retailer by store, product, and customer over time it should be possible to access:

- The channels data product (e.g., stores, website)
- The customers data product
- The products data product
- The sales data product
- The time data product

and assemble them into a star schema as shown in Figure 4. Note that because the data products have enterprise-wide identifiers, they are semantically linked and so should snap together easily to enable self-service analytics. Using common data names also ensures a *common semantic layer* is created and so all tools see the same data names.

By using common business data names and global identifiers, data products become the building blocks to incrementally build a universal semantic layer for self-service analytics

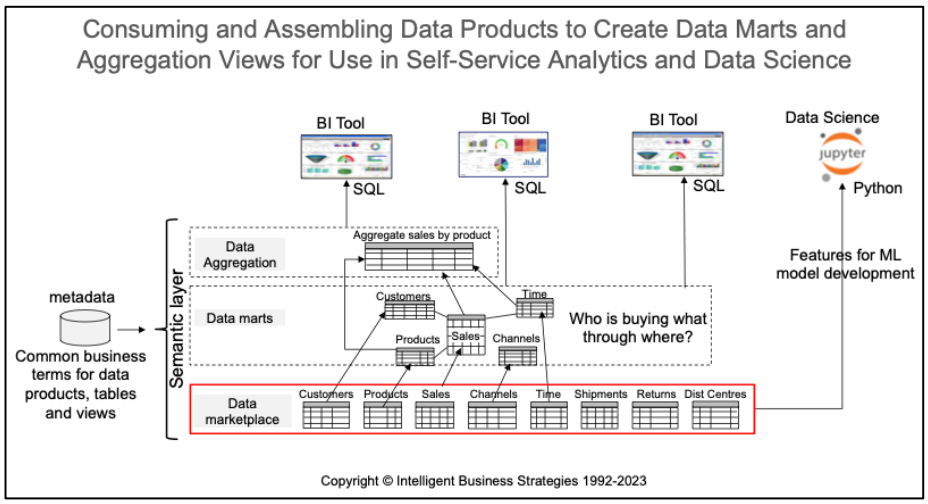


Figure 4

All BI and data science tools can access commonly understood data via a common semantic layer

- Enable consumers to define different flattened views of data in star schemas to simplify access in the semantic layer
- Enable access to data via the common semantic layer from BI and data science tools for use in self-service analytics and ML model development

BUILDING A DATA MESH FOR SELF-SERVICE ANALYTICS ON DREMIO'S OPEN DATA LAKEHOUSE

Dremio has two main products

Having understood the requirements to implement a data mesh, this section of the paper looks at how one vendor is stepping up to meet these requirements. That vendor is Dremio.

Dremio offers two main products. These are:

- Dremio Sonar
- Dremio Arctic

DREMIO SONAR

Dremio Sonar is a data lakehouse query engine that supports self-service analytics with data warehouse functionality and data lake flexibility. Before looking at Dremio Sonar, let's first define what a data lakehouse is.

What is a Data Lakehouse?

A data lakehouse is a single platform that provides the scale and flexibility of a data lake together with the data integrity and query capability and fine-grained governance of a data warehouse

For many years data warehouses have been used to organize and store data in relational tables in a proprietary format tied to a specific data warehouse database management system (DBMS) for query using SQL. This means data is only accessible by that specific DBMS engine. Data lakes on the other hand have stored data in files that could be accessed by multiple engines. Historically, this data was primarily analyzed by data scientists using programs written in popular languages such as Python and R. Then the data lakehouse emerged to provide the scale and flexibility of a data lake with columnar file formats, ACID, open table formats (e.g., Iceberg) with fine grained governance, Spark and SQL data transformation capability and SQL query performance that is becoming competitive with data warehouses. With a data lakehouse, multiple engines can access and manipulate data directly from data lake storage without compromising data integrity and without the need to copy data into a proprietary format specific to a single data warehouse DBMS.

Creating a Universal Semantic Layer with Dremio Sonar

Dremio Sonar is a lakehouse query engine for self-service analytics

Dremio Sonar is an Apache Arrow-based columnar lakehouse query engine that supports BI reporting, dashboards and interactive analytics over data in data lake storage as well as unified access to data in cloud object storage and other data sources. Using Dremio Sonar, self-service interactive BI tools can access data via unified views that use common business data names to provide a common semantic layer across all self-service analytics tools. It enables organizations to run high performance queries on data lakes, simplify access to data in multiple underlying data stores, modernize their data architecture and facilitate self-service creation of data products anywhere, be it on-premises, hybrid, or cloud.

Teams can create a logical view of the data for data consumers, with the physical data residing in cloud data lakes (S3, ADLS), on-prem data lakes (HDFS), metastores (Nessie, Hive, AWS Glue), relational databases (Snowflake, Redshift, etc.), NoSQL databases, and a Dremio-to-Dremio connector.

