HOW TO KNIT YOUR DATA MESH ON SNOWFLAKE

snowflake

WHITE PAPER

Data mesh[1] 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 organizational bottlenecks often connected with centralized and monolithic approaches.

This paper discusses the Snowflake approach to data mesh. It describes some of the most critical Snowflake capabilities for a data mesh and presents typical architecture options that our clients have chosen in order to implement a self-service data platform that supports distributed domains.

Data mesh is primarily an organizational approach that defines responsibilities and coordination across separate domain teams and their data products. However, the right technology is needed to enable the domains to follow the data mesh concept in a feasible way.

> **"Data mesh is not about technology [...] but you need the right technology to enable the data product teams with a variety of capabilities. Every domain and data product team doesn't have to reinvent the wheel and start from scratch and build their data & analytics platform. Similarly, we need to make it simple for the data product teams. Without this, there is no empowerment and no decentralization."**
>
> **—OMAR KHAWAJA,** Global Head BI, Roche (2022)

Snowflake is being used successfully as a data platform by many companies that follow a data mesh approach. There is no single technology platform that provides a complete end-to-end solution to support the data mesh concept. However, Snowflake provides many of the capabilities needed for a self-service data platform, enabling a distributed, domain-driven architecture and offering capabilities to help implement data as a product and federated computational governance.

THE SNOWFLAKE APPROACH TO DATA MESH

After working with numerous clients on their data mesh initiatives, Snowflake has embraced the following approach:

- We recognize that data mesh is first and foremost an organizational transformation. This transformation has many non-technical implications but often also requires changes at the IT architecture and technology level.
- Be pragmatic. We advise our clients not to aim at implementing the "perfect" data mesh but to be guided by addressing their specific pain points and objectives. For example, polyglot storage and multi-modal access are useful concepts, but companies should focus on their actual requirements to maximize impact.
- 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 as consistent as possible across all domains while satisfying all critical domain requirements.
- 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. For example, tools for data governance, automation, DevOps, and other areas are often part of a data mesh solution, even if not discussed in detail in this article.

[1] www.thoughtworks.com/en-us/what-we-do/data-and-ai/data-mesh

RELEVANT SNOWFLAKE CAPABILITIES

Snowflake offers a number of key capabilities that our clients have found helpful in building the self-service data platform for a data mesh.

Snowflake is much more than a cloud data warehouse

Snowflake is an integrated cloud service provider that offers a broad range of capabilities for data engineering, data lakes, data warehousing, data sharing, and significant portions of a typical machine learning lifecycle.

In particular, users can build and automate data transformation pipelines to turn diverse input data into governed data products. Snowflake can operate on common file formats in your cloud storage buckets as easily as on input streams (e.g., from Kafka) or its relational tables. Supported file formats include JSON, XML, Parquet, AVRO, Delta Lake[2], Apache Iceberg[3], and others. Additionally, Snowflake has support for unstructured data, such as images, video, or other binary formats. Data can be manipulated in the Snowflake platform using SQL, Python[4], Scala, Java, and Javascript, or by invoking external functions on the broader cloud platform.

Snowflake may or may not provide all functional capabilities that your domain teams require, but it offers a significant range of capabilities in a *single* service that would otherwise require a *collection* of cloud services to be integrated. Such integration can be complex and time consuming and requires highly skilled individuals.

