HOW TO KNIT YOUR DATA MESH ON SNOWFLAKE

snowflake®

WHITE PAPER

TABLE OF CONTENTS

INTRODUCTION

Data mesh has become an increasingly popular approach to data management in recent years. Companies in all industries are choosing data mesh for decentralized data management to improve data agility and avoid the bottlenecks often connected with centralized approaches.

Data mesh is primarily an organizational approach that defines responsibilities and coordination across separate domain teams and their data products. However, suitable technology is needed to enable the domains to create, share, govern and monitor data products effectively.

The data mesh methodology is based on four principles:

1. Domain-oriented architecture and ownership of data
2. Data as a product
3. Self-service data platform
4. Federated governance

These four principles require much more than technology. For example, domain-oriented ownership of data requires various organizational and behavioral patterns to be successful. Similarly, managing data as a product is an overall strategy and a mindset toward data assets. Effective governance is heavily dependent on both well-defined processes and people who own different responsibilities across an organization.

A data mesh approach involves a lot more than technology, but it still must be *supported* by technology. This white paper explores the technical and architectural support that the Snowflake platform can provide to a data mesh implementation. For a discussion of the organizational and nontechnical factors involved in data mesh, please see **this blog post**.

WHAT IS A DATA PRODUCT?

A data product is defined as the combination of data plus metadata, code and infrastructure dependencies.

While this document focuses on Snowflake, we fully appreciate that a data management platform to support a data mesh rarely consists of a single product. Companies that have chosen Snowflake as a core component for their data mesh platform often also use Snowflake partner products or CSP services for specific purposes, such as automation, low-code/no-code data transformations or enterprisewide data cataloging. The selection of tools and services depends on a company's requirements, preferences and existing IT landscape.

THE SNOWFLAKE APPROACH TO DATA MESH

After working with hundreds of customers on their data mesh initiatives, Snowflake has embraced the following approach:

- We recognize data mesh as an organizational transformation with many nontechnical implications, but also often requires changes at the IT architecture and technology level.
- Be pragmatic. We advise our customers not to aim at implementing the "perfect" data mesh but to be guided by their specific pain points and objectives. A data mesh may work for one part of the organization, while the Snowflake Data Cloud Architecture may be a better fit for another.
- Define incentives and success criteria early on, including measurable KPIs for domains, data products, the self-service data platform and governance controls.
- There is no out-of-the-box data mesh solution. We embrace our extensive partner network to build joint solutions that meet client requirements.
- Start small, expand incrementally and work your way up along the data mesh maturity curve over time. For example, start with one or two domains and data products to satisfy an immediate business need, and then exploit the early success to expand the mesh.
- Be mindful of cost and complexity. For example, it has proven beneficial to keep the set of tools in the self-service data platform as small and consistent as possible across all domains while satisfying all critical domain requirements. Standardizing and automating the data product platform across domains can also significantly reduce complexity and cost.

RELEVANT SNOWFLAKE CAPABILITIES FOR DATA MESH

Snowflake is a single integrated and fully managed cloud service for data warehousing, data lake, data sharing, AI/ML, data engineering, governance and applications. It offers several key capabilities that help organizations build the self-service data platform for a data mesh.

Companies that exploit the breadth of the Snowflake platform (Figure 1) can often reduce the total number of products and services in their data mesh platform, which in turn helps reduce complexity, cost and operational burden. **Snowflake Horizon Catalog** provides an integrated and consistent approach to data governance and access control across usage scenarios as different as data engineering, AI/ML and Iceberg-based data lakes, for example.

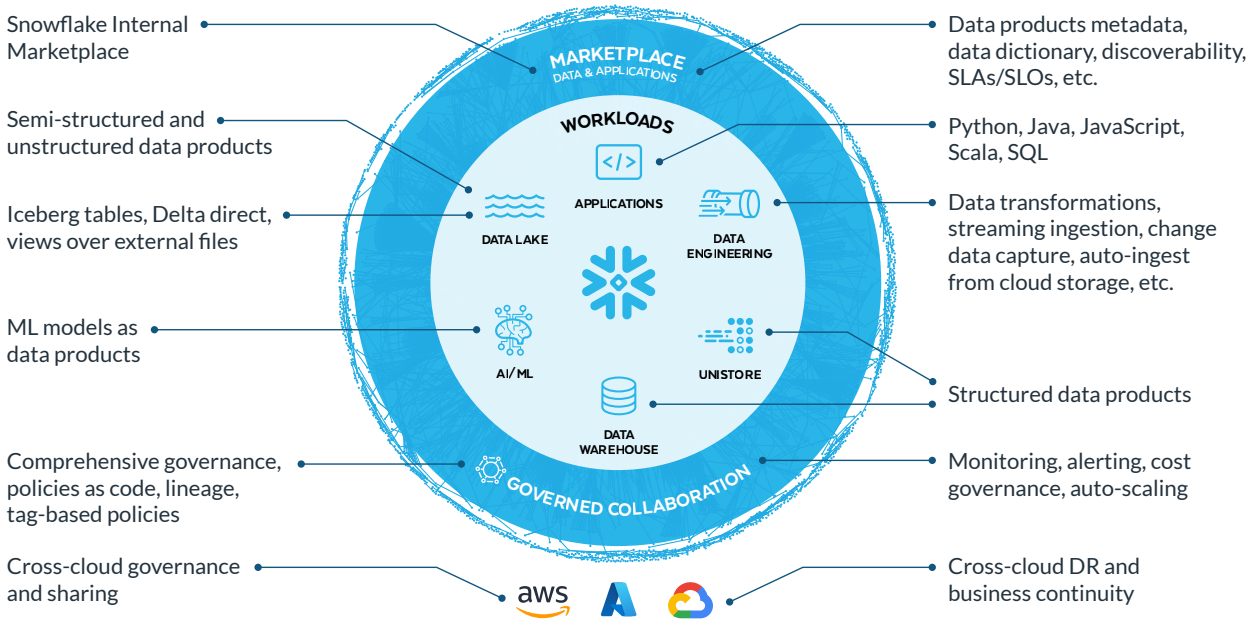

FIGURE 1: SNOWFLAKE IS A SINGLE PLATFORM THAT SUPPORTS MULTIPLE TYPES OF DATA AND WORKLOADS.