It supports virtual views that use SQL to transform and integrate data

Dremio Sonar's key features include:

- **Virtual views**: Data consumers can transform and integrate data via SQL-based virtual views. Users can take advantage of an integrated

SQL editor and low/no-code interface that auto-generates federated SQL queries to filter, extract, transform and join data in multiple underlying data sources

Views can be organized into spaces to make it easy to find and provide secure access to data

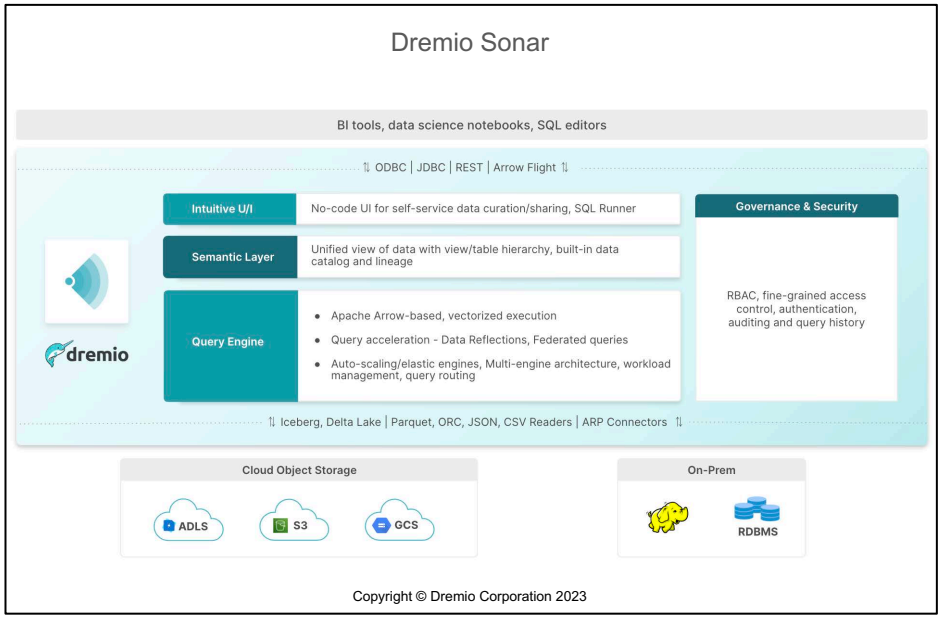
- **Spaces:** Data teams can create a single, consistent view of data for data consumers. They can organize views of data products in a convenient hierarchy of folders to make it easy to browse and find data. Privileges can be set on a folder anywhere in the hierarchy to ensure data products are shared securely within the organization.
- **Reflections:** Dremio transparently accelerates queries using reflections so users can work with data logically without needing to worry about physical tables/copies to make queries run quickly. Even as additional views are added, Dremio's optimizer can intelligently utilize existing reflections to accelerate queries on those new views.

Metadata is available to show the meaning of data and how data products were constructed

- **Integrated catalog:** To enable users to discover data products, understand their meaning and access built-in lineage to visualize how they were constructed. Users can also add descriptions and tags to datasets for others to leverage. The catalog information is exposed through a UI that simplifies data discovery and SQL editing.

Data can be described and tagged to make it easier to collaborate and share securely

- **Collaboration:** Users can collaborate by securely sharing data products with other users/teams with the click of a button (or a SQL command). In addition, users can collaboratively author metadata such as descriptions and tags.



Source: Dremio

Figure 5

Dremio Sonar includes support for:

It supports a range of cloud and on-premises data sources

- Connectors to access data in the following data sources:
  - Cloud data lakes
    - AWS S3, Azure Blob Storage, Azure Data Lake Storage (ADLS), and Google Cloud Storage
  - On-premises data lakes
    - Hadoop distributed file system (HDFS) and S3 compatible object storage (e.g., MinIO, Dell)
  - Relational databases

        - Examples include Amazon Redshift, IBM Db2, Microsoft Azure Synapse Analytics, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Snowflake, Teradata
    - NoSQL databases
        - MongoDB
    - Open table formats
        - Apache Iceberg tables, Delta tables
    - Metastores
        - Nessie
        - Hive tables accessing datasets on HDFS, S3 and ADLS
        - AWS Glue Catalog
    - Elasticsearch
    - NAS
    - Other Dremio Sonar clusters (nested architecture)

There is also a Dremio-to-Dremio connector to query data managed by multiple Sonar instances