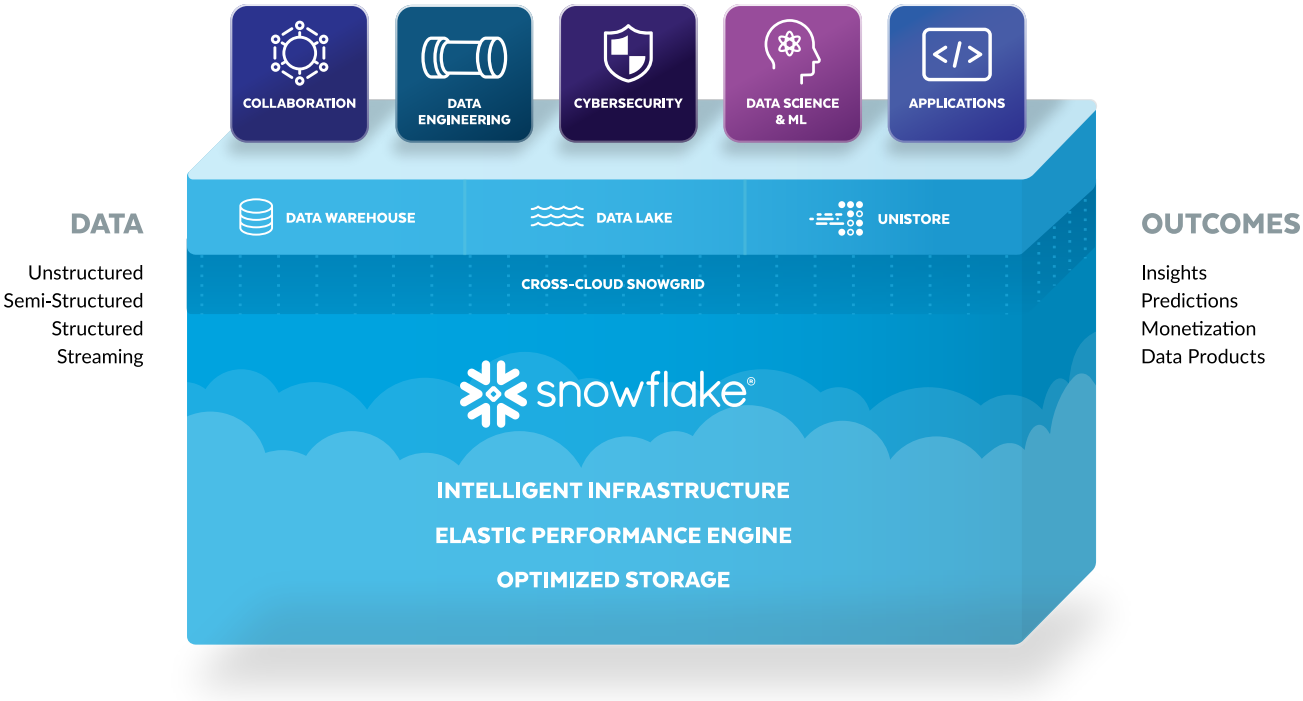

FIGURE 1: SNOWFLAKE AS A SINGLE PLATFORM FOR DIFFERENT TYPES OF DATA AND WORKLOADS

[2] In public preview at the time of publication, August 2022.
[3] In private preview at the time of publication, August 2022.
[4] In public preview at the time of publication, August 2022.

Snowflake is a distributed platform, not a monolith

Snowflake is a distributed but interconnected platform that avoids silos and enables distributed teams to exchange data in a governed and secure manner. How does that work? A company can create one or multiple Snowflake accounts that can reside in the same or in different cloud regions and platforms (Figure 2). Each account can be home to multiple 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 or even in separate accounts while still using the underlying Snowflake platform to share data assets with each other. Note that the Snowflake concept of a "database" is not merely a traditional relational database but also includes all other functional capabilities in Snowflake such as data engineering, data lake, data warehousing, data sharing, and data science. Using compute clusters to combine and process data from multiple databases or accounts is a core capability of the Snowflake platform.

Snowflake has built-in data sharing and marketplace capabilities

Data producers in Snowflake can share data, data services or applications with other accounts by publishing metadata ("listings"). Using listing discovery controls, producers can privately share with other accounts, group of accounts or publicly share via the Snowflake Marketplace. Data producers can specify SLAs or SLOs for the data that they are sharing, such as the update frequency, the amount of history, the temporal granularity of the data, and other properties that help describe the data as a product.

Other teams can search to discover relevant data assets available to them and obtain or request access. Such data consumers gain live access to the shared data, which remains under the control of the producer who can customize access policies or revoke access at any time. The access to shared data does not require an ETL or data movement process to be implemented by the producer or consumer. Producers can also publish and share external tables, which are "views" over files stored outside of Snowflake, and which can optionally include Delta Lake and Iceberg formats. Producers can even share data with third parties outside of the company even if those parties are not active Snowflake clients. For example, a data producer can share data externally via a so-called Snowflake reader account and the full breadth of supported APIs. Or, they can periodically export (partitioned) data to a cloud storage bucket using any of today's popular file formats.

