



E-BOOK

Powering Your Data Mesh with Snowflake & Immuta

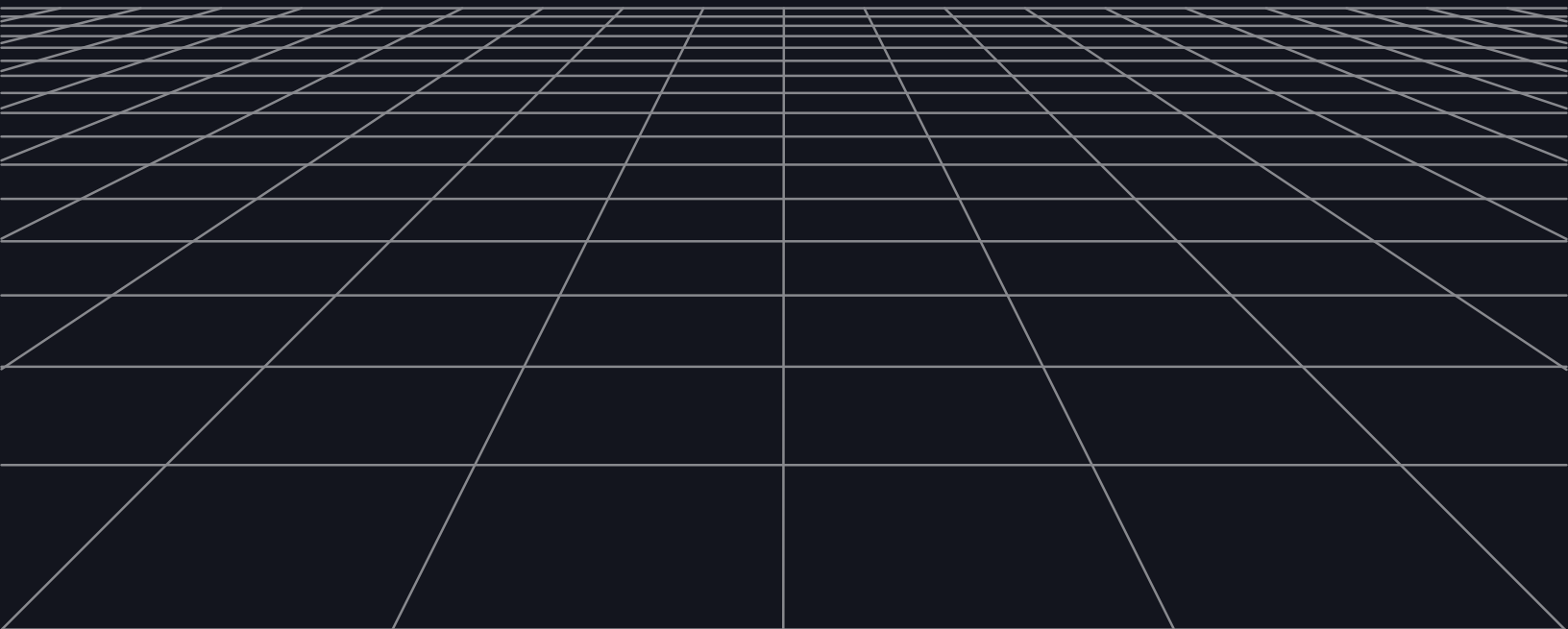


Table of Contents

Introduction	3
Centralization for Decentralization	4
Self-Service Data Platform: Snowflake	5
Federated Governance & Security: Immuta & Snowflake	7
Data-As-A-Product	11
Conclusion	21

Introduction

Data Mesh is an architectural and organizational paradigm shift for how companies work with and share data internally within their organization and/or externally with their business partners. A key component of this paradigm shift is treating data as a product—allowing vertical teams, “domains,” to build and share data products horizontally across your company. The benefits of this layered approach can be enormous; each vertical team builds relevant and valuable data products to be used and combined with other domains’ data products. It also allows decentralized creativity, flexibility, and utilization of data products while empowering centralized discovery and federated governance.

Data Mesh can be thought of like an “app store,” except the apps are not apps at all, they are data products. And for data products to be “installed” and “run,” you need discovery, storage, compute, data definitions and documentation and, of course, governance and security. As an example, from the context of Healthcare and Life Sciences industry, “app store” can be a collection of data topics such as patient demographics, claims, clinical and real-world evidence data that help rapidly uncover patient/population level health insights, enable better patient/member experience, accelerate drug development, and more. This paper discusses how combining Snowflake and Immuta builds the foundation to create your data mesh “app store.”

Centralization for Decentralization

Centralized standards, federated governance, and frameworks enable the scalability and flexibility of decentralization. To maximize the flexibility that decentralization provides, centralized standards, federated governance, and frameworks must be in place. Using our app store analogy again, you can build any app you want, but in order to monetize it, you must follow guidelines, publish to standards, and meet security controls before anyone can find or buy it. A decentralized Data Mesh is no different.

The key components are:

1. Domain-Centric Ownership

The vertical teams, or the domains, that own the creation of the data products. They act as product managers and data engineers, owning their own data product roadmaps, pipelines, and data transformations, as well as documentation.

2. Self-Service Data Platform

A distributed but interconnected set of compute, storage, tools and capabilities that avoids silos and enables distributed domain teams to build and exchange data products through ingestion, transformation, and provisioning of data.

3. Federated Governance and Security

Horizontal interoperability standards and policies, horizontal data governance policies, and vertical domain-specific governance policies. This is how the company can ensure data remains secure while still providing data product teams the freedom and power of decentralization.