Snowflake is a distributed platform, not a monolith

As a distributed yet interconnected platform, Snowflake avoids silos and enables distributed teams to exchange data in a governed and secure manner. A company can create one or many Snowflake accounts that can reside in the same or in different cloud regions and platforms (Figure 2). Each account can be home to many logically separate databases for which compute and storage resources can be deployed and scaled independently, in a distributed way.

Different domain teams can work autonomously using independent compute power in separate databases across one or more accounts while still using the underlying Snowflake platform to share data products with each other. Note that the Snowflake concept of a "database" is not merely a traditional relational database, but an environment in which users can exploit all the functional capabilities in Snowflake as shown in Figure 1.

Using compute clusters to combine and process data from multiple databases or accounts is a core capability of the Snowflake platform. Domain teams that share a single Snowflake account or operate across multiple accounts can use **Snowflake Internal Marketplace** to publish and share data products with each other (see Figure 2). On top of that, Snowflake Horizon Catalog offers a rich set of governance capabilities to understand, track and protect your data assets, including across cloud regions and platforms if needed.

Let's take a more detailed look at how all this applies to data mesh.

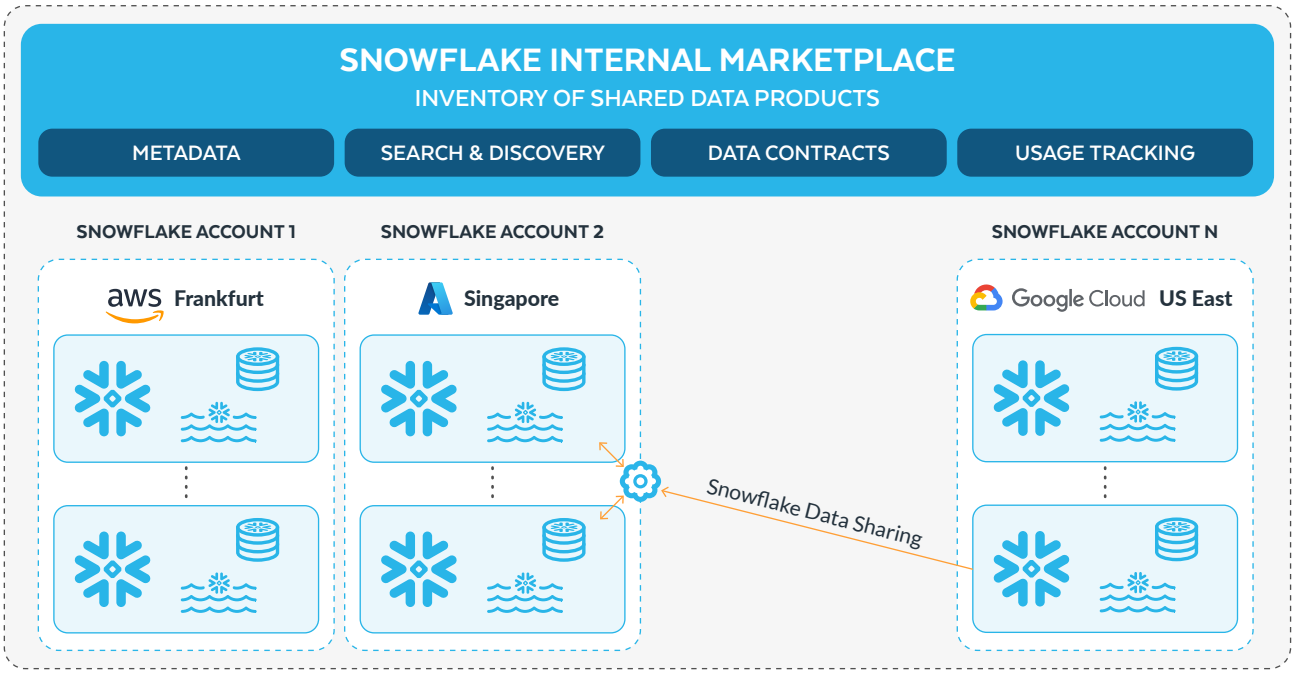

FIGURE 2: SNOWFLAKE IS A DISTRIBUTED PLATFORM

DATA PRODUCTS IN SNOWFLAKE

In a data mesh, each domain creates, maintains and owns one or more data products that are shared with other domains and data consumers. A data product is defined as the combination of data plus metadata, code and infrastructure dependencies. Treating data as a product requires a product-oriented mindset that must become an organizational habit.

What can be in a data product

Two key ingredients of a data product are the data ***artifacts*** (the payload) being shared and the ***metadata*** that describes these artifacts.

In Snowflake, a data product can contain various artifacts such as:

- Tables and views from one or more schemas or databases
- External tables referencing files in cloud storage buckets outside of Snowflake (Parquet, JSON, CSV, etc.)
- External tables pointing to files in S3-compatible on-premise storage systems
- Iceberg tables
- Semi-structured data such as JSON or XML documents, either as files or as objects in an Snowflake VARIANT column
- Unstructured data such as images, video, PDF documents, etc. sitting in Snowflake internal or external storage buckets
- User-defined functions (UDFs)
- ML models
- Snowflake Native Apps
- Table-functions accessing external services or data sources
- URLs to non-Snowflake artifacts such as Power BI or Tableau reports
- Data quality metrics

The details of *how* such artifacts, as well as their metadata, can be shared as data products depend on how you choose to implement and manage data products, as discussed in the next section.