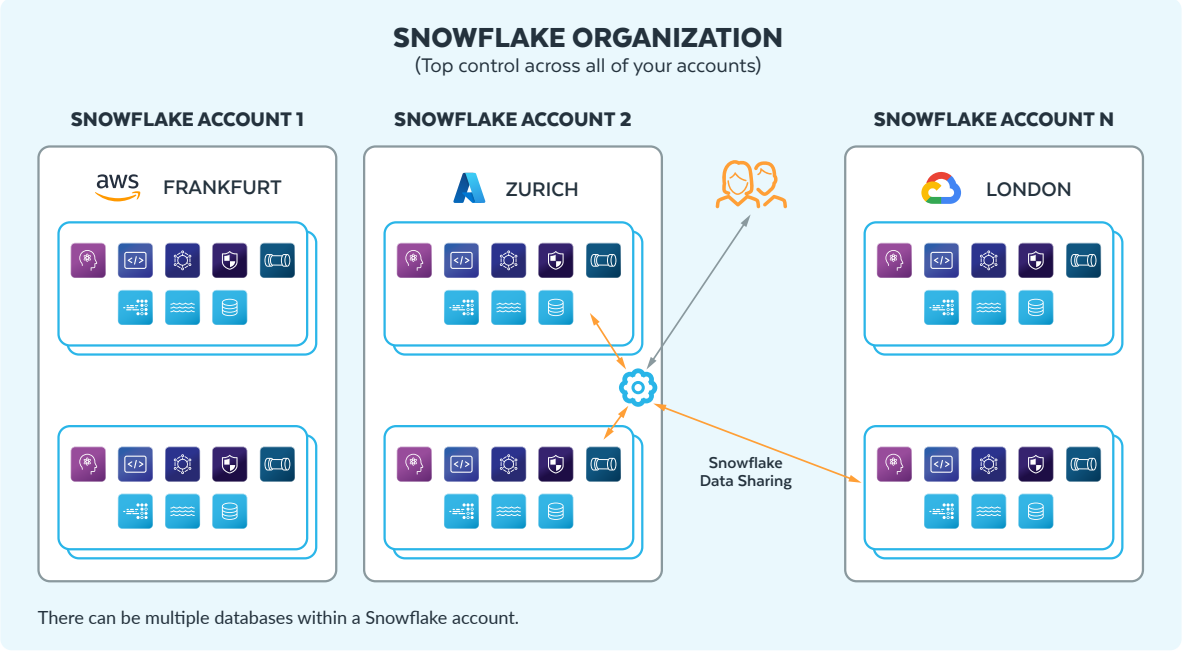

FIGURE 2: SNOWFLAKE ORGANIZATION, ACCOUNTS, AND DATABASES SUPPORT A DISTRIBUTED ARCHITECTURE

Snowflake offers a broad range of security and governance features

Federated governance is arguably one of the most challenging parts of a data mesh journey and often requires one or more tools in combination to satisfy all requirements. At the platform level, Snowflake supports role-based access control, row-level access policies, column-level data masking, external tokenization, as well as data lineage, audit capabilities, and more. Users can also assign one or more metadata tags (key-value pairs) to most any kind of object in Snowflake, such as accounts, databases, schemas, tables, columns, compute clusters, users, roles, tasks, shares, 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[5] enable users to associate an access restriction with a tag such that the access policy is automatically applied to any matching data object that carries the relevant tag.

In Snowflake, the definition of most governance controls such as tags, access policies, or masking rules can be defined separately from applying these controls to data objects. This enables domain owners to agree on common tags or policies across domains while leaving their enforcement or extension to each domain individually. Additionally, secure views and data clean room functionality can be used for analytics over sensitive data that could not be shared otherwise.

From a product usage perspective, metrics such as telemetry and consumption data are collected, which can be used for impact analysis. This enables domain teams to track how and how often their data products are being used by different consumers.

Snowflake provides a near self-service experience

Some of the most common reasons for our clients to choose Snowflake include ease of use and near-zero maintenance. These are critical properties for a self-service platform. For example, users can easily instantiate and scale their own compute clusters without support from an IT infrastructure team. Cloning dev and test environments is equally straightforward. A change data capture mechanism can be set up with a 1-line SQL DDL statement. This focus on ease of use has been a guiding principle for all features and functions on the Snowflake platform.

[5] In public preview at the time of publication, August 2022.

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. Treating data as a product requires most of all a product-oriented mindset that must become an organizational habit. Additionally, domains need suitable self-service tools that support the creation and management of data products. Let's outline how Snowflake can help you implement the concept of data as a product.

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

- **Data:** In Snowflake the data of a data product can have various forms, such as tables, views, files (JSON, XML, Parquet, Avro, CSV, etc.), or external tables that act as views over files outside of Snowflake. A single data product can consist of multiple such objects. A typical practice for domains is to use one schema per data product to group the data objects and optionally also the code for each data product. Data producers can model the data in whichever way is best suited to satisfy the needs of the data consumers.
- **Metadata:** The metadata of a data product includes the technical metadata of its data objects, such as tables names, column names, data types, or file format definitions. The metadata also includes object dependencies, data lineage, and access history. Each object can also be annotated with tags which are key-value pairs to express arbitrary metadata such as data origin, domain, sensitivity, business terms, taxonomy, cost center, or other user-defined attributes.

  When a data product is published in the Snowflake Marketplace, the data producer is prompted to provide documentation such as a product description, the business need, examples, terms of service, and a link to data product support. The producer is also prompted to specify data product SLOs such as the update frequency, the amount of history, the temporal granularity of the data, and other properties (see Figure 3).

- **Code:** The code of a data product includes the pipelines and transformations that create and refresh a data product. In Snowflake, such code can include Snowflake Tasks, pipes, Streams, Stored Procedures[6], user-defined functions, etc., all of which are Snowflake objects that can be grouped in a schema per data product. The code in these objects can be SQL, Java, Javascript, Scala, or Python and runs natively in the Snowflake platform.

  The code can also include policies. In Snowflake this can be code for role-based access control, dynamic data masking policies, row-level access control policies, secure views, object tagging, or code to classify or anonymize/tokenize the data.

- **Infrastructure dependencies:** For example, a Snowflake task that schedules and orchestrates the pipeline to refresh a data product can specify a certain compute cluster for the job. This can be a dedicated compute resource for just one data product or shared among multiple data products. Either way, the cluster can be suspended and resumed automatically as needed to incur cost only when it performs work. Also, clusters can be scaled up and down in a self-service manner. Tasks, pipes, and other operations can also be serverless to reduce or remove the need for explicit infrastructure dependencies.