4. Data-as-a-Product

The delivery of the data products. It must be easily discoverable, subscribable, and understandable through documentation. A platform allows domain teams to operate independently and easily share data products with each other.

As you can see, the first component of Data Mesh is an organizational change; the other three components also involve some organization change (people, process aspects) on top of the technology changes. In this eBook, we will focus on technical changes required for Data Mesh and suggest organizations to come up with a strong change management process for an effective roll out of Data Mesh architecture.

Self-Service Data Platform: Snowflake

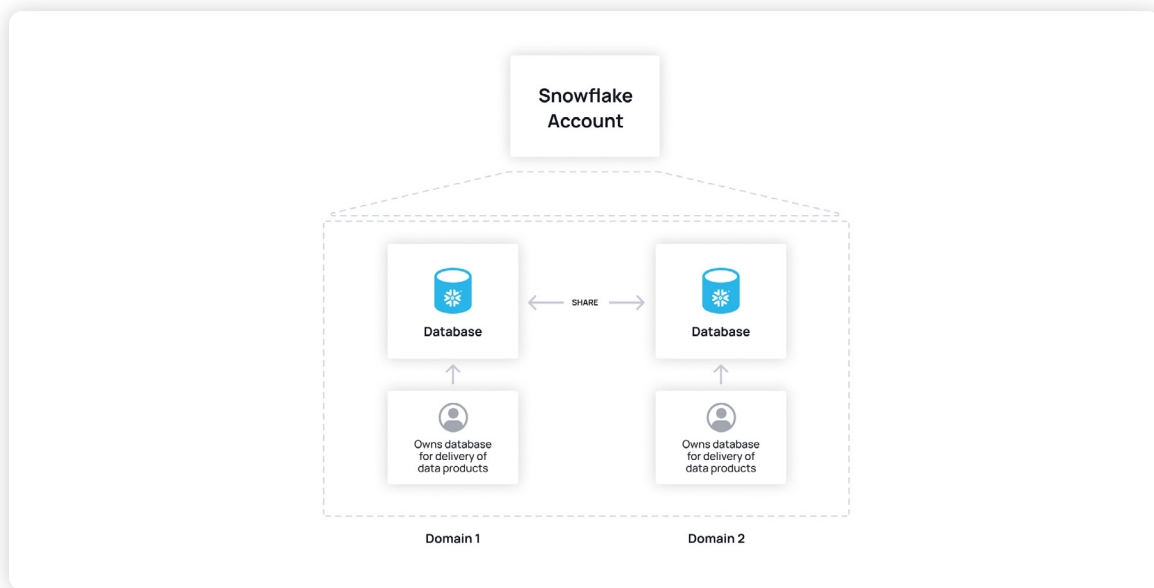
One of the key characteristics of a Data Mesh is that the data is natively accessible. This is critical to the exchange of data; otherwise, it is not interoperable or “joinable.” For example, if you were to create and share a data product without native accessibility, it would be impossible for the consumer of your data product to join it with other data products without introducing a data virtualization layer or creating an entire copy and transformation of your data to make it interoperable.

Snowflake solves this problem by acting as a distributed but interconnected platform that avoids silos and enables distributed teams to exchange data in a governed and secure manner. In other words, Snowflake empowers storage and computation of data in a decentralized manner, yet provides centralized guardrails to allow that data to be easily shared, governed, and consumed.

Here are two common Snowflake topologies that give teams self-service autonomy to create and share data products:

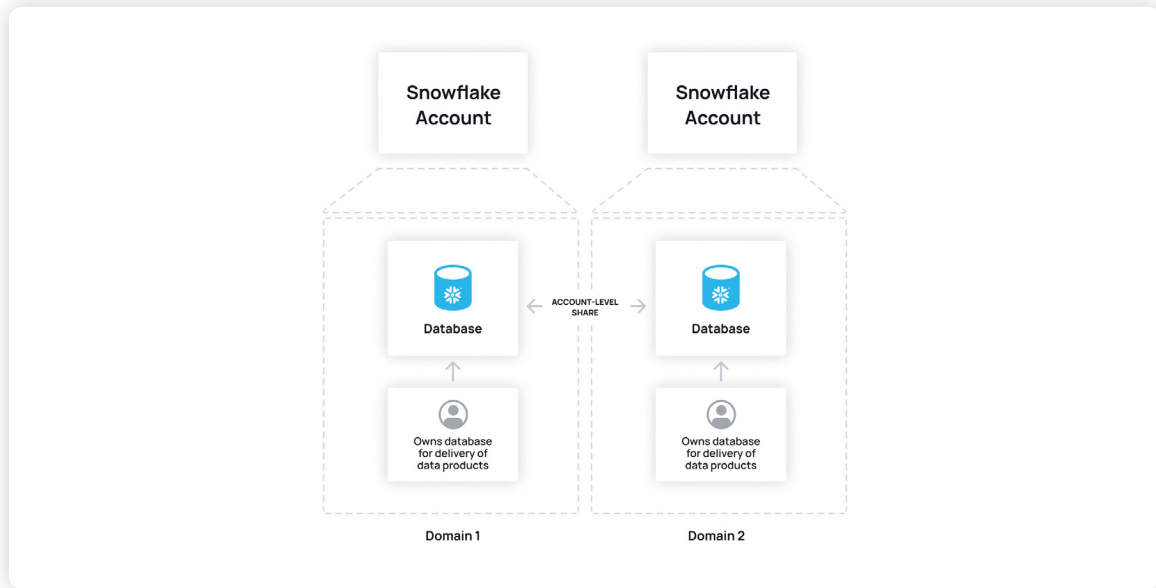
Option 1: Database per Domain

In this scenario, each domain owns a database in a single Snowflake account. This option is most applicable when all domains can be appropriately supported in a single cloud region on a single cloud platform. It also makes it easy to define granular per-user access control if that is required across consuming domains.