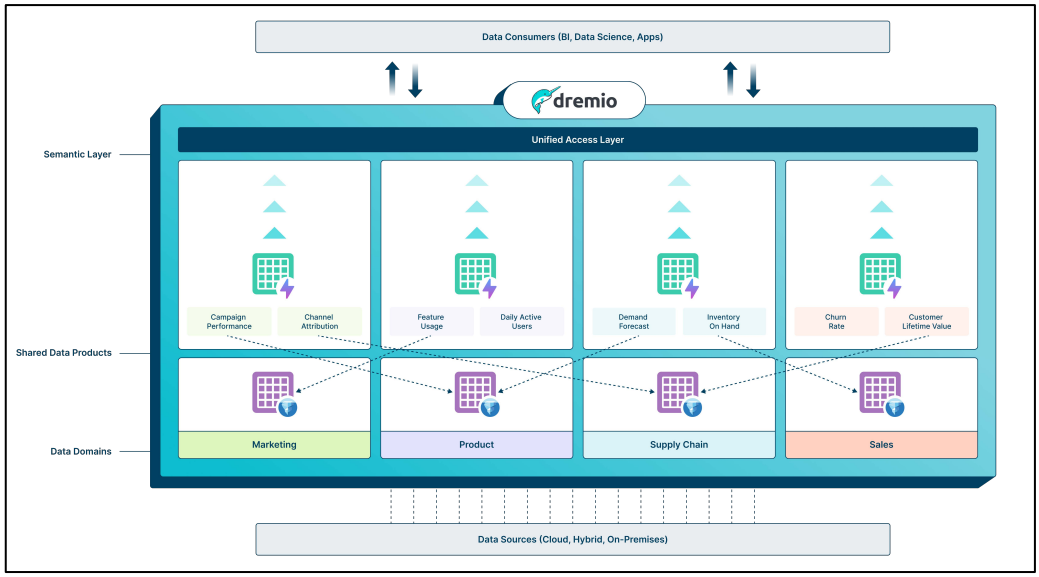
- Dremio Sonar virtual tables and views
- A cost-based query optimizer
- Pushdown optimization to make back-end databases and lakehouses do the work to retrieve and filter the required data for queries
- Query acceleration to boost query performance using:
    - Dremio Reflections - these are optimized materializations of source data or queries that are held in a reflections data store. Both raw and aggregated reflections can be created to improve query performance
    - Columnar Cloud Cache (C3) based on Apache Arrow

Columnar cloud cache, Dremio reflections and pushdown optimization all help improve query performance and are transparent to users

Both mechanisms are used by the Dremio Sonar optimizer to accelerate popular BI tool self-service queries and are transparent to users.

Figure 6 shows how domain-oriented data producers can use Dremio Sonar to integrate data from multiple data sources to create virtual data products in a data mesh.

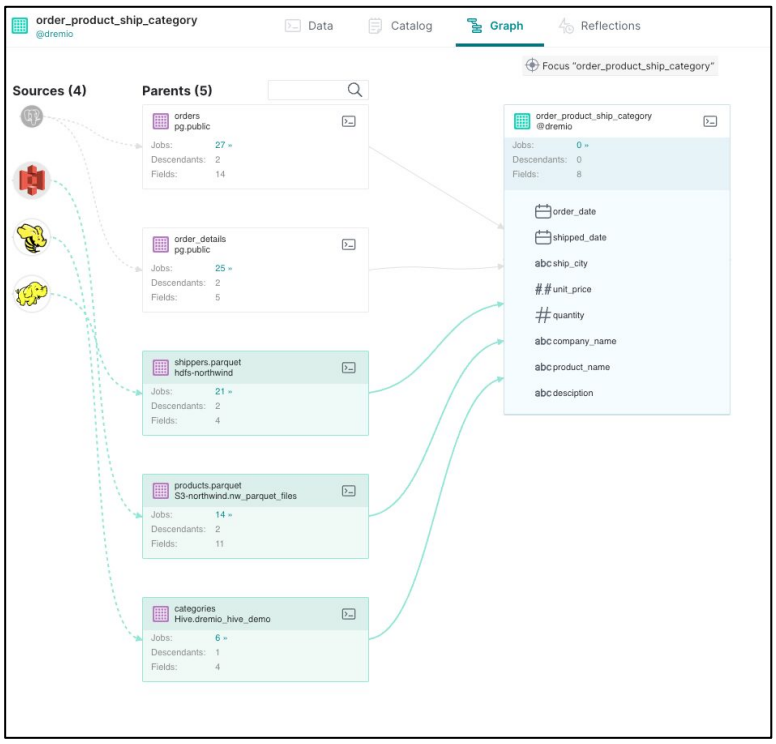
Teams of data producers can use Dremio Sonar to integrate data from multiple data sources to create virtual data products in a data mesh



Source: Dremio

Figure 6

Note that Dremio maintains full lineage showing transformation and integration mappings between sources and virtual views. With respect to data products in a data mesh, data consumers can see the lineage on how virtual views representing data products are constructed. Figure 7 shows a lineage example in Dremio Sonar.



Source: Dremio Figure 7

DREMIO ARCTIC

Before proceeding with Dremio Arctic, let's first explain what Apache Iceberg is.

What is Apache Iceberg?