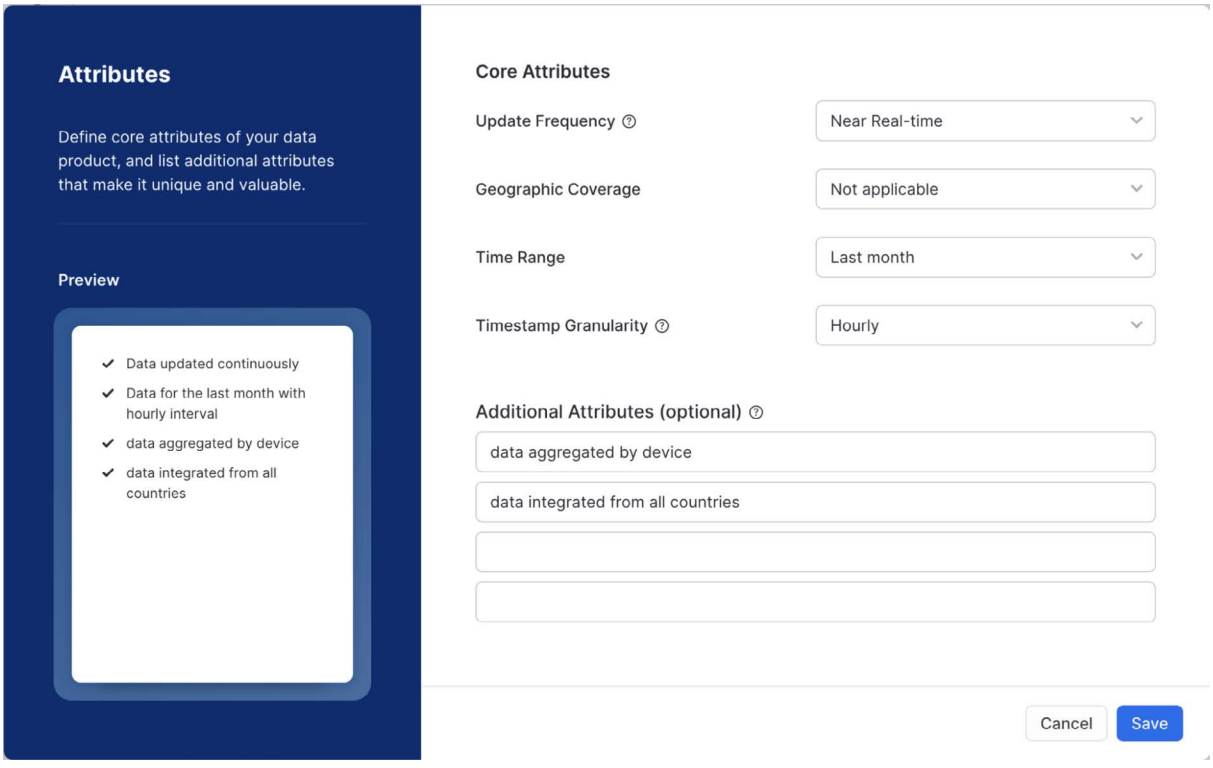

FIGURE 3: SPECIFYING DATA PRODUCT SLOS FOR A DATA PRODUCT LISTING

[6] In public preview at the time of publication, August 2022.

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 JDBC, ODBC, .NET, and APIs for many popular programming languages. Snowflake's collaboration capabilities can also be used to securely access and distribute data, data services and applications seamlessly across clouds without requiring additional ETL pipeline or integrations.

Data products should also exhibit a number of important properties. Table 1 lists some examples of Snowflake capabilities that can help you achieve these characteristics.

DATA PRODUCT CHARACTERISTICS	EXAMPLES OF SNOWFLAKE CAPABILITIES (NOT EXHAUSTIVE)
Secure	Role-Based Access Control, Row-Level Access Policies, Dynamic Data Masking, Encryption, Tokenization
Discoverable	Targeted Discovery/Snowflake Marketplace, Optional integration with 3rd-party catalog
Addressable	Snowflake data shares, standardized access cross-cloud and cross-region
Understandable	Custom metadata tags, data listings with documentation, statistical shape of the data in Snowsight
Trustworthy	SLOs/SLAs such as update frequency or granularity, data lineage, object dependencies, access history
Natively accessible	SQL, Python, Java, Scala, SQL APIs, REST API, Dataframes, etc., 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, Snowflake collaboration, data sharing, marketplace, data exchange
Valuable on its own	Composite data products that consist of multiple objects, Data products can consist of data objects plus functions that can be shared with data product consumers

TABLE 1: DATA PRODUCT CHARACTERISTICS SUPPORTED IN SNOWFLAKE

ARCHITECTURE OPTIONS FOR DISTRIBUTED DOMAINS

Now let's look at different Snowflake topologies that companies have chosen as a platform to support distributed domains. These topologies are general patterns and an actual implementation can vary based on specific requirements and preferences.

- **"Account per domain": Each domain uses a separate Snowflake account**
  - 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 consistent Snowflake experience and built-in data sharing capabilities between domains, based on a central data exchange of metadata where all domains can publish and obtain access to data products.
- **"Database per domain": Each domain uses one or more separate Snowflake databases**
  - All of these databases are managed in a single Snowflake account.
  - Simplified management of users, security, and governance across domains.
  - Access to data products can easily be provided by setting object level permissions across databases.
  - Each domain team can still spin up and scale their own computer clusters independent from other domains.
- **"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 isolated from other domains.
  - Potentially higher effort in naming conventions to distinguish between objects from different domains.
  - Can be useful for sub-domains in a domain/subdomain scenario.
  - In the remainder of this article we will not discuss the "Schema per domain" option in detail but it has many similarities to "Database per domain."
- **"Heterogeneous domains": Domains can use different IT stacks**
  - Some domains use Snowflake and some use other systems.
  - Some domains are in the cloud and some can potentially be on-premises.
  - Usually incurs a higher degree of complexity in order to accommodate heterogeneous domain environments.
  - Requires special consideration since it can be contrary to the data mesh goal of using a common domain-agnostic self-serve platform across all domains.

Further architecture variations or hybrid approaches derived from these base types are possible and plausible. For example, a company might choose "database per domain" and have these databases in several accounts instead of a single account. Also, some domains might be using separate databases while others use an entire Snowflake account. Also, the environment that a domain team uses often consists of Snowflake plus additional tools based on their requirements and skills.

The key point is that Snowflake supports multiple architecture choices that enable different trade-offs between domain autonomy and decentralization on the one hand versus different degrees of complexity and operational management on the other hand.

Each company needs to find the right balance between centralization and decentralization that works best for their size, legacy, and organizational culture. The same applies to federated governance where companies need to choose the right balance between centralized control and local domain autonomy that works best for them.