Option 2: Snowflake Account per Domain

This provides maximum isolation between domains and supports domains that operate in different cloud regions and cloud platforms, while having more control over global Snowflake configuration.



In either option, Snowflake's platform allows data product engineers to:

- Model and transform data per the requirements of their data product(s) using powerful Snowflake features such as Snowflake Tasks, Pipes, Streams, Stored Procedures, user-defined functions, and Snowpark.
- Code them in SQL, Java, Javascript, Scala, or Python and execute them natively in the Snowflake platform.
- Design data products that consist of various forms, such as tables, views, user-defined functions, or external tables (that act as views over files outside of Snowflake).
- Collate data products that consist of multiple such objects in a schema per data product, along with the code that manages the modeling/transformations; or, define such composite data products as "listings" that can be published, shared, and discovered across accounts via a private listing on Snowflake marketplace.

Federated Governance & Security: Immuta & Snowflake

Now that your data product teams have the autonomy of a self-service data platform, you must add access control and security. Complexity arises here because many data access control policies are not domain-specific and need to be enforced globally and consistently across domains, regardless of the data product. But this must be done in a way that also allows the data product teams to layer on additional domain-specific controls.

The following are key components of federated governance and security:

1. Ability to delegate and assign **policy ownership** to users globally or scoped more precisely to specific domain owners and their data products.
2. Ability to create a **common taxonomy** for how to describe and represent data through table and column tags and automatically tag said data.
3. Ability to **author policies specific to your domain of control** (global or domain-specific) without creating policy conflicts or disrupting existing policy.
4. Ability to **detect** and examine query activity against your data product(s).

Policy Ownership

It is critical that you are able to “scope power” of authoring policies to different levels of domain control. This allows the company to build “horizontal” policies that span all data products—policies that must be enforced no matter the data product based on the data they contain. Yet, you also need to delegate domain-specific policy authoring (“vertical policies”) to the data product domain owners so they can layer on domain-specific controls. As an example in the Healthcare and Life Sciences industry, classifying and tagging PHI and PII-sensitive data elements and ensuring the right governance policies are in place is critical for HIPAA compliance.

Immuta is able to scope policy authoring power in this manner through different permissions and data ownership. Horizontal policies that span all data products are managed through a permission in Immuta termed GOVERNOR. GOVERNOR is able to write consistent policies that span all data products, even as new data products arise, without needing any human intervention.

Vertical policies are handled through data ownership. As data products are registered with Immuta, the data product owners are automatically assigned ownership rights and can layer on additional domain-specific vertical policies.

Common Taxonomy

A common taxonomy is not only critical to search and discovery of data products, it is required for creating a semantic layer on top of your physical layer as the glue between policy and data. For example, it is much more powerful to build a horizontal policy that targets the semantic layer “mask all columns tagged PII” rather than authoring policy one at a time for every physical table “mask columns x, y, and z in table a”, “mask columns a, b, and c in table z”, etc.

When policies are built at the semantic layer, they apply to all relevant tables across your entire Data Mesh with a single policy. It also guarantees they are future-proof, because any time a new tag is attached to a data product (PII in our example), the horizontal policy targeting that tag will automatically attach to the column(s)/table(s) without human intervention.

This is only possible if all data products are tagged using a single, common taxonomy. Two approaches can be used:

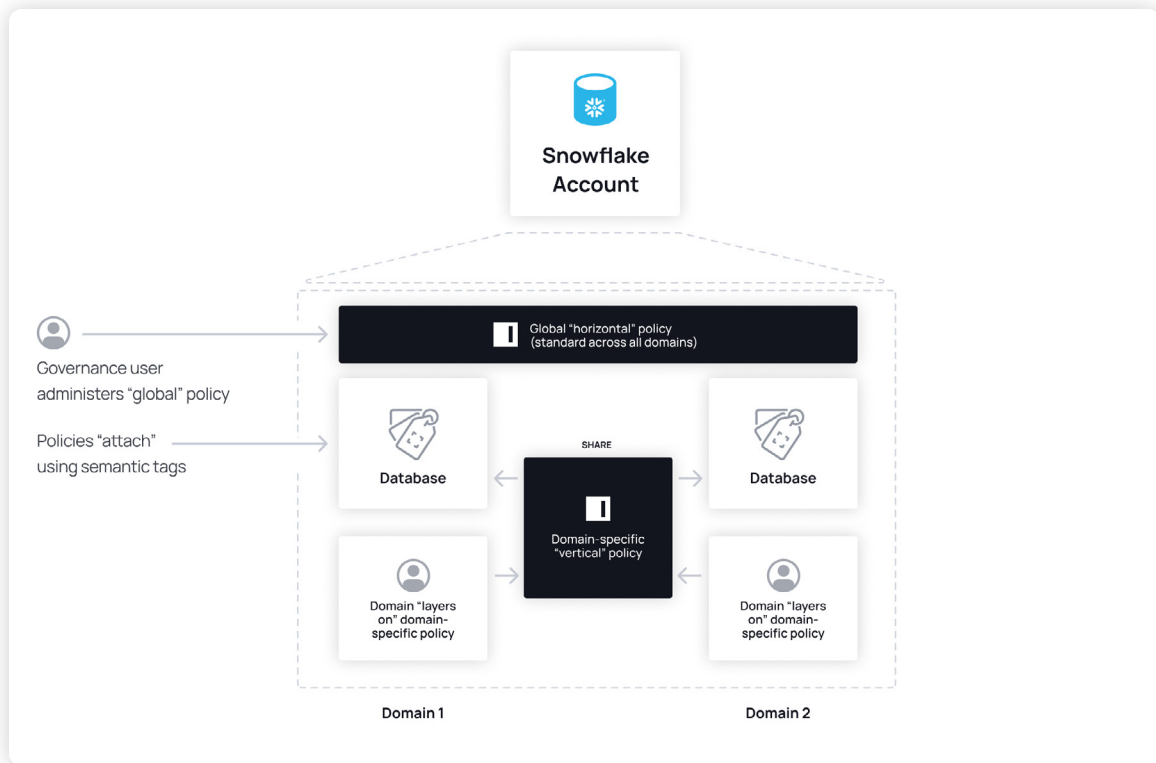
1. Trust the data product owners to manually tag the tables and columns appropriately when registering data products.
2. Automatically tag the columns by inspecting their content through Immuta’s sensitive data discovery capability. You can use Immuta’s standard set of over 100 entity classifiers or customize the classifiers to discover company-specific entities. This means as soon as a new data product is created and registered, it will be automatically scanned and tagged, and based on those tags, horizontal (or vertical) policies will be applied. Immuta’s auto-tagging can be combined with Snowflake classifiers/tagging as well as other catalog vendors—Immuta can act as a central authority for data tags across the organization. Note that Immuta does not need to extract any data outside of your Snowflake account to discover the entities; the discovery engine processes completely within your Snowflake account(s).