Data lakes were originally created for use in data science

Enterprises are increasingly relying on data lakes for data science and over the last several years companies have amassed considerable amounts of data in data lakes typically held in columnar-oriented Parquet files. First generation data lakes were often on-premises storing data in files in Hadoop HDFS. Today, those files are more likely to be stored on cloud object storage. However, while cloud object storage data lakes gave companies scalability, they have often been lacking in governance which is something almost always found in auditable systems like data warehouses. Lack of governance often meant that it wasn't long before data in these data lakes started to become inconsistent. Data scientists would also often copy data for their own use resulting in significant data redundancy. There was little understanding of what was in these data lakes and so they began to deteriorate and were dubbed 'data swamps'.

Apache Iceberg is an open table format for files in object storage

Apache Iceberg is an open table format first developed by Netflix. It is an open-source project with contributors from companies such as Apple, Netflix, AWS, Snowflake, and Dremio. Apache Iceberg offers important capabilities that enable multiple applications to collaboratively work on data while ensuring transactional consistency. Additionally, Iceberg provides insights into dataset evolution and changes over time.

It overlays tables and metadata on top of columnar compressed data files in cloud object storage

The Iceberg table format is similar to tables found in traditional relational databases. The difference is that it is open which means that *multiple* engines like Dremio and Spark can query, process, and maintain data in the same underlying dataset via Iceberg tables without compromising data integrity. Iceberg offers a wide range of features, including:

- **Transactional Consistency:** Iceberg ensures transactional consistency across multiple applications. Files can be added, removed, or modified

atomically, providing full read isolation and allowing for multiple concurrent writes.

Iceberg ensures the data is transactionally consistent even when data is maintained using multiple query engines and supports automatic

- **Schema Evolution:** The enables Iceberg to track changes to a table over time to adapt to and accommodate evolving data requirements
- **Time Travel:** With Iceberg, you can query historical data and verify changes between updates using the time travel feature. This provides insights into the data's evolution and facilitates auditing and analysis
- **Data Partitioning and Evolution:** Iceberg facilitates updates to partition schemes as queries and data volumes change. This ensures flexibility and efficiency in managing large datasets
- **Rollback:** Iceberg allows you to quickly correct issues by returning tables to a known good state by rolling back to prior versions. This offers guaranteed data consistency, data integrity and simplifies data correction

It supports schema evolution, time travel, rollback, data partitioning and evolution as well as parallel query processing and filtering

- **Parallel Query Processing and Filtering:** Iceberg open tables can be overlaid on compressed, columnar file formats such as Parquet and ORC. It also supports data skipping, statistics and automated hidden partitioning. All of this enables massively parallel query execution, performance and filtering on very large data volumes

These features make Iceberg a robust and flexible table format for managing data in data lakes, providing improved data consistency, schema management, historical analysis, partitioning flexibility, data rollback, and enhanced query performance.

Metadata can be tracked though point in time snapshots and table changes tracked

Iceberg achieves these capabilities for a table by utilizing metadata files, known as manifests, which are tracked through point-in-time snapshots. As the table is updated over time, Iceberg maintains all the deltas, allowing for a comprehensive view of the table's schema, partitioning, and file information within each snapshot. This approach ensures full isolation and consistency and enables scalable parallel query performance on large volumes of data stored in data lakes.

Iceberg's open table format enables multiple different engines to query, process and update data in underlying data files while ensuring transaction validity and data integrity

Iceberg also employs a hierarchical structure to organize snapshot metadata efficiently which enables rapid changes to tables without the need to redefine all dataset files. In addition, it maintains the complete history within the table format itself, independent of any specific storage system. This design choice enables an open table format that provides the flexibility to change and accommodate multiple query engines on top of your data in object store without disrupting users or existing workloads.

The immutability of the historical state and the tracking of history in Iceberg enable users to query prior states at any snapshot or historical point in time. This allows for consistent results, facilitates comparisons, and offers the ability to roll back to previous versions to address any issues or corrections needed.

In summary, Iceberg's utilization of metadata files and point-in-time snapshots, along with its hierarchical organization and open architecture, ensures efficient management of table changes, parallel query performance at scale from multiple query engines, and the ability to query and roll back to historical states.

What is Dremio Arctic?