The following sections discuss these architecture options in more detail. The focus of that discussion lies mainly on the Snowflake topologies rather than the integration with third-party tools that our clients often use together with Snowflake for their data mesh initiatives.

Single Account: Database Per Domain

A popular topology that many of our data mesh clients are adopting uses a single Snowflake account in which domains use separate databases and separate compute clusters as their autonomous environments. Each domain can be assigned one or more databases and clusters for their development, test, and production needs. The self-service nature of the platform enables domains to use Snowflake's Zero-Copy Cloning capabilities to (re)create dev and test environments instantly and frequently. Additionally, different users within a domain can be allowed to spin up and scale their own compute clusters for their respective needs in a self-service manner. Still, cost and consumption monitors as well as quotas can be configured for domains or other levels of granularity in the user and resource hierarchy.

Having all domains use the Snowflake Data Cloud enables them to have their separate environments and compute resources without being physical silos that would make data product access challenging.

Some governance is decided centrally and is applied to all databases with a DevOps process. This can be facilitated by features such as object tags to keep an easy overview of the different objects owned by the domains. Within the domains governance is controlled by the domain teams that apply role-based access control as well as row- and column-level access policies to secure data and exclude users and domains from getting unwanted access to certain data.

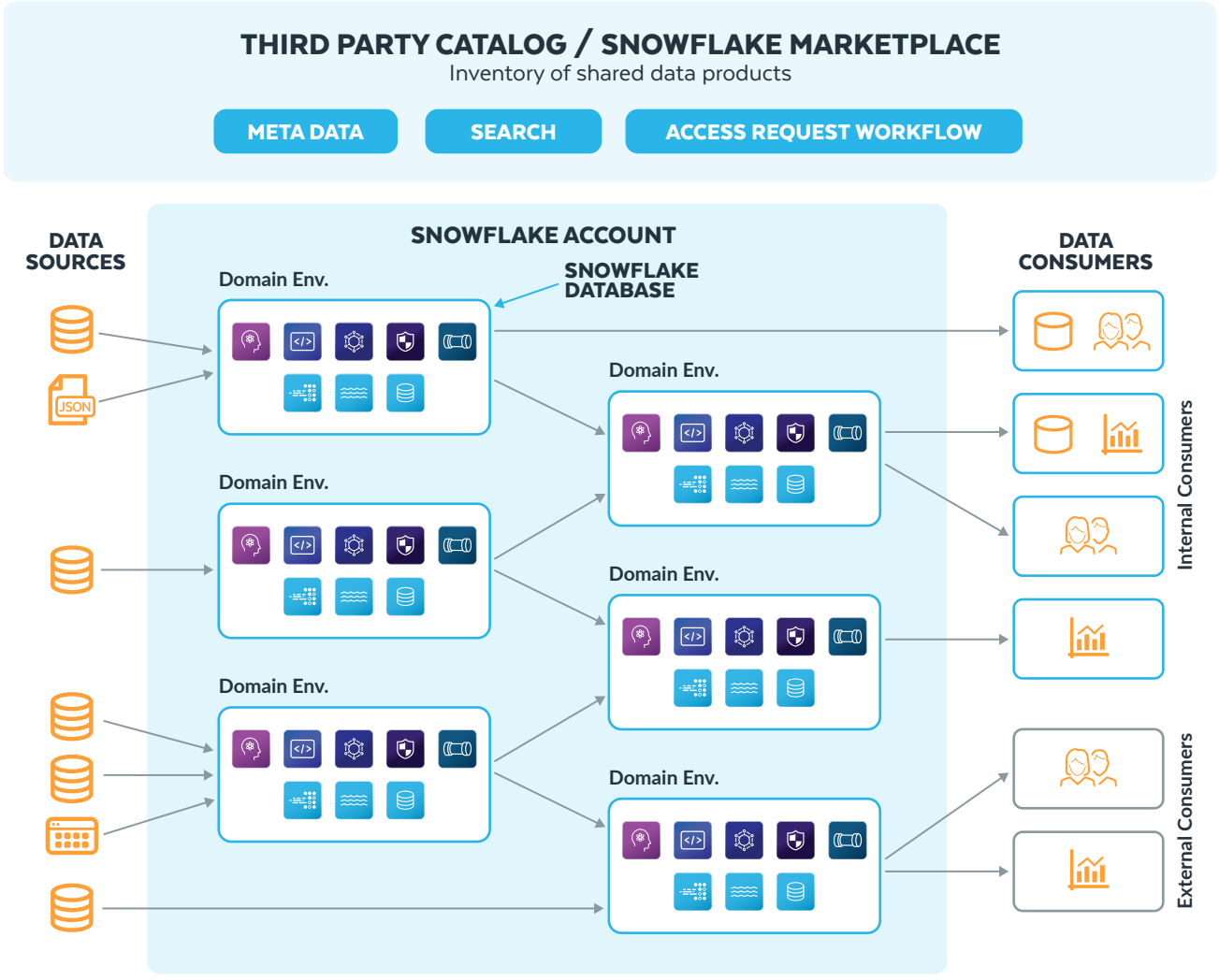

FIGURE 4: SINGLE SNOWFLAKE ACCOUNT - DATABASE PER DOMAIN

Every domain can have multiple schemas, with one serving as a layer to make products available to other domains. Another approach would be using a common share database where every domain will have a schema to publish its data products as views (no copies). These products can be structured, semi-structured, or unstructured data, depending on what the product comprises. The products are then listed in a third-party data catalog to be discoverable.