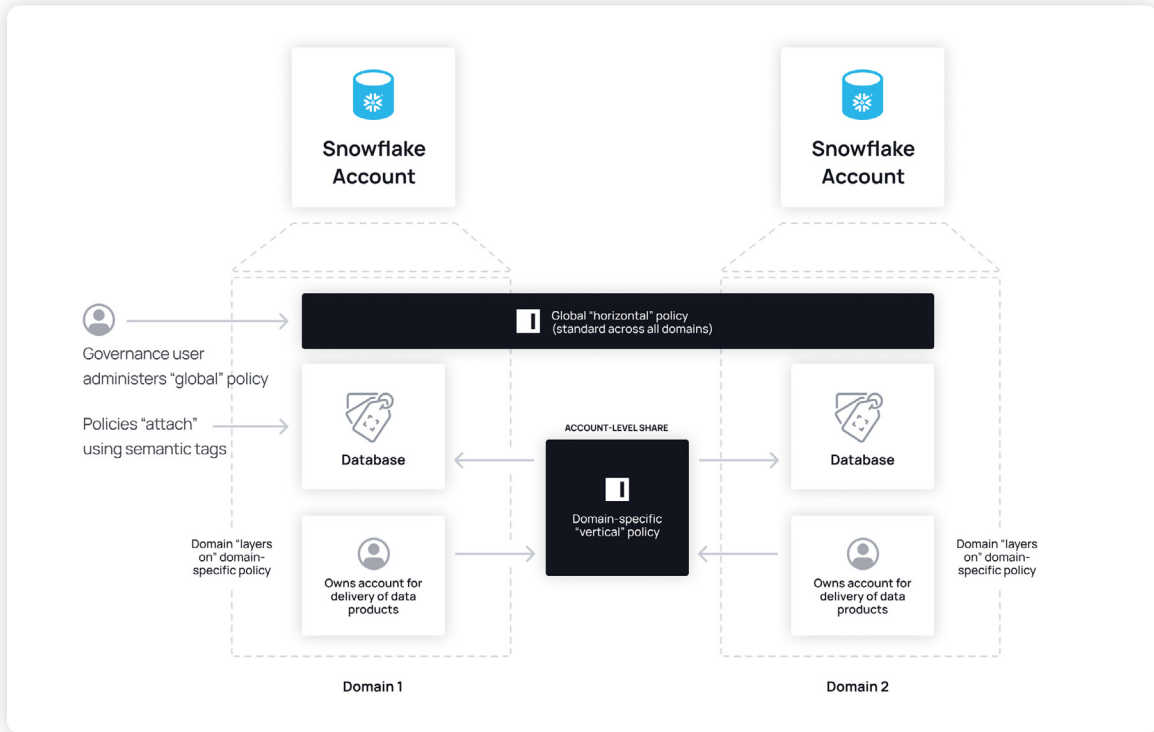
Consistent tagging is especially challenging to accomplish without Immuta if using the “account per domain” Data Mesh approach discussed above because you have no (straightforward) way to distribute a common tagging taxonomy across Snowflake accounts. Immuta is that centralized tag store, and it pushes the policies down into each account based on the discovered tags across accounts.

Author Policies Specific to Your Domain(s) of Control

We've discussed how Immuta can allow delegation of policy ownership to specific domains as well as use a semantic taxonomy (tags) to author policy. There are six more capabilities that make authoring policy with Immuta powerful for Data Mesh use cases:

1. Table-, row-, column-, and cell-level policies can all be authored against the semantic tag layer.
2. Policy conflicts—where two or more policies hit the same column or table—are automatically resolved/merged and all policies enforced as expected without any human intervention.
3. Policies authored in Immuta do not need to reference users explicitly, only user metadata. This makes access policies more resilient to organizational changes and can impact users broadly, as opposed to individually. These policies can also support just-in-time manual approvals, if desired.
4. Immuta supports DevOps workflows for policy management. Declarative files that represent policy state can be source controlled, PR'd, and pushed to Immuta to represent that policy state.
5. Policies are represented in plain language and easily understood by legal and compliance teams, allowing you to easily prove your Data Mesh is compliant.
6. All policies authored by Immuta are enforced invisibly in Snowflake using native Snowflake governance controls, allowing users to continue querying Snowflake directly.