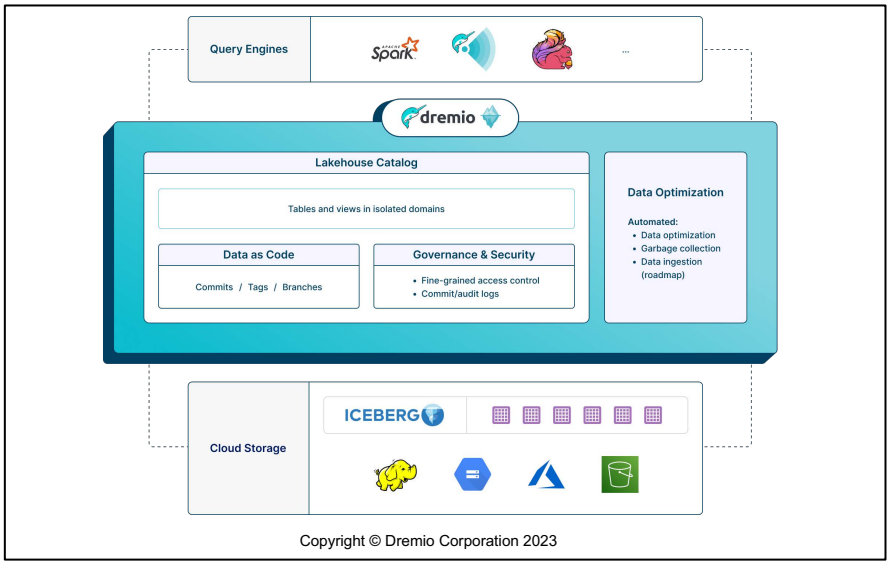
Dremio Arctic is a lakehouse management service for managing Apache Iceberg tables that overlay files in cloud storage data lakes

Dremio Arctic is a data lakehouse management service for Apache Iceberg tables that overlay files held in cloud object storage (e.g., AWS S3). It is a metadata service powered by Project Nessie, a cloud-native metastore that provides automatic optimization for Iceberg tables and 'data as code' capabilities that enable data teams to build, manage, and deliver data products the same way software developers manage code. It supports a range of execution engines including Dremio Sonar, Hive, Spark, and Flink so data consumers can utilize the engine best suited to each analytic workload. This is important because data can be streamed by one engine and updated or queried by another.

It supports multiple engines which means data can be updated by one engine and queried by another

Dremio Arctic contains a list of all Iceberg tables and views that are in the data lake together with their locations. It also contains pointers to metadata files that include metadata on Iceberg tables, schema, and partitions. The data, metadata and indexes are stored in AWS S3 cloud object storage. Dremio Arctic runs independently from Dremio Sonar but Dremio Sonar can connect to Dremio Arctic to gain access to data in Iceberg tables and views as a data source. Figure 8 shows the Dremio Arctic architecture.

Dremio Sonar can connect to Dremio Arctic to access Iceberg tables and views as a data source



Source: Dremio

Figure 8

Dremio Arctic supports 'data as code' so that data producers can engineer data without impacting on data consumers

Dremio Arctic uses branches, commits and tags in a similar fashion to Git repositories. This allows data engineers to create a branch where they can transform and integrate data in Iceberg tables. Updates (including changing table schema, changing view definitions, and ingesting data) can be made to tables via SQL or Spark in isolation. Any changes to the table can be thoroughly tested before they are merged into the main branch.

HOW CAN DREMIO BE USED TO BUILD A DATA MESH?

Dremio uses a phased approach to data mesh implementation

Now that we understand Dremio Sonar and Dremio Arctic the next question is how can they be used to build data products in a data mesh? This can be done in a three-phased approach.

Phase 1 – Unify Data Access

Start by leaving source data where it is and building virtual data products using Dremio Sonar

The first phase is unifying data access by using Dremio Sonar connect to and federate queries across multiple data sources. This enables each team of domain-oriented data producers to build data products using Dremio Sonar only. They can do this by creating virtual views with common business data names as column names for each data product they want to create. These views use federated SQL queries to access, transform and integrate data in one or more data sources at runtime to build virtual data products. Dremio Sonar uses reflections to provide faster access to this data without impacting source production systems. Virtual data products can be tagged by data producers and published in a shared space within Dremio Sonar, which acts as a *data marketplace* for consumers to find ready-made data products that are available for consumption and business use. Figure 9 shows both master data products (Customers, Products, Channels, Distribution Centers) and transaction data products (Sales, Shipments, Returns). The use of common business data names ensures that all data is commonly defined in a universal semantic layer.

Teams of data producers each have their own workspace and publish data products in a shared workspace

Dremio Sonar can govern access to virtual data products and protect sensitive data

All data products published in the data marketplace (a shared space) are searchable, have supporting metadata (wiki, tagging and data lineage) and are available for consumption without the need to copy data. The wiki can contain a business glossary to help consumers understand the meaning of data in data products, who the owners are, data freshness etc., while lineage shows how a data product was created. Dremio Sonar offers role-based access control (RBAC) to govern data product security. User defined functions can also be created to mask personal data to ensure compliance with any data privacy legislation if the data product contains sensitive data.