For requesting access to a product, we have seen multiple ways such as a manual approach, where the requestor needs to open a ticket, which then will be processed by the domain team and access allowed or denied by granting an access role to the requestor. Some catalogs provide a more automatic flow.

The advantages of having all the domains in one Snowflake account are:

- Access to data products can easily be done by setting intra-database permissions.
- Centralized network, security and governance policy administration simplifies the overall management.
- Disaster recovery is simpler as it only requires one other account in another region or cloud to support.

Naming conventions should be planned carefully, as there can be a lot of objects, considering that every domain might require DT(A)P (Development, Test, Acceptance, Production) environments, which they can easily create with Zero-Copy Cloning.

Similar to this approach is using a schema per domain. The implications of this approach are similar to the database per domain approach, as everything is organized logically within one Snowflake account. It should be noted, however, that having a database per domain is easier regarding sharing data with external consumers publicly via Snowflake Marketplace or privately using listing discovery controls.

Multiple Accounts: Account Per Domain

Another possible topology allows each domain to operate in a separate Snowflake account. These accounts can be in the same or in different cloud regions and cloud platforms. The global Snowflake Data Cloud enables companies and domains to share data across accounts, regions, and cloud platforms and easily obtain standardized access to each other's data products in a secure and governed fashion. Some of our clients are using this capability to support a multi-region, multi-cloud data mesh.

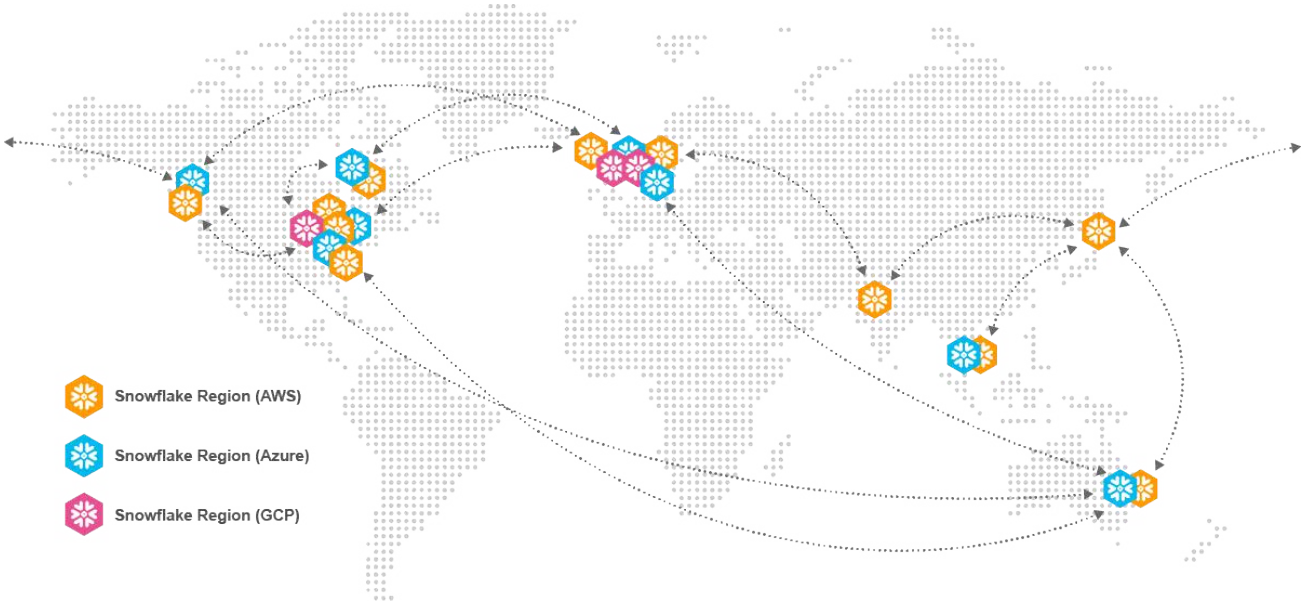

FIGURE 5: THE SNOWGRID AS A GLOBAL DATA CLOUD PLATFORM

There are various reasons why companies are choosing this topology. For example, a company may be operating in a globally distributed manner where different domains might naturally align with different locations and regions in the world. Some companies may be operating globally and need to observe data locality requirements (for example, an international company where some of the data is not allowed to leave Europe without anonymization, masking, or other measures to ensure compliance with data privacy regulations).

Another common reason is mergers and acquisitions that may force a company to exchange data across regions or cloud platforms. Some companies intentionally pursue a multi-cloud strategy for diversification or to accommodate preferences and existing investments that different business units have already made. Using separate accounts also achieves greater autonomy of the domains for example, if there is a requirement to have separate user management and security management for each domain).

The resulting topology (Figure 6) is logically very similar to using a separate database per domain, except that each domain now "owns" a separate Snowflake account and uses the Snowflake data sharing and Marketplace capabilities to make data products accessible to others.

The advantages compared to the database per domain approach can be the following:

- Data sharing and collaboration capabilities can be used across domains.
- Global naming standards can be applied more easily as each account is an independent namespace.
- Cloud platform and regional preferences can be supported.
- There is separate security and user management per account.