Detect

All actions in Immuta are audited to include queries run by users against the data products. This allows both users with the GOVERNANCE permission and individual data product owners to understand how the data products are being used and how frequently.

Outcome		Records	Native Query Audit		Records per Page		
<input type="checkbox"/> Success	24	24 Results	Timestamp	Outcome	Context	Record Type	User
<input type="checkbox"/> Project			09 Nov 2022 10:38:21 -0500	Success	Data Source: Pov Data Immuta Fake Credit Card Transactions	Native Query	steve@immuta.com
<input type="checkbox"/> Record Type			09 Nov 2022 10:37:20 -0500	Success	Data Source: Pov Data Immuta Fake Credit Card Transactions	Native Query	steve@immuta.com
<input type="checkbox"/> Global Policy Applied	277		07 Nov 2022 11:40:55 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input type="checkbox"/> Global Policy Removed	101		07 Nov 2022 11:40:44 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input type="checkbox"/> Global Policy Update	80		07 Nov 2022 11:40:37 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input type="checkbox"/> Access User	50		07 Nov 2022 11:39:35 -0500	Success	Data Source: Trades2	Native Query	mvogt@immuta.com
<input type="checkbox"/> Data Source Subscription	39		07 Nov 2022 11:39:27 -0500	Success	Data Source: Trades2	Native Query	mvogt@immuta.com
<input type="checkbox"/> Global Policy Create	38		07 Nov 2022 11:39:04 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input type="checkbox"/> Global Policy Delete	37		07 Nov 2022 11:36:07 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input type="checkbox"/> Authenticate	32		07 Nov 2022 11:34:38 -0500	Success	Data Source: Trades2	Native Query	mvogt@immuta.com
<input type="checkbox"/> Tag Added	29		07 Nov 2022 11:34:30 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input type="checkbox"/> Data Source Delete	24		07 Nov 2022 11:31:48 -0500	Success	Data Source: Trades2	Native Query	steve@immuta.com
<input checked="" type="checkbox"/> Native Query	24						
<input type="checkbox"/> Switch Current Project	20						
<input type="checkbox"/> Api Key	17						

Additionally, reporting to prove compliance is possible, allowing teams to understand who has access to what data and why.

Data-As-A-Product

The final critical component of Data Mesh is data as a product. Just like an app store, data products must be easily discoverable, subscribable, and understandable and Immuta provides a data portal user interface to support this. Key components of this data portal are that:

- Data products are searchable by tag, name, or documentation content
- Data product owners can fully document the data products to include documenting individual columns
- Data products can be hidden from consumers based on policy
- Consumers that don't meet policy to gain access to a data product can be allowed access through manual just-in-time approvals (if prescribed in the policy)

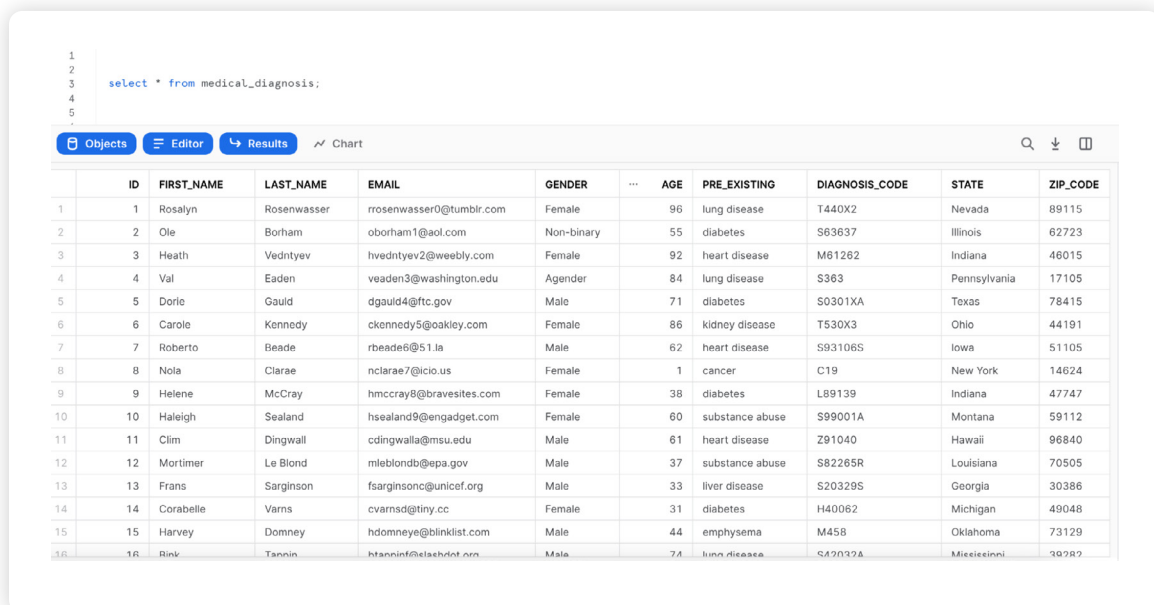
This is demonstrated in the next section.

See It In Action

We will now walk through a concrete example to demonstrate the power of Immuta and Snowflake for Data Mesh.

First, we have a user, Matt, who has created a new data product that merges patients' preexisting conditions with new diagnosis information. He has placed this data product in a schema under a database owned by his domain (his domain is "Medical Information").

