eBook

Data Governance Architecture Patterns for Data Architects

databricks

Contents

Contents

Introduction

The dynamic interplay of data, analytics and AI is driving a wave of transformative innovations across diverse sectors, reshaping revenue streams and redefining corporate management paradigms.

Governance ensures data and AI products are consistently developed and maintained, adhering to precise guidelines and standards. It's the blueprint for organizations to bring solutions and data vision to life with consistency, guidelines and standards. It enables scale and speed for data engineers through repeatable workflow management. It allows you to collaboratively build and operationalize AI models for data scientists with transparency. It's security for data managers, ensuring data assets are shared far and wide to benefit all, yet are private when needed. It's trust for executives, with transparency of business insights based on their data and AI assets. It also drives operational efficiency for finance, particularly when powered by Databricks Unity Catalog.

This eBook provides practitioners with tools, tips, best practices and methodologies that drive successful data and AI governance implementation using Unity Catalog. Additionally, it uncovers the role of the Databricks Data Intelligence Platform as a catalyst for streamlining these transformative efforts. As organizations navigate an era characterized by AI-centric data strategies, the convergence of innovation and governance becomes the foundation for sustainable growth and responsible technological advancement.

Databricks Unity Catalog is the industry's only unified governance solution for all of a company's data and AI — across clouds and data platforms. Its foundation is the Databricks Data Intelligence Platform, which understands the uniqueness of your data — and drives the most comprehensive and unified governance solution for all of your company's data and AI. And it's built on a lakehouse to be open, scalable, low cost and high performance — the best of all worlds!

So, what are Unity Catalog's main value levers?

- Mitigating data and architectural risks
- Ensuring compliance
- Accelerating innovation
- Reducing platform complexity and cost while improving operational efficiencies
- Enabling collaboration and monetizing the value of data

How does Unity Catalog specifically provide for these positive outcomes?

- It provides a unified view and discovery of the entire data estate for accelerating innovation, which is quite helpful for data and solution architects. Having a unified solution for access management and auditing not only lowers license costs (often by 50% or even 90%), but it also enhances data and AI security
- By offering comprehensive data and AI monitoring and reporting, it improves trustworthiness for nontechnical users and experts alike
- By providing a collaborative environment — platform agnostic for data and model sharing — it democratizes every person within a business, unlocking new business values

Throughout, governance is simplified with intelligence. Data intelligence enables context-aware search using AI-powered knowledge engines. It automatically generates AI-enhanced descriptions, comments and documentation. It finds data using natural language — so that nontechnical people can ask questions themselves — without having to go through their IT staff to create SQL queries. Questions like, "Which marketing campaigns are most successful?" or "What vendors have been least productive across my supply chain?" This is finally the real-life democratization of data.

The democratization is broad — Unity Catalog unifies data and AI-enhanced governance across BI, data warehousing, data engineering, data streaming, data science and ML. It provides views and controls across all structured, semi-structured, unstructured and streaming data, as well as AI models, notebooks, workplaces, files, tables and dashboards. It provides more informative and actionable oversight through AI-enhanced holistic search, discovery and monitoring of usage trends, data lineage, discovery and model transparency. Whether with natural language or with SQL, organizations that harness this transformative AI-enhanced technology successfully unleash all of their data assets and become leaders for the future.

Unity Catalog Fits Across Cloud Provider Abstractions