How to implement and manage data products

There is no single right or wrong way to implement data products in Snowflake. We have seen that different requirements and preferences justify different implementation choices for data products. For example:

- Some Snowflake clients treat **individual objects** (table, views, functions, etc.) as data products and wrap management processes around them using Snowflake capabilities and/or third-party tools.
- Another option is to use a **separate database schema** for each data product. This allows users to create and manage multi-object data products with the schema being the product "container."

  Optionally, schemas can be defined with future grants so users who have access to a schema will automatically have access to any new objects that might get added to a schema. This can be useful for data products. If the data product owner adds, for example, another view to a data product (schema), then existing consumers of the data product won't need to be authorized explicitly to use this extension of the product.
- Some companies have decided on managing more complex data products and are using a **separate Snowflake database** for each data product. This can be a valid approach too, if it fits the requirements at hand. It also raises interesting questions about the optimal granularity of data products, which can vary by use case.
- Data products can also be implemented as **Snowflake listings** for intra-company (or even cross-company) management and sharing of data products.

While each of these options can be very successful, we think that implementing data products as Snowflake listings is particularly powerful for the reasons outlined in the following section.

Snowflake supports a variety of input and output ports for data products, including streaming ingestion, a Kafka connector, a Spark connector, a Dataframe API, automatic data ingestion from cloud storage buckets, a REST API, file formats, and of course SQL APIs such as JDBC, ODBC, .NET and APIs for many popular programming languages.

Representing and managing data products as Snowflake listings

Using Snowflake listings as a vehicle for data products has three key benefits.

First, a listing enables users to **bundle data and metadata items into a single unit** to facilitate discovery, sharing, governing and tracking of data products. This makes a data product understandable and trustworthy. A listing can include:

- Data objects (structured, semi-structured or unstructured)
- Functions, applications or models
- Ownership and contact information, e.g. for the domain or product owner
- Data documentation, e.g. a business description
- Data dictionary, e.g. column-level data type and semantic information, or the signature and description of a function, etc.
- Sample data
- Data quality metrics
- Usage examples
- Service-level objectives, such as a refresh interval for the product
- Links to terms and conditions
- Links to further data documentation

Second, listings enable **easy and efficient management of data products** by providing capabilities such as:

- Sharing data products across cloud regions and cloud platforms as needed, without any setup, while minimizing data movement and egress costs
- A Listings API and a YAML representation of listings to manage data products programmatically or integrate with third-party tools
- Built-in versioning of data product definitions and their metadata (private preview)
- Additional governance and monitoring at the data product level, rather than at the level of the individual data objects

And third, an organizational listing makes a data product **discoverable and accessible through Snowflake Internal Marketplace**, which we discuss in the next section. Listings can also be integrated with other discovery products, such as a third-party data catalog or self-built data portal.

SNOWFLAKE INTERNAL MARKETPLACE

Snowflake Internal Marketplace is conceptually similar to **Snowflake Marketplace** but designed for use within a single company or organization.

Internal Marketplace enables **data product owners** to:

- Publish well-documented data products as listings within the organization
- Associate the data product with its domain (public preview)
- Make the data product discoverable and accessible to roles within and/or across Snowflake accounts
- Easily define or dynamically adjust who can discover and/or access the product
- Optionally enforce an access request and approval process for data product consumers
- Easily define or dynamically adjust fine-grained access policies
- Monitor data product access and usage statistics

Internal Marketplace enables **data product consumers** to:

- Discover data products through **Snowflake Universal Search**
- Discover data products by browsing the marketplace and filtering by domain and other dimensions
- Understand data products by reviewing their metadata and sample data
- Gain immediate access to data products (if authorized) or request access from the data owner or designated data steward
- Combine multiple data products from one or different owners
- Combine data products with their own data and to build and share new data products
- Conveniently access a data product via its Uniform Listing Locator (ULL)
- Use data products within Snowflake using SQL, Java, Javascript, Python or any containerized application within Snowflake
- Use data products in applications outside of Snowflake via a broad range of APIs

DATA CONTRACTS

Sharing data products between data producers and consumers in a data mesh works best if all relevant data product properties are clearly communicated from data product owners to consumers. The communication should be readable by humans and machines to enable programmatic management and verification of data products. This requirement fueled the rapidly emerging concept of **data contracts.**

A data contract documents and communicates the structure, format, purpose, quality and other properties of a data product. The contract is maintained by the data provider and shared with the data consumers. It describes the objectives and possibly commitments of the data provider with respect to the properties of the data product. The contract helps data consumers understand what to expect from a given data product.

Listings represent data products and their metadata in Snowflake. At the same time, listings can serve as a lightweight data contract because they already contain many of the informational items that are typically expected in a data contract (such as ownership information, a business description and schema information). The process to include data quality metrics and information about data freshness or other SLAs in a listing is straightforward, which helps ensure the contract information is shared seamlessly along with the data product metadata.

Snowflake represents the contract and metadata of a listing not only in the UI but also as a YAML file to enable programmatic management of data products and integration with other tools. Data providers and/or data consumers can choose to monitor the compliance of a data product with a data contract, e.g. by using Snowflake alerts. An alert can monitor arbitrary properties of a data product, its payload, and its metadata and execute any desired action or notification if those properties meet (or fail to meet) specified criteria. Notifications can be emitted via email, Slack, MS-Teams, PagerDuty or your cloud provider's messaging system (such as Amazon SNS, Azure EVent Grid, Google Pub/Sub).

SNOWFLAKE SUPPORT FOR DOMAINS

A domain in data mesh is very much an organizational concept. A domain is typically some organizational unit that has a common responsibility and/or need for data. Domains might correspond to business units, departments, actual or virtual teams, or some other scope of a shared interest or responsibility. Domains are subject to critical nontechnical considerations that we discuss in this article.

From a technical perspective, Snowflake supports organizational domains in two significant ways:

1. Domain identity and ownership of data products (public preview)
2. Domain environments to produce and consume data products

Domain identity and data product ownership

Ownership is critically important for managing data as a product. Ownership implies a certain set of responsibilities for a data product, indicates the origin of the data product, and is the basis for communication between data product providers and data consumers.