The data looks like this:



```
1
2
3 select * from medical_diagnosis;
4
5
6
```

	ID	FIRST_NAME	LAST_NAME	EMAIL	GENDER	...	AGE	PRE_EXISTING	DIAGNOSIS_CODE	STATE	ZIP_CODE
1	1	Rosalyn	Rosenwasser	rosenwasser0@tumblr.com	Female		96	lung disease	T440X2	Nevada	89115
2	2	Ole	Borham	oborham1@aol.com	Non-binary		55	diabetes	S83637	Illinois	62723
3	3	Heath	Vedintyev	hvedintyev2@weebly.com	Female		92	heart disease	M61262	Indiana	46015
4	4	Val	Eaden	veaden3@washington.edu	Agender		84	lung disease	S363	Pennsylvania	17105
5	5	Dorie	Gauld	dgauld4@ftc.gov	Male		71	diabetes	S0301XA	Texas	78415
6	6	Carole	Kennedy	ckennedy5@oakley.com	Female		86	kidney disease	T530X3	Ohio	44191
7	7	Roberto	Bead	rbeade6@51.ia	Male		62	heart disease	S93106S	Iowa	51105
8	8	Nola	Clarae	nclarae7@cio.us	Female		1	cancer	C19	New York	14624
9	9	Helene	McCray	hmccray8@bravesites.com	Female		38	diabetes	L69139	Indiana	47747
10	10	Haleigh	Sealand	hsealand9@engadget.com	Female		60	substance abuse	S99001A	Montana	59112
11	11	Clim	Dingwall	cdingwall@msu.edu	Male		61	heart disease	Z91040	Hawaii	96840
12	12	Mortimer	Le Blond	mleblond@epa.gov	Male		37	substance abuse	S82265R	Louisiana	70505
13	13	Frans	Sarginson	fsarginsonc@unicef.org	Male		33	liver disease	S20329S	Georgia	30386
14	14	Corabelle	Varns	cvarnsd@tiny.cc	Female		31	diabetes	H40062	Michigan	49048
15	15	Harvey	Domney	hdomney@blinklist.com	Male		44	emphysema	M458	Oklahoma	73129
16	16	Rink	Tannin	htanninf@seahub.net	Male		74	lung disease	S42032A	Mississippi	39282

Now Matt will register that table with Immuta in order to share it on the Immuta data portal. As soon as that table is registered, Matt notices policies are already applied.

Subscribe to PHI

Users with Specific Groups/Attributes

Global Moderately Restricted Locked Discoverable

Users with the specified groups/attributes will be able to subscribe to the data source.

Allow users to subscribe to the data source when user possesses attribute Domain with value Medical Information

Otherwise

Allow users to subscribe when approved by
(anyone with permission Owner (of this data source))

This is because global policies were created and applied to the table earlier by a user with GOVERNANCE permission. Let's break this global policy down by looking at how it was originally configured.

This first part defines who is able to subscribe. In this case, it's any user with the attribute **Domain: Medical Information**.

What is the name of this policy? ⓘ

Subscribe to PHI

How should this policy grant access? ⓘ

Allow users with specific groups/attributes ⚙


Who should be allowed to subscribe? ⓘ

Create using plain language Create using Advanced DSL ⓘ

Allow users to subscribe when user possesses attribute ⚙ in Domain Medical Information

+ Add Another Condition

Next, the policy defines how the table is treated in the data portal:

Allow Data Source Discovery 



Users can still see that this data source exists in Immuta, even if they do not have these attributes and groups.

Require Manual Subscription

Users will not be automatically subscribed to the data source. They must manually subscribe to gain access.

Request Approval to Access


Users can request approval to the data source, even if they do not have these attributes and groups. This will require an approver with permissions to be set.

Approver is Anyone  with permission Owner (of the data source) 

[+ Add Another Approver](#)

The first box means that the listing will appear to users in the data portal even if they don't meet the policy. It is grayed out because of the bottom box, which specifies that a data owner can override the policy through a manual just-in-time approval; for that to work, the table must be discoverable in the portal. Lastly, the middle box means that to gain access to the table, users must actually discover and subscribe to it in the data portal UI rather than being given immediate access if they meet the policy.

The next section of the data policy defines where the policy should be applied and how conflicts should be handled:

Should this policy always be required or share responsibility? 

Always Required

Users must meet this policy's conditions to access governed data sources, regardless of other policies' conditions.

Share Responsibility

Users need to meet the conditions set in this policy or another Share Responsibility policy if more than one Share Responsibility policy is applied to the same governed data source(s).

Selecting the first option means they would require this policy to always remain considered, no matter what additional policies are layered on top (effectively, a boolean AND). By selecting the second option, they indicate that other users' policies can be met in lieu of theirs (effectively, a boolean OR). In this case, they are okay sharing responsibility.

Lastly, the user building the global policy will define where the policy should be applied, now or in the future:

Where should this policy be applied? ?

On data sources ⇅ with columns tagged ⇅ Discovered > PHI

[+ Add Another Circumstance](#)

As you can see, the policy is being applied using the semantic tag layer, specifically, the Discovered > PHI tag. Remember, the policy author is allowing users with **Domain: Medical Information** to subscribe, so it would make sense to target the PHI tag. This is also why the policy was immediately applied when Matt registered the table with Immuta, with his data product containing PHI that Immuta automatically discovered.