Data is engineered by combining the SQL queries in different virtual views

Virtual data products are published in a shared space that acts as a data marketplace for consumers to find business-ready data

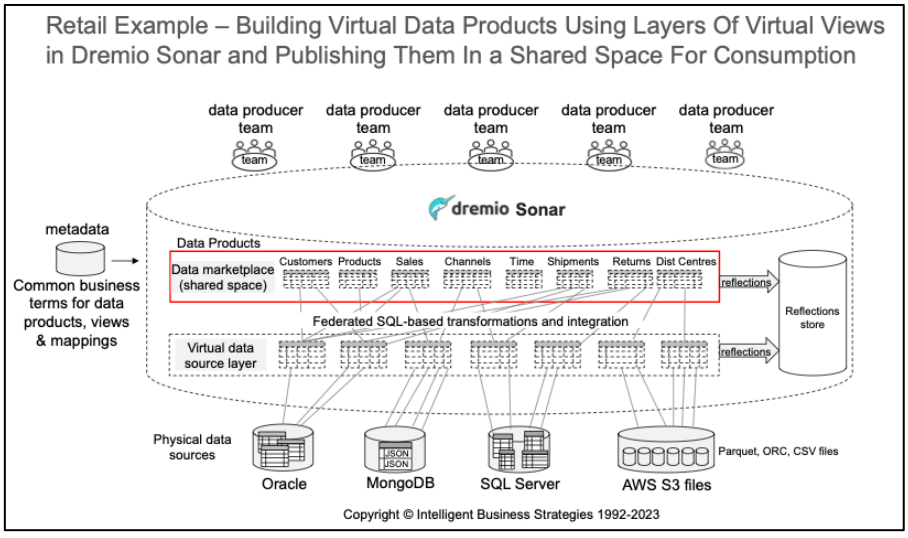


Figure 9

Consumers can find, request access to and consume data in virtual data products from a shared space

Figure 10 shows the consumption side on how virtual data products can be searched for and assembled into virtual star schemas and aggregate views created for use in self-service analytics all using common business terms.

Virtual data products can be used to create virtual data marts for use in self-service analytics all without copying data

Common data names ensure incremental creation of a universal semantic layer to guarantee consistent data meaning across all tools

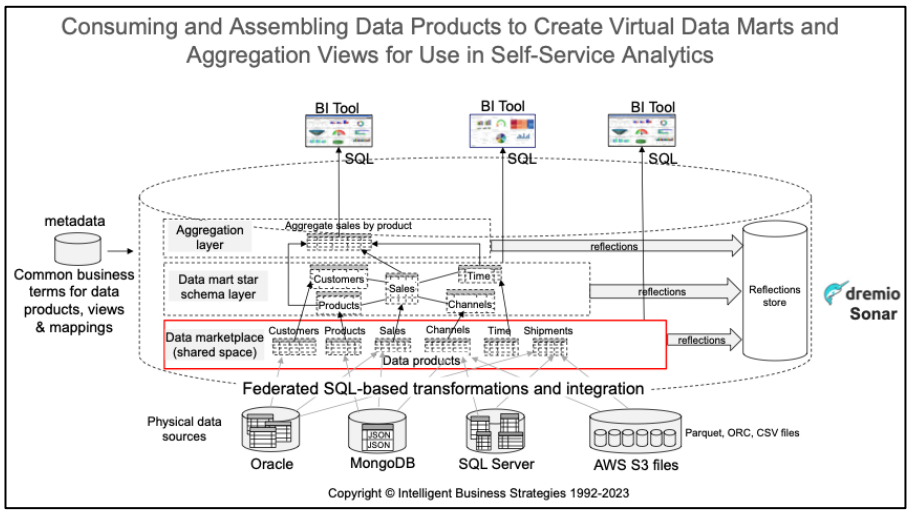


Figure 10

Virtual data marts can inherit the common data names. Query execution from BI tools can be accelerated using columnar cloud caching, data reflections and pushdown optimization. This breeds consistency and confidence.

Phase 2 – Deliver a Data Lakehouse

Migrate legacy data lakes and ingest source data into an Iceberg lakehouse managed by Dremio Arctic to guarantee consistent data, and allow teams of data engineers to build virtual data products leveraging data as code

The shift to using SaaS transaction applications and the adoption of Internet of Things (IoT) has seen a huge amount of data being captured outside the firewall. This data is increasingly being brought into cloud storage to stage and integrate it with other data for input into cloud-based data warehouses and for use in data science. The second phase is to convert this data to Parquet files to ready it for use in a lakehouse. In addition, data in legacy on-premises data lakes such as HDFS should also be migrated to columnar Parquet files in object storage. As on-premises data warehouses are migrated to the cloud, data in data warehouse staging tables should also be added into object storage if data is from data sources not being ingested already. This decouples staged source data from single analytical systems.

Turn a data lake into a lakehouse and reduce data copying

At this point Dremio Arctic with Iceberg should be deployed to enable Iceberg tables to be created and overlaid on top of the files in object storage. This turns the data lake into a lakehouse, and centralizes data needed and makes it possible to create data products once for use in multiple analytical use cases such as self-service analytics and data science.