In Snowflake, domain identity and data product ownership is manifested by the concept of data provider *profiles*. Provider profiles have been available in Snowflake Marketplace for years and are now coming to Internal Marketplace as internal profiles, known as *organizational profiles*. Each team, business unit or other type of domain that publishes data products has its own profile. A profile specifies the domain name, a description, contact information, domain-specific approval process for access requests to data products, and other information. A profile also defines the members of a domain by specifying one or more Snowflake roles across one or more Snowflake accounts.

When a data product owner who belongs to a given domain publishes a data product (listing), they can select the domain's profile so the new data product links to it. This enables a number of capabilities in Internal Marketplace:

- Each data product clearly shows a domain profile and contact information as owners
- Data consumers can navigate from the data product to the domain profile to see further ownership details as well as other data products published by the same domain
- Data products are categorized and can easily be browsed by domain
- If a domain requires explicit access requests and approvals for some or all of its data products, then the domain's default approval process is automatically linked to the product

A domain can customize the UI appearance of its profile with a color, icon and avatar to establish "domain branding" and recognizability on Internal Marketplace.

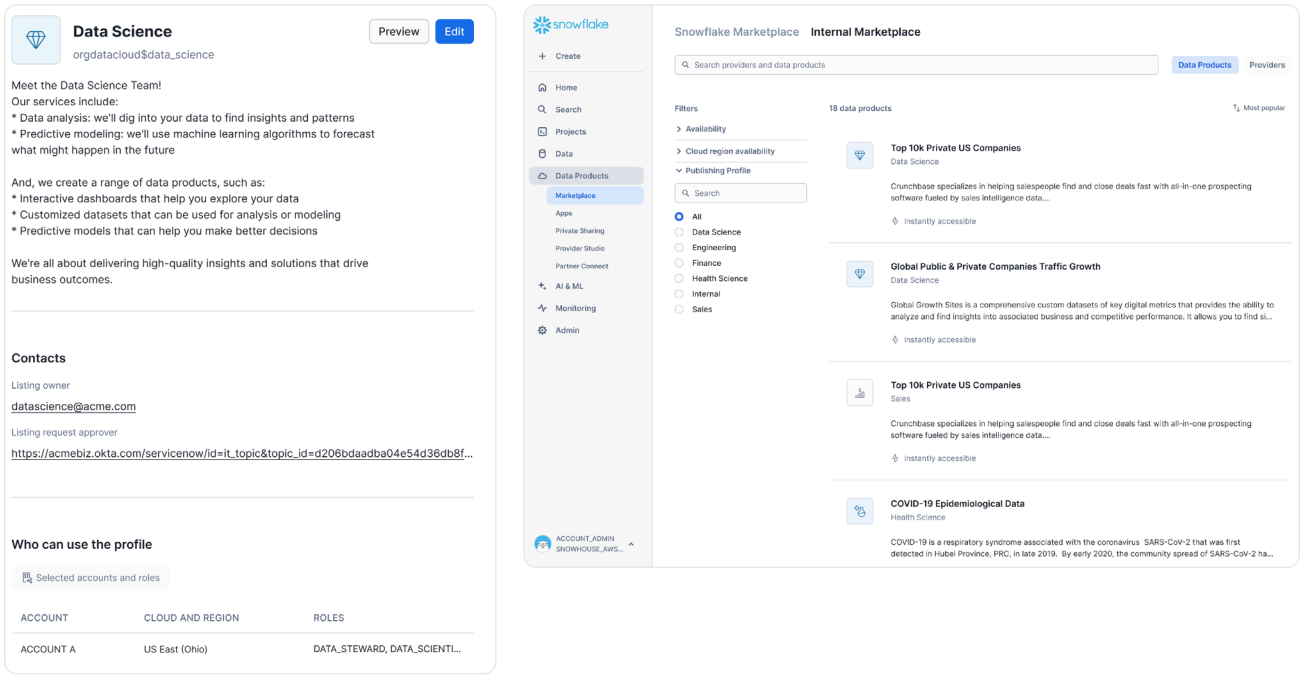

FIGURE 3: WHEN A DATA PRODUCT OWNER PUBLISHES A DATA PRODUCT (LISTING), THEY CAN SELECT THE DOMAIN'S PROFILE. EACH PROFILE INCLUDES A DETAILED DESCRIPTION OF THE DOMAIN ALONG WITH CONTACT INFORMATION, A LIST OF MEMBERS AND MORE (SEE LEFT IMAGE). YOU CAN ALSO FILTER LISTINGS BY OWNERSHIP AND MANY OTHER CATEGORIES (SEE RIGHT IMAGE).

Domain environments

Domain teams often use separate environments to create, manage and share their data products. This is motivated by their need to be independent and agile with a certain degree of autonomous control over their internal processes for data product management. (**Conway's law** and its implications play a significant role in data mesh.)

Organizations have a variety of options for providing domains with dedicated environments in the Snowflake distributed architecture shown in Figure 2. Each domain can work independently but without being isolated or disconnected. **These topology options allow domains to manage and share data products as organizational listings and through Snowflake Internal Marketplace.** Therefore, listings provide a single unified and consistent mechanism for managing and sharing data products regardless of the chosen topology or the future *evolution* of that topology.

The topology options are:

Account per domain: Each domain one or more separate Snowflake accounts

- Maximum isolation between domains
- Different domains can operate in different cloud regions and cloud platforms
- Enables a multi-region and multi-cloud data mesh with a Snowflake as the common data product and governance layer across regions and clouds
- Different domains can have different security and network configuration as well as separate identity providers
- Central and aggregated monitoring across accounts is available

Database per domain: Each domain uses one or more separate Snowflake databases

- All these databases are managed in a single Snowflake account in a single cloud region
- User and security management can easily be managed across domains
- Each domain team can still spin up and scale their own computer clusters independent from other domains
- Database roles enable domain admin to manage permissions and access control within their databases autonomously, if desired

Schema per domain: Each domain uses separate schemas in a single database

- Lowest degree of isolation between domain environments
- Each domain team can still spin up and scale their own computer clusters
- Potentially higher effort in naming conventions to distinguish between objects from different domains
- Can be useful for subdomains in a domain/subdomain scenario
- Less commonly used than the previous options

Hybrid approach: Domains use a mix of the three options

- Some domains use separate Snowflake accounts while others use separate databases or schemas
- Can be useful if some domains are much larger, more mature or more autonomous than others.
- Potentially higher effort in provisioning, monitoring and managing domain environments

Further architecture variations derived from these base types are possible. Also, the environment that a domain team uses often consists of Snowflake plus additional tools, based on their requirements and skills.

Domain environments are often provided by a central data platform team that is responsible for the availability and administration of the data mesh platform. Snowflake enables companies to implement the desired balance between central standardization and oversight versus local domain autonomy and flexibility. Determining the "right" balance is very much an organizational question that depends heavily on a variety of organizational factors such as size, IT legacy, maturity, skills, culture and organizational structure.

FEDERATED GOVERNANCE WITH SNOWFLAKE

Federated governance is arguably one of the most challenging parts of a data mesh journey. It requires the definition of governance objectives, processes and metrics to monitor and audit data product governance. While the success of federated governance depends heavily on nontechnical and operational factors, it also requires a suitable technical implementation.

At the platform level, Snowflake Horizon offers a broad range of **governance capabilities** that help users implement and monitor data product governance, such as:

- Role-based access control
- Access policies
  - Row-level access policies
  - Column-level data masking
  - Aggregation and projection policies to restrict permissible usage of data products
  - Differential privacy policies
- External tokenization
- Data lineage and dependencies
- Data classification, i.e. auto-detection and tagging of PII and other sensitive data
- Audit of practically all user and data related activities in Snowflake
- Data quality metrics
- Object tags

We are going to focus on three of the features as examples and their relevance to data mesh: object tagging, access policies and data quality metrics.

Object tagging

In Snowflake, users can create and assign one or more tags (key-value pairs) to almost any kind of object in Snowflake, such as databases, schemas, tables, columns, compute clusters, users, roles, tasks and other objects. Tags are inherited through the object hierarchy and can be exploited to discover, track, restrict, monitor and audit objects based on user-defined semantics. Additionally, tag-based access policies enable users to associate an access restriction with a tag so that the access policy is automatically applied to any data object that carries the relevant tag.

A data product owner can apply tags to schemas, tables, views or individual columns in order to annotate these objects with useful metainformation. Users can create, alter and drop their own custom tags and optionally specify permissible values for a given tag. For example:

```
create tag my_tags.sensitivity
  allowed_values 'Low', 'Medium', 'High';

create tag my_tags.data_product_category
  allowed_values 'Raw', 'Intermediate',
  'Curated', 'Master';


alter table sales.customers
  set tags my_tags.sensitivity = 'High';

alter view sales.orders
  set tags my_tags.sensitivity = 'Medium';

alter table sales.orders
  set tags my_tags.data_product_category = 'Raw';
```

Similarly, users can apply tags to specify arbitrary metainformation such as data origin, line of business, data categories and subcategories, taxonomy, business terms, use cases, cost center, retention periods, GDPR relevance, certification of quality, or anything else they deem useful.

Tags enable data product owners to make their products more understandable for data consumers. Access policies and other governance controls can also check for the existence of specific tags and/or their values to customize how governance controls are applied to data objects.

In data mesh, federated governance often involves the definition of global cross-domain governance standards with the responsibility for domain and data product owners to apply and implement these standards for their data products. For this purpose, Snowflake separates the *definition* of most governance controls (such as tags, policies, quality metrics, etc.) from the *application* of these controls to data objects. This enables domain owners to agree on common tags or policies *across domains* while leaving it to each domain to apply them to their data products as needed.

Access policies

Access policies in Snowflake give data product owners fine-grained control over who can access which part of a data product and how. Such policies can take effect within and across databases in a single Snowflake account, but also across Snowflake accounts in different regions and cloud platforms. This makes Snowflake not only an excellent data product layer but also a multi-cloud data governance layer. Data owners have the ability to restrict or adjust data product access dynamically and at all times, multi-region and multi-cloud.

Row- and column-based policies control which views and slices of tables in a data product a particular data consumer can see. This avoids the need to create different data products or different views for different consumers of data from the same tables.

Protection policies prevent data consumers from including protected columns in the result set of a query while allowing those columns to be used in predicates, join conditions or group by clauses.

Aggregation policies prohibit access to individual records but allow queries that aggregate these records into groups of a specified minimum size. This can be used, for example, to prevent access to individual customer records while allowing analytics that aggregate at least 100 customers records:

```
CREATE AGGREGATION POLICY my_agg_policy
  AS () RETURNS AGGREGATION_CONSTRAINT ->
    CASE
      WHEN CURRENT_ACCOUNT_NAME() = 'DOMAIN_D3'
        THEN NO_AGGREGATION_CONSTRAINT()
      ELSE AGGREGATION_CONSTRAINT(MIN_GROUP_SIZE
        => 100)
    END;
ALTER TABLE customer SET AGGREGATION POLICY
my_agg_policy;
```

Policies can contain arbitrary SQL conditions to express the desired logic that decides data access when a data consumer reads a data product. Optionally, the SQL can contain a subquery to a custom mapping table that describes permissible access declaratively.

Context functions such as `current_account_name()`, `current_role()`, and others can be used to apply different conditions to different types of data consumers or domains within or across Snowflake accounts, multi-region and multi-cloud.

In the spirit of federated governance Snowflake enables companies to define standard policies *across domains* for the protection of certain types of data but leave it to the responsibility of data product owners to apply these policies to their data objects as needed.