You can see the data dictionary and discovered tags here in the data portal (also notice that the pre_existing and diagnosis_code columns are documented):

Name	Type	Description
id	numeric(38,0)	No description provided.
first_name	text	No description provided.
last_name	text	No description provided.
email	text	No description provided.
		Discovered...Electronic Mail Address × Discovered > Identifier Direct × Discovered > PHI ×
gender	text	No description provided.
age	numeric(38,0)	No description provided.
pre_existing	text	any preexisting conditions via our historical lookup table
diagnosis_code	text	from the diagnosis table
		Discovered...ICD10 Code × Discovered > Identifier Indirect × Discovered > PHI ×
state	text	No description provided.
		Discovered...US × Discovered...Location × Discovered...State × Discovered > Identifier Indirect × Discovered > PHI ×
zip_code	text	No description provided.
		Discovered...Postal Code × Discovered > Identifier Indirect × Discovered > PHI ×

As mentioned, there was also a second policy on the table that appeared as soon as Matt registered it. Instead of a subscription policy, this is a data policy that masks sensitive PHI data and, again, was created globally. This is what Matt sees on his table in the portal:

Mask PHI Steve Touw Global

- Mask using a constant REDACTED the value in the column(s) `diagnosis_code` for everyone except when user possesses attribute Domain `Medical Information`
- Mask using a constant REDACTED the value in the column(s) `email` for everyone except when user possesses attribute Domain `Medical Information`
- Mask using a constant REDACTED the value in the column(s) `state` for everyone except when user possesses attribute Domain `Medical Information`
- Mask using a constant REDACTED the value in the column(s) `zip_code` for everyone except when user possesses attribute Domain `Medical Information`

While it may appear that the policy references actual physical columns, it does not. Let's look at the exact policy definition created by the user with GOVERNANCE permission.

What is the name of this policy? ?

Mask PHI

How should this policy protect the data? ?

Mask columns tagged Discovered > PHI add another tag (optional)

using a constant REDACTED for

everyone except when user possesses attribute in Domain Medical Information

[+ Add Another Condition](#)

Enter Rationale for Policy (Optional)

This part of the policy defines what is being masked. As you can see, it's self-explanatory and written in plain language.

The last part, just like the previous subscription policy, defines where the policy should be applied and targets data with columns tagged PHI, which again will end up targeting Matt's table. This policy was applied to Matt's table as soon as it was registered because the PHI tags were automatically discovered. When Matt sees the policy in the data portal (figure above), it references the physical columns it was applied to based on the semantic layer.

Where should this policy be applied? ⓘ

On data sources ⌵ with columns tagged ⌵ Discovered > PHI

[+ Add Another Circumstance](#)

As we know, Matt is part of **Domain: Medical Information**, so he is exempt from the masking policy. But if a user that was not in that domain subscribed to this table and queried it directly in Snowflake, they would see REDACTED PHI because Immuta administered that native Snowflake masking policy on the table. And, because the global policy was applied at data product registration time, this all worked without Matt having to understand every organizational policy.

Now, Matt wishes to create his own domain-specific policy to automatically allow users in group **Physicians** access to his data product. He can't edit the global policy because that would impact all data products, not just his (nor does he have the permission to, anyway). Instead, he can build a completely separate policy using Immuta which will only target the data products he's registered, and Immuta will completely handle policy merging and deconfliction.

Matt goes through the exact same policy authoring workflow that the user with GOVERNANCE permission did earlier. Step 1 is specifying who should have access, which is group **Physicians**:

What is the name of this policy? ⓘ

Share with physicians

How should this policy grant access? ⓘ

Allow users with specific groups/attributes ⌵

Who should be allowed to subscribe? ⓘ

Create using plain language Create using Advanced DSL ⓘ

Allow users to subscribe when user is a member of group ⌵ Physicians

[+ Add Another Condition](#)

Next, Matt must choose how he wants this policy treated in the data portal. In this case, Matt selects the first two options but does not specify that he wants just-in-time overrides of the policy through manual approval:

- Allow Data Source Discovery**
Users can still see that this data source exists in Immuta, even if they do not have these attributes and groups.
- Require Manual Subscription**
Users will not be automatically subscribed to the data source. They must manually subscribe to gain access.
- Request Approval to Access**
Users can request approval to the data source, even if they do not have these attributes and groups. This will require an approver with permissions to be set.

In terms of deconfliction/merging logic, Matt is also fine sharing responsibility:

Should this policy always be required or share responsibility? ⓘ

- Always Required**
Users must meet this policy's conditions to access governed data sources, regardless of other policies' conditions.
- Share Responsibility**
Users need to meet the conditions set in this policy or another Share Responsibility policy if more than one Share Responsibility policy is applied to the same governed data source(s).

Last, when deciding where the policy should apply, Matt does not need to only target his data products. Immuta understands that Matt does not have GOVERNANCE permission, so it automatically scopes the policy to only tables Matt (mvogt@immuta.com) owns. That way, Matt can focus the logic of where to apply the policy strictly to the semantic tags; in this case, PHI.

Where should this policy be applied? ⓘ

On data sources ⌵ with columns tagged ⌵ Discovered > PHI

+ Add Another Circumstance

Whose Data Sources should this policy be restricted to? ⓘ

Restricted to data sources owned by users add another user

mvogt@immuta.com (Immuta) x

or groups add another group

After saving this policy, let's look at it in the data portal to see how it was merged. Remember, before Matt created the subscription policy above, the original global subscription policy that automatically attached to his data product looked like this:

Subscribe to PHI

Users with Specific Groups/Attributes

Global Moderately Restricted Locked Discoverable

Users with the specified groups/attributes will be able to subscribe to the data source.