Phase 3 – Enterprise Data Mesh

Phase 3 enables data ingestion to be managed and the adoption of Iceberg open table format while maintaining data consistency

After migrating your workloads to object storage, you are now on a path to an enterprise data mesh. In phase three, you can start building a data lakehouse centered around Apache Iceberg. This allows you to simplify and manage your lakehouse and also control decentralized data product development using Dremio Arctic. With Dremio Arctic and Dremio Sonar on top of your data in Iceberg tables, virtual data products can be produced, published, discovered, understood, assembled and accessed via a common semantic layer. This ensures redundancy is minimized when you are building data products and that data is consistent.

Dremio Arctic provides lakehouse management

Dremio Arctic enables:

- Data ingestion to be managed - new data can be ingested into your cloud data lakehouse and checked for quality before making it available to data engineers to integrate to build data products.

Decentralized teams can build and manage data products in isolation and publish them for consumption in a shared searchable space without impacting data consumers

Consumer queries run in parallel on compressed columnar storage

Federated ownership and governance of data products

Self-service access across decentralized teams

- Data to be streamed into Iceberg tables in columnar format by one engine (e.g., Spark or Flink) with full ACID support and transformed and integrated to create up-to-date data products by Dremio Sonar
- Decentralized teams can engineer lakehouse data using Dremio Sonar in complete isolation and check it before making new or new versions of virtual data products available to others. Dremio Sonar and Arctic therefore enable virtual data product development.
- Iceberg tables to be overlayed on columnar files which means data compression, automatic partitioning, data skipping, indexing, statistics and parallel query processing are all possible on object storage-based data lakes during data product development and when querying data in virtual data products in self-service analytics
- All updates to data can be audited and tracked
- Virtual data products can be consumed into cloud-based data warehouses with virtual data marts built on Dremio Sonar
- Data scientists can consume virtual data products, to provide features for building ML models and do it all in isolation without copying data
- Business analysts can run reports on virtual data marts built using virtual data products and also run BI reports knowing they are working with the most consistent and up to date version of their data
- Automatic data optimization for Iceberg tables, including compaction to write smaller files into larger files, and garbage collection, which removes unused files, to optimize performance and storage utilization

Figure 11 is a variation on Figure 9 and shows how Dremio Sonar and Dremio Arctic can enable decentralized teams of domain-oriented data engineers to create their own workspaces to build virtual data products in a data mesh on consistent data in an Iceberg lakehouse. It is Dremio's intention to extend the use of Arctic to offer it as a source for on-premises object stores in the near future which would allow Dremio to use Sonar to create a universal semantic layer for self-service analytics across data in a hybrid open lakehouse.

Virtual data products can be consumed and used in data science and combined to create virtual star-schema data marts for use in self-service analytics all without copying data

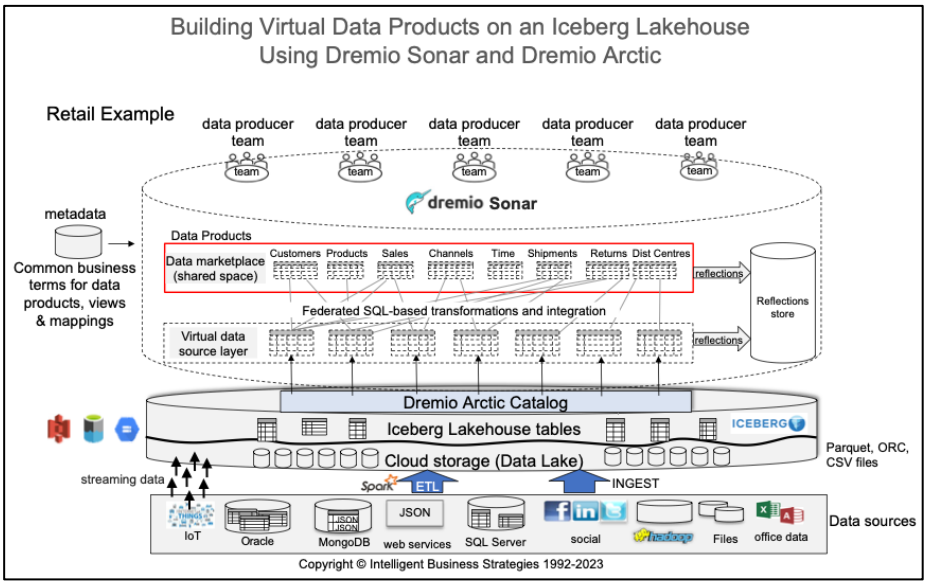


Figure 11

Dremio Sonar allows virtual data products to be tagged and organized to make them easier to find, understand and use

Dremio Sonar also provides full data lineage to explain how data products have been created

Common data names make it possible to create a universal semantic layer to drive consistent understanding across all tools

Note that the data products in the shared space data marketplace could be organized. For example, they could be organized into:

- Master data products (these are cross domain – e.g., customers)
- Transaction data products (e.g., sales, shipments, returns)
- Aggregate data products (domain specific)

Once data products have been created, data owners can be assigned, and policies created to govern access to this data before they are made available to consumers. Federated data governance can be implemented by enforcing enterprise-wide data privacy policies to protect personally identifiable information (PII) data everywhere and implementing domain specific data product access security policies (set by decentralized domain data product owners) to govern access to specific data products. Dremio Sonar can enforce both data privacy and data access security at run time to implement federated computational data governance in a data mesh.