Additionally, you can define tag-based policies that are automatically applied to columns, views or tables if they carry the tag that is referenced in the policies. And finally, the governance dashboard in the Snowflake UI as well as custom alerts can be used to detect unprotected data.

Data quality metrics

Maintaining data product quality is a particularly challenging part of data governance. In a data mesh, data product owners should monitor, maintain and communicate the quality of their data products to make them trustworthy.

In Snowflake, data product owners can define their own custom data quality metrics or use a set of built-in metrics such as null count, freshness, unique count, duplicate count and various others. In a custom quality metric you can use SQL to express any desired technical or business condition that represents a useful measure of quality. For example, you can check permissible value ranges, date and time formats, expected data distributions, valid string formats based on regular expressions or other functions, or other single-column and multi-column conditions. The SQL expression can also reference multiple tables or views to validate referential integrity or other cross-object dependencies of interest.

Data product owners can associate one or more data quality metrics with the data objects in their data product. The evaluation of the metrics can be set on a schedule or, optionally, triggered by data changes in the data objects. Every evaluation of a data quality metric is recorded in a history table that provides a full audit log of all current and historic quality measurements. Desired subsets of these quality results can be included in a data product for transparency to data consumers. Snowflake alerts can be created to trigger notifications or mitigating actions when quality measurements do not meet expectations.

CONSIDERATIONS FOR YOUR DATA MESH ARCHITECTURE

There is no one-size-fits-all architecture for a data mesh platform. Snowflake provides choice, such as the different options for domain environments and data products discussed earlier in this paper. Here are a few items you should consider when building your data mesh.

Heterogeneous technologies

A recurring architecture question involves the need or desire to allow different domain teams to use different technologies and systems for their data management. This is perfectly plausible since different parts of a business often have different types of data sources and different systems of record. But it creates severe challenges when teams create data products with the intent or hope of sharing these products across the company, as indicated by the arrows in the left side of Figure 4. The challenges include:

- Enforcing consistent data and interoperability standards
- Ensuring consistent metadata
- Implementing consistent governance and access control across domains
- Avoiding data format conversions
- Avoiding the proliferation of data copies

We found that companies that attempt to share data products across heterogeneous technologies without a harmonization layer face drastically higher complexity and cost when building and operating their data mesh platform. We believe that it is essential to introduce a common data product layer across domains to simplify the consistent definition, sharing and governance of data products, as shown on the right side of Figure 4. Domains can continue to use some or all of their existing systems as needed, but should use a cross-domain platform such as Snowflake to create data products that have built-in data sharing functionality as well as consistent governance, fine-grained access control and interoperability across domains.

This does not necessarily require complete replication of all data into Snowflake. You only need the subset of data that is required to create data products. Also, Snowflake can create and share data products based on data that resides outside of Snowflake, such as Iceberg tables, unstructured and semi-structured data files, and others.

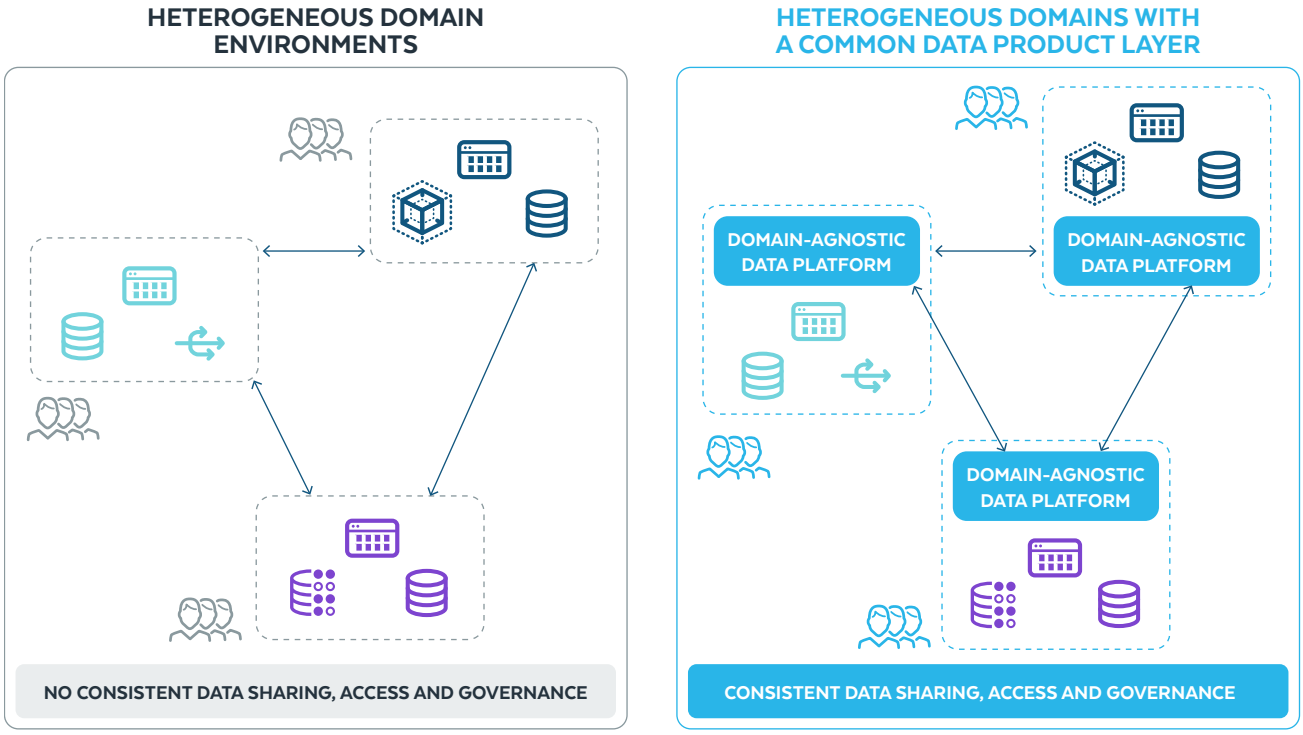

FIGURE 4: HETEROGENEOUS DOMAIN ENVIRONMENTS

We have seen some companies trying to implement the domain-agnostic data platform based on a lowest common denominator across domains, such as flat files in cloud storage buckets that can be read and written by many different data management products. This is possible but sacrifices the ability to apply fine-grained access policies and other governance controls discussed in previous sections. Additionally, companies with multi-cloud requirements must invest additional efforts to bridge across hyperscalers, which Snowflake provides out of the box.