Allow users to subscribe to the data source when user possesses attribute Domain with value Medical Information

Otherwise

Allow users to subscribe when approved by
(anyone with permission Owner (of this data source))

After Matt activates the policy, he just defined and it attaches to his data product; the merged subscription policy looks like this:

Multiple Global Subscription Policies

Users with Specific Groups/Attributes

Global Moderately Restricted Locked Discoverable

Users with the specified groups/attributes will be able to subscribe to the data source.

The following Global Policies are being applied to this data source:

- Subscribe to PHI
- Share with physicians

Allow users to subscribe to the data source when user (@hasAttribute('Domain', 'Medical Information')) OR (@isInGroups('Physicians'))

Otherwise

Allow users to subscribe when approved by
(anyone with permission Owner (of this data source))

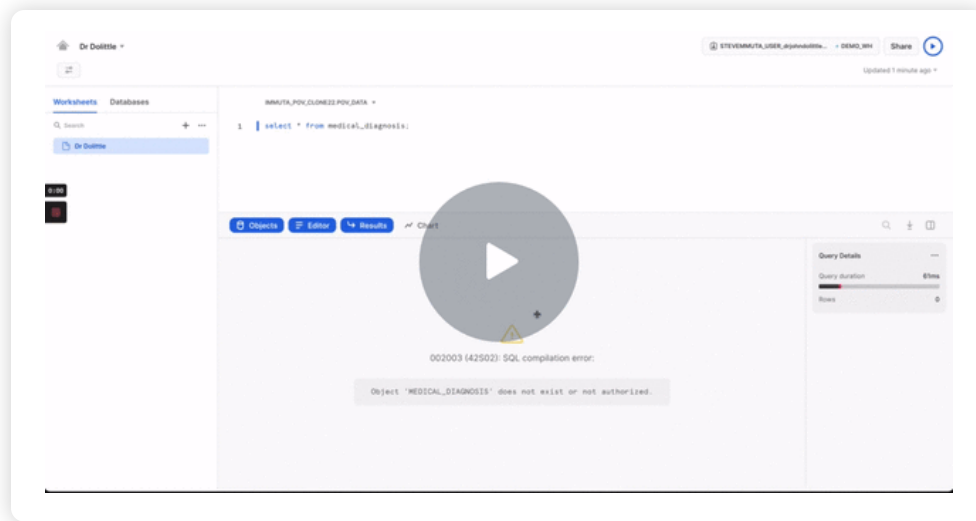
As you can see, if a user has **Domain: Medical Information** OR is in group **Physicians**, they will be able to subscribe. If not, they will need a manual approval from Matt (the data owner) to subscribe, per the original global policy. This is a great example of federated governance where organizational global policies can be created independent of domain-specific policies, yet work harmoniously together without manual collaboration between parties.

To bring this example full circle, let's now look at how data product consumers discover and subscribe to this data product in the data portal.

To do that, we'll use two different users:

1. Dr. John Dolittle – This user falls under group **Physicians** and should gain access immediately, without human intervention, when requested. Remember, even though Dr. Dolittle is in group **Physicians**, he is not subscribed to the data product until he actually requests access because that's how the policy was defined (“Require manual subscription”).
2. Juan Dixon – This user has neither **Physicians** nor **Domain: Medical Information**, so he needs manual approval from Matt when requesting access.

Here's a quick four-minute video of those two users walking through this scenario in the data portal.



Watch Video

Lastly, Matt is able to monitor and investigate queries against his data product:

Audit Record bea925d5-7f10-4f46-bca4f52f729a93d

```
{
  "id": "bea925d5-7f10-4f46-bca4f52f729a93d",
  "dateTime": "1669149108453",
  "month": 1474,
  "profileId": 5,
  "userId": "5",
  "dataSourceId": 16,
  "dataSourceName": "Medical Diagnosis",
  "count": 1,
  "recordType": "nativeQuery",
  "success": true,
  "component": "nativeSql",
  "accessType": "query",
  "query": "select * from medical_diagnosis;",
  "queryId": "01a87c8f-0406-1063-0000-18152e3185d6",
  "extra": {
    "handler": "Snowflake",
    "startTime": "2022-11-22 20:31:48.453000000 +0000",
    "endTime": "2022-11-22 20:31:50.431000000 +0000",
    "duration": "1978",
    "nativeObject": "IMMUTA_POV_CLONE22.POV_DATA.MEDICAL_DIAGNOSIS",
    "nativeObjectType": "table",
```

Conclusion

A true Data Mesh can be accomplished by directly addressing the four pillars discussed at the start of this paper:

1. Domain-centric ownership – This requires an organizational mindset shift to allowing decentralized ownership of data products.
2. Self-Service Data Platform – Powered by Snowflake data engineering, data warehouse/data lake, Data Science Data Collaboration features, and data sharing.
3. Federated Governance and Security – Immuta's scalable policy management abstraction allows federated governance of the data products using native Snowflake controls.
4. Data as a Product – Immuta's data portal manages discoverability, subscription requests, and audit of data products.

For more information, visit [Snowflake for Data Mesh](#) or download Immuta's [Accelerating Data-Driven Insights with Data Mesh](#) ebook. If you're a Snowflake customer and would like to learn more, please contact your sales lead to schedule a live demo.

About Immuta

Immuta is the **leader in Data Security**, providing data teams one universal platform to control access to analytical data sets in the cloud. Only Immuta can automate access to data by discovering, protecting, and monitoring data. Data-driven organizations around the world trust Immuta to speed time to data, safely share more data with more users, and mitigate the risk of data leaks and breaches. Founded in 2015, Immuta is headquartered in Boston, MA.