Finally, as shown in Figure 4, by using common data names for all data products, in all virtual data marts and all virtual aggregate views, it is possible to establish a universal semantic layer in Dremio Sonar to guarantee that all BI tools and data science notebooks see consistent data names and definitions.

CONCLUSIONS

Insatiable demand to analyze more and more data has resulted in IT becoming a data engineering bottleneck

For many companies today, the demand to analyze more data from more data sources by more business users has become so great that IT can't keep pace with requests to engineer data. This has led to IT becoming a bottleneck to producing richer business insights. To overcome this bottleneck, businesses are looking to upskill people around the enterprise to increase the number of data engineers.

Companies want to democratize and coordinate decentralized data engineering activity to produce reusable data products for use in self-service analytics and data science

They also want to provision data without the need to copy it

At the same time, they want to avoid mistakes of the past including the chaos of 'every person for themselves' self-service data preparation using many different tools and siloed analytical systems with overlapping subsets of the same data being repeatedly re-engineered for different data warehouses, data marts, graph databases and machine-learning models. Instead, they are looking to move towards a decentralized *but coordinated*, incremental development approach to data engineering where different business domain-oriented teams of data engineers around the enterprise integrate data to produce reusable data products and make them available for sharing. This is the data mesh approach which aims to build data products once and reuse them for many different analytical use cases including data warehouses and data science. The data mesh approach separates the data producers who engineer data to create data products and data consumers who select the data products they need and consume and assemble them for use in self-service analytics and data science. The challenge is to avoid every consumer wanting copies of data products. Therefore, a zero data copy approach to provisioning data products is needed. In addition, consumers can produce new data products (e.g., aggregate data) and new analytical products (e.g., ML models, BI reports) and also publish these.

Data Mesh enables incremental development of data products and should help to progressively shorten time to value

The benefit of data mesh is that as more data (and analytical) products are incrementally created, consumers should be able to get faster at delivering value because more and more of what they need is already available for reuse.

A common self-service data platform is needed to coordinate decentralized data product development and facilitate data sharing in a data mesh

An open table format and catalog are needed if firms want to integrate data warehouse and data science workloads on a data lakehouse

A universal semantic layer is also needed to drive common understanding of data across multiple tools

With so much demand, companies need a common platform to build data products quickly and easily where projects can be coordinated, where decentralized teams can engineer data in isolation using DataOps (DevOps applied to data) practices, and where metadata can be shared. In addition, they need to connect to a number of data sources and either leave the data where it is and engineer it from there or move it to a centralized data lake where data consistency must be upheld. If it is the latter, cloud storage is not enough. A lakehouse is needed that supports ACID transactions, that can cater for schema evolution, that supports managed data ingestion and where teams can engineer in their own workspaces in isolation until it is ready for use. Furthermore, once produced, data products need to be published in a shared data marketplace where consumers can request access, where access security and data privacy is governed and where data can be provisioned without the need to copy it for use in self-service analytics. Also, common business data names need to be enforced to create a universal semantic layer so that all tools accessing this data see consistent, and meaningful business data names. Dremio provides all these things and can be used to industrialize development of data products in a data mesh. For companies looking for software to help them to do this, it should be a contender on anyone's shortlist.

About Intelligent Business Strategies

Intelligent Business Strategies is an independent research, education, and consulting company whose goal is to help companies understand and exploit new developments in business intelligence, machine-learning, advanced analytics, data management, big data, and enterprise business integration. Together, these technologies help an organization become an *intelligent business*.

Author

Mike Ferguson is Managing Director of Intelligent Business Strategies Limited. As an independent IT industry analyst and consultant, he specializes in BI / analytics and data management. With over 40 years of IT experience, Mike has consulted for dozens of companies on BI/Analytics, data strategy, technology selection, enterprise architecture, and data management. Mike is also conference chairman of Big Data LDN, the largest data and analytics conference in Europe and a member of the EDM Council CDMC Executive Advisory Board. He has spoken at events all over the world and written numerous papers and articles. Formerly he was a principal and co-founder of Codd and Date – the inventors of the Relational Model, a Chief Architect at Teradata on the Teradata DBMS, and European Managing Director of Database Associates. He teaches popular master classes in Data Warehouse Modernization, Big Data, Centralized Data Governance of a Distributed Data Landscape, Practical Guidelines for Implementing a Data Mesh, Machine-learning and Advanced Analytics, and Embedded Analytics, Intelligent Apps and AI Automation.

Telephone: (+44)1625 520700
Internet URL: www.intelligentbusiness.biz
E-Mail: info@intelligentbusiness.biz

Simplifying Data Mesh for Self-Service Analytics on an Open Data Lakehouse