Some companies look at data virtualization as a potential solution for integrating diverse domain environments. While there are certainly valid use cases for data virtualization, it also creates a number of challenges. One challenge is maintaining performance when the source systems are not sized for additional workloads or when data from multiple repositories needs to be joined. Such joins typically require you to move data into one common place to compute the join, even if other predicates can be pushed down to the data sources. The time and effort required to move data can prohibit virtualization for performance-sensitive use cases.

With Snowflake, however, a join across data objects in separate databases or even separate Snowflake accounts has approximately the same performance as a join across the same data objects in a single database. This can be a significant advantage for federated queries across multiple domain environments.

Semi-decentralized approaches

We have worked with several companies that selected a semi-decentralized approach when a fully decentralized operation was not feasible for their needs.

One such pattern provides each domain with a separate domain environment to accommodate a level of autonomy and agility across the domains. Each domain creates and manages data products which are then shared into a central data product, resulting in a hub-and-spoke architecture. The shared data products are typically not materialized in the central hub but use Snowflake data sharing instead. The central hub can be a separate Snowflake environment and serves as a single point of data product discovery and oversight which can include a central component of the federated data governance. This can appeal to companies that are not ready to move to a fully decentralized approach.

Another pattern combines distributed domain-oriented ownership of curated data products with a centralized responsibility for onboarding source data from a broad variety of data sources across the company. In this pattern, a central team makes source data available to all domains based on a unified storage layer such as a raw-data data lake or a lake of Iceberg tables for interoperability across technologies. The central team essentially operates like a domain that makes source-oriented data products available to the rest of the mesh to increase the ease of data usage. Common reasons for choosing this approach include a shortage of deep data integration skills across the domains that produce consumer-oriented data products, and a desire to avoid duplication or inconsistencies in the ingestion of raw data from a diverse data landscape.

Both of these semi-decentralized approaches raise the question about whether their central components are contrary to the spirit of the distributed data mesh methodology. We view this as a rather academic question. The ultimate criterion for success is whether the chosen approach helps the organization achieve their data management objectives within their given constraints of maturity, IT legacy and organizational flexibility.

Snowflake data integration with other technologies

ICEBERG

Snowflake supports data interoperability with other storage and compute engines through the use of Apache Iceberg tables. Iceberg tables are stored in your own cloud storage buckets that you manage and make accessible to other compute engines that may need access to the data.

Access to Iceberg tables requires an Iceberg catalog that maintains the pointers to the tables' metadata files. You can use Snowflake as the Iceberg catalog or use an external catalog such as AWS Glue.

An Iceberg table that uses Snowflake as the Iceberg catalog provides full Snowflake platform support with read and write access, fine-grained access policies, replication, streaming ingest, and most other Snowflake native capabilities. Other engines can also read these Iceberg tables.

An Iceberg table that uses an external catalog enables other engines to read and write while Snowflake has read-only access. You can use this option to create Iceberg tables in Snowflake based on sources such as AWS Glue Data Catalog, Delta table files, Delta Direct, a remote Iceberg REST catalog, Open Catalog or Iceberg metadata files in object storage. Snowflake can automatically sync with the external catalog to always expose the latest version of the external Iceberg table.

Iceberg can enable a broad range of integration options, including with **Microsoft Fabric**, data lakes based on **AWS Glue and S3** or **Google Cloud**.

CONNECTORS

Snowflake and Snowflake partners provide a variety of connectors to integrate various data sources and SaaS services. Some of these integrations use Snowflake data sharing techniques to make external data accessible in Snowflake without having to replicate the data into your Snowflake environment, such as the Salesforce and **ServiceNow** Connectors. Other connectors enable teams to replicate relevant subsets of data in Snowflake from sources such as **Google Analytics**, Postgres and other relational databases.

Several Snowflake partners, including **Informatica**, **Fivetran**, **SNP**, **Matillion**, **Mendix** and **Gitlab**, provide **Snowflake Native Apps** for data integration, which are available on Snowflake Marketplace. These apps run inside Snowflake and ingest data from external sources. Proven Snowflake features such as **Snowpipe**, **Snowpipe Streaming** and the **Snowflake Kafka Connector** continue to be popular choices as well. In addition, a majority of ETL/ELT tools and native cloud services integrate well with Snowflake.

Roles and access management

Setting up roles and access management for a data mesh supported by Snowflake is not a one-size-fits-all venture. The types of roles and the granularity of access management you need will depend on your organization's governance requirements, independent of Snowflake.

Different company and industry requirements lead to different design choices for data mesh access control. For some organizations, this involves three roles per domain: a read-only role, a read/write role for data product developers, and an admin role. Data consumers from other parts of the company are given the read-only role so they have read-only access to *all* of the data products from that domain. Row- and column-level policies are used to restrict access to sensitive data fields.

Other businesses require separate roles for each data product, resulting in a more elaborate approach to defining and granting roles. Also, some companies tie data product access to a specific purpose or use case: a user may have access to a data product for one particular use case, but not have access to the same data product for a different purpose. For this type of requirement, it can make sense to define one or more roles per use case rather than per data product. Yet another option involves implementing attribute-based access control, and companies may opt to use a third-party access management tool together with Snowflake.

SUMMARY: RELEVANT SNOWFLAKE CAPABILITIES FOR BUILDING AND RUNNING A DATA MESH

Let's summarize the various capabilities that Snowflake customers are often using when building and operating a platform for a data mesh. Table 1 lists a subset of the core Snowflake platform features that are relevant for the four data mesh principles.

DATA MESH PRINCIPLE	KEY SNOWFLAKE CAPABILITIES (NOT EXHAUSTIVE)
Domain-oriented ownership and architecture	• Separate databases and/or accounts for the domains • Internal data provider profiles • Distributed topology to match your business
Data products	• Internal Marketplace and organizational listings (Snowflake Horizon Catalog) • Pipelines and automation • Data quality metrics • Object tags • Data product tracking and usage analytics
Self-service data platform	• Instantly deploy or scale resources • Highly available platform over three availability zones • Ease of use, near-zero maintenance • Git integration, Snowflake CLI • Intra-domain and cross-domain monitoring, audit, alerts and cost attribution • Cross-region/cross-cloud failover and business continuity • Easy integration with third-party tools and services
Federated governance	• Snowflake Horizon Catalog • RBAC and fine-grained cross-domain access policies • Lineage • Governance dashboard

TABLE 1: KEY SNOWFLAKE CAPABILITIES FOR DATA MESH PRINCIPLES

Data mesh defines data products as the combination of data plus metadata, code and infrastructure dependencies. Examples of relevant Snowflake features for data product dimensions are shown in Table 2.

DATA PRODUCT DIMENSIONS	EXAMPLES OF SNOWFLAKE CAPABILITIES (NOT EXHAUSTIVE)
Data/payload	• One or more data assets per data product • Structured, semi-structured and unstructured data objects in Snowflake or external storage buckets • Iceberg tables stored outside of Snowflake • Functions, applications, ML models • Various non-Snowflake artifacts

Metadata	• Metadata in listings
	  ○ Ownership and contact information
	  ○ Data dictionary: column names, data types, column descriptions, etc.
	  ○ Business description of the data product
	  ○ Sample data
	  ○ Usage examples
	  ○ Data product attributes such as the refresh interval, etc.
	  ○ Links to further data documentation and terms and conditions
	• Further metadata (optional)
	  ○ Data quality metrics
	  ○ Object tags
	  ○ Object dependencies and lineage
	• Cost and usage information

Code	• Pipelines and transformations to create and refresh products • Tasks, streams, dynamic tables • UDFs and stored procedures (SQL, Python, Java, JavaScript, Scala) • Snowpark Container Services to deploy and run custom docker containers, including image registry and image repository • Snowflake devops and CI/CD features • Policies as code, such as row- and column-level access policies, tag-based data masking, object tagging and so on
Infrastructure dependencies	• Dynamic tables, tasks, notebooks, etc. can reference a desired computer cluster • Serverless capabilities in Snowflake reduce the need for infrastructure dependencies • A data product service based on a docker container can be assigned to a desired compute pool

TABLE 2: DATA PRODUCT DIMENSIONS IN SNOWFLAKE

Data mesh defines that data products should exhibit eight important properties, which are shown in Table 3; we are adding "cost-effective" and "measurable" to that list because we have repeatedly seen the importance of those two properties as they relate to data products. Table 3 lists examples of Snowflake capabilities that can help you achieve these 10 data product characteristics.

DATA PRODUCT PROPERTIES	EXAMPLES OF SNOWFLAKE CAPABILITIES (NOT EXHAUSTIVE)
Secure	RBAC, fine-grained access policies, privacy policies, encryption, tokenization, Snowflake Trust Center
Discoverable	Snowflake Internal Marketplace and listings, optional integration with third-party catalog tools
Addressable	Snowflake data shares, uniform listing locators (ULLs), standardized access across cloud platforms
Understandable	Listings with data documentation, data dictionary, sample data, usage examples and so on
Trustworthy	SLOs/SLAs such as update frequency, data quality metrics, data lineage, object dependencies and access history
Natively accessible	SQL, Python, Java, Scala, SQL APIs, REST API, dataframes and more to access multi-model data (structured, semi-structured, unstructured, various file types, etc.)
Interoperable	ANSI SQL data types, unified metadata and common APIs across domains, listings and data sharing
Valuable on its own	Composite data products that consist of multiple objects (data products can consist of data, functions, apps, etc.)
Cost-effective	Budgets, quotas, compute with auto-resume/auto-scale
Measurable	Access history, usage statistics, resource monitors

TABLE 3: DATA PRODUCT CHARACTERISTICS SUPPORTED IN SNOWFLAKE

CONCLUSION

If you determine that a data mesh is the right approach for your enterprise, make sure you focus on the organizational and nontechnical questions that are mandatory for success, such as:

- Organizational changes
- Roles and responsibilities
- Staffing
- Incentives and accountability
- Buy-in from key stakeholders
- Shifting the data mindset to product thinking

As part of building a data mesh, you will eventually need to design a self-service data platform that can support distributed domains and data products with federated governance. Snowflake can play a key role as an easy-to-use self-service platform for domain teams.

Snowflake supports different topologies that allow companies to choose the desired degree of decentralization and domain autonomy while ensuring that domains remain interconnected and interoperable. Snowflake enables single-account topologies as well as multi-region, multi-cloud architectures, and supports the integration of external domains or multi-company data sharing and collaboration. The underlying Snowflake platform, including Snowgrid cross-cloud technology and Snowflake Internal Marketplace, acts as the connecting tissue that helps organizations avoid data silos and the risks and inefficiencies that come with them.

Additionally, Snowflake provides a broad range of features to help companies implement the concepts of data as a product and federated governance across their organization. Snowflake also integrates easily with a broad range of third-party tools that provide additional platform capabilities. To learn more about how Snowflake can help you realize your data mesh vision, visit **snowflake.com/data-mesh**

ABOUT SNOWFLAKE

Snowflake makes enterprise AI easy, efficient and trusted. Thousands of companies around the globe, including hundreds of the world's largest, use Snowflake's AI Data Cloud to share data, build applications and power their business with AI. The era of enterprise AI is here.

Learn more at **snowflake.com** (NYSE: SNOW).