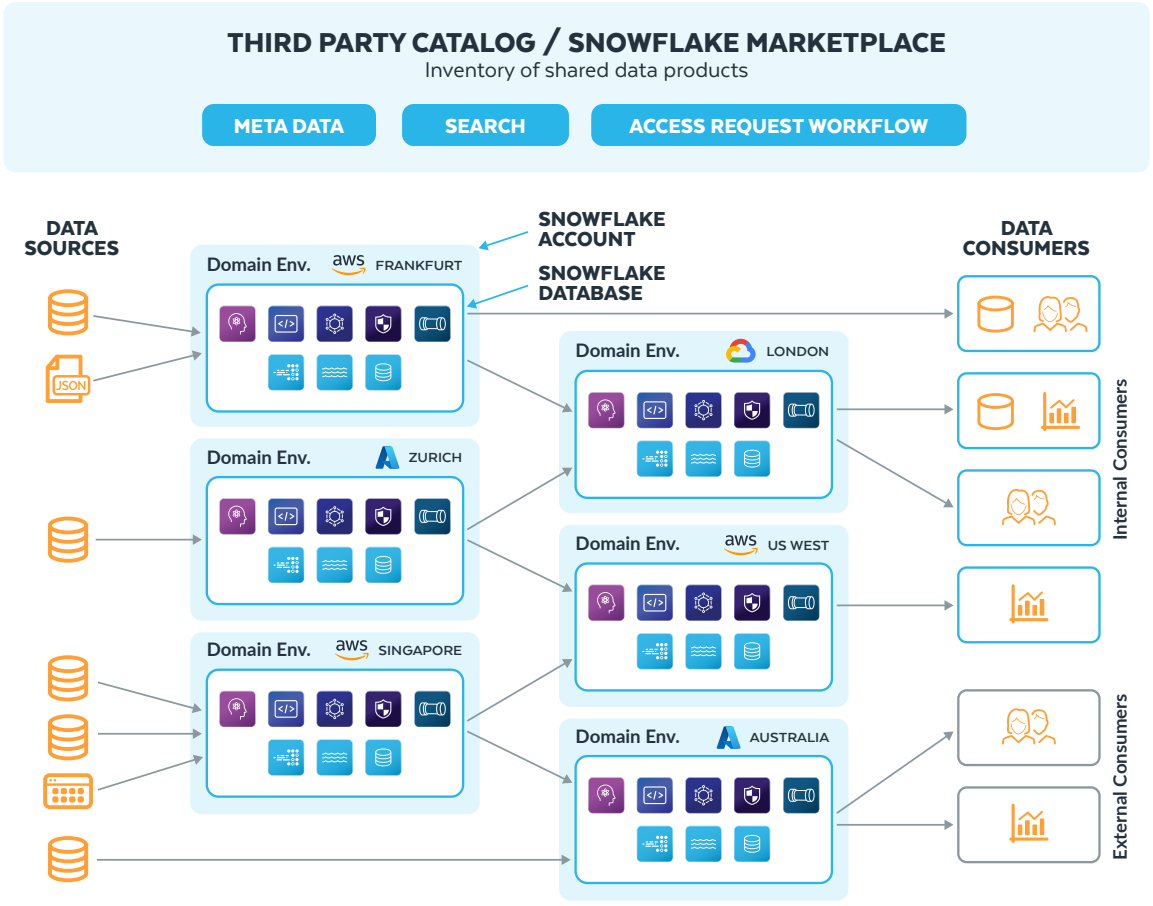

FIGURE 6: MULTIPLE ACCOUNTS - ONE ACCOUNT PER DOMAIN

Heterogeneous Architecture

Some clients have asked us how to integrate other non-Snowflake domain environments into the topologies discussed above. Such integrations result in a heterogeneous architecture where not all of the domains are using the same domain-agnostic data platform to implement their pipelines and data products. Often this is motivated by the desire to reuse multiple different repositories or technology stacks that already exist in different parts of the organization.

We found that such a heterogeneous architecture usually drives up the cost and complexity of a data mesh journey. The reason is that greater heterogeneity of the participating systems makes it harder to ensure consistency in governance, security, metadata, interoperability standards, performance, required skills, IT support, and other critical areas. Therefore we encourage clients to carefully consider the *role* of the diverse systems and repositories that they seek to integrate into a data mesh. Are those systems truly used as domain environments that build and serve data products? Or should they rather be viewed as data sources that domains take as input? The latter case often enables clients to fall back to the topologies discussed above.

One approach to integrating non-Snowflake domain implementations is for them to push data assets or "near-ready data products" to an intermediate layer for which Snowflake can act as a "proxy" that serves the data products with consistent governance, security, interoperability, etc., to the rest of the data mesh.

This intermediate layer could be, for example, Kafka topics that feed into Snowflake via continuous ingestion followed by automatic updates of data products in Snowflake. The intermediate layer could also be one or more cloud storage buckets in Amazon S3, Azure Blob Storage, Azure Data Lake Storage, or Google Cloud Storage. Data formats can include JSON, XML, Parquet, AVRO, Apache Iceberg, Delta Lake, and others. Snowflake can either auto-ingest new files from storage buckets continuously for best performance, security, and automatic management,

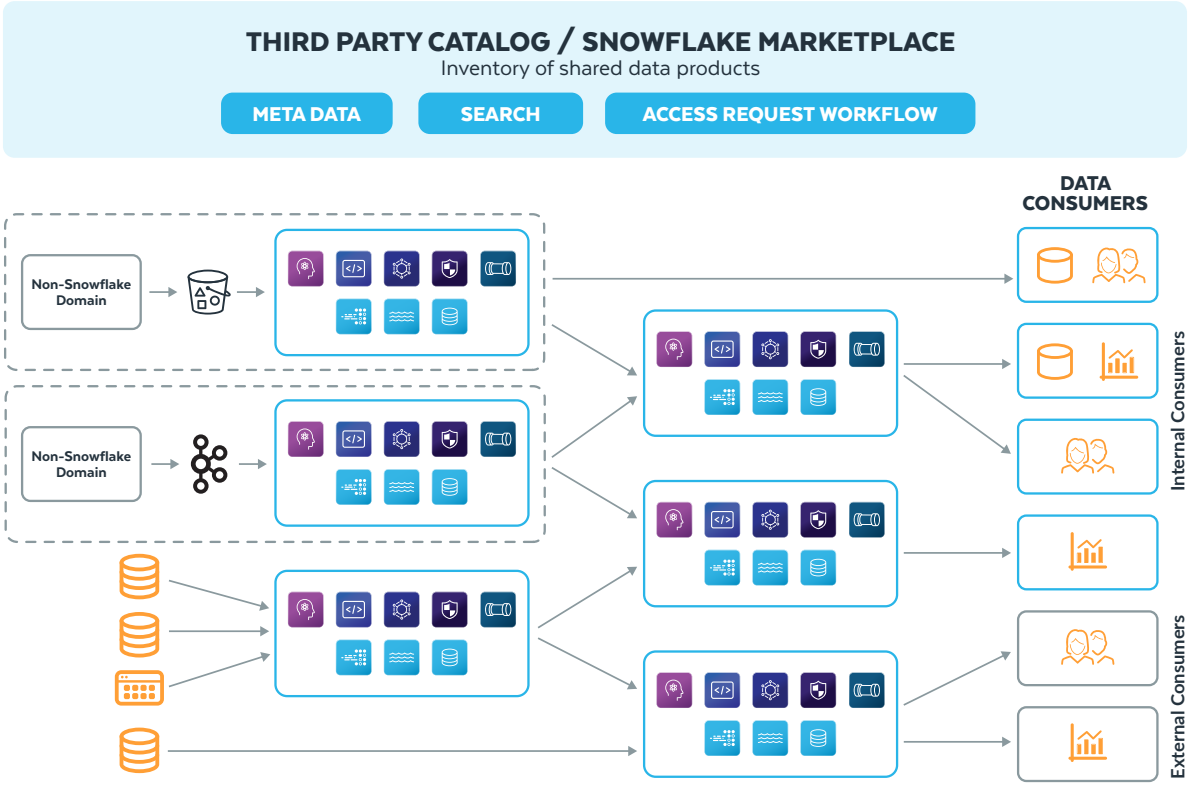

FIGURE 7: HETEROGENEOUS ARCHITECTURE

or expose read access to such files as External Tables to the rest of the data mesh. Snowflake's External Tables are essentially views over data that resides in files outside of Snowflake. Still, External Tables are first-class data objects in Snowflake that can be secured and governed, joined, and even shared via Snowflake collaboration much like other data objects in Snowflake. This enables Snowflake to act as an integration layer that can expose outside data in a consistent and governed manner without necessarily ingesting and duplicating the data.

Various other options for integrating non-Snowflake environments into the data platform exist but are beyond the scope of this document.

Some companies look at data virtualization as a potential solution to integrating a diverse set of domain environments. While there are certainly valid use cases for data virtualization, we found that it also creates a number of challenges. One challenge is performance when data from multiple different repositories needs to be joined. This typically requires data movement to bring data into one common place to compute the join, even if other predicates can be pushed down to the data sources.

This can prohibit virtualization for performance sensitive use cases. With Snowflake, a join across multiple data objects in a single database has approximately the same performance characteristics as when these data objects are in separate databases or even in separate Snowflake accounts, which is a non-trivial property of the Snowflake platform architecture. Another challenge that we have seen with virtualization in some companies is that it can encourage teams to continue to work in their respective technology islands, which are often very domain-specific, instead of striving towards a common and domain-agnostic, self-service platform.

SUMMARY

Data mesh is not a silver bullet for all data management and data integration challenges. But if you determine that data mesh is the right approach for your enterprise, make sure that you focus on the organizational and non-technical questions that are mandatory for success. Some examples include organizational changes, roles and responsibilities, staffing, incentives and accountability, buy-in from key stakeholders, or shifting the mindset to product thinking.

Eventually, you will need to design a self-service IT architecture that can support distributed domains and data products with federated governance. Snowflake can play that 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 collaboration setups. The underlying Snowflake platform with the global Snowgrid and the Snowflake Marketplace act as the connecting tissue that helps organizations avoid the risk of creating data silos.

Additionally, Snowflake provides a broad range of features that help companies implement the concepts of data as a product and federated governance. Snowflake also integrates easily with a broad range of third-party tools that can provide additional platform capabilities. The Snowflake Data Cloud is an excellent technology choice to complement the organizational changes and processes that are required for a successful data mesh transformation. To learn more about how Snowflake can help, visit **snowflake.com/data-mesh**

ABOUT SNOWFLAKE

Snowflake delivers the Data Cloud—a global network where thousands of organizations mobilize data with near-unlimited scale, concurrency, and performance. Inside the Data Cloud, organizations unite their siloed data, easily discover and securely share governed data, and execute diverse analytic workloads. Wherever data or users live, Snowflake delivers a single and seamless experience across multiple public clouds. Snowflake's platform is the engine that powers and provides access to the Data Cloud, creating a solution for data warehousing, data lakes, data engineering, data science, data application development, and data sharing. Join Snowflake customers, partners, and data providers already taking their businesses to new frontiers in the Data Cloud. **snowflake.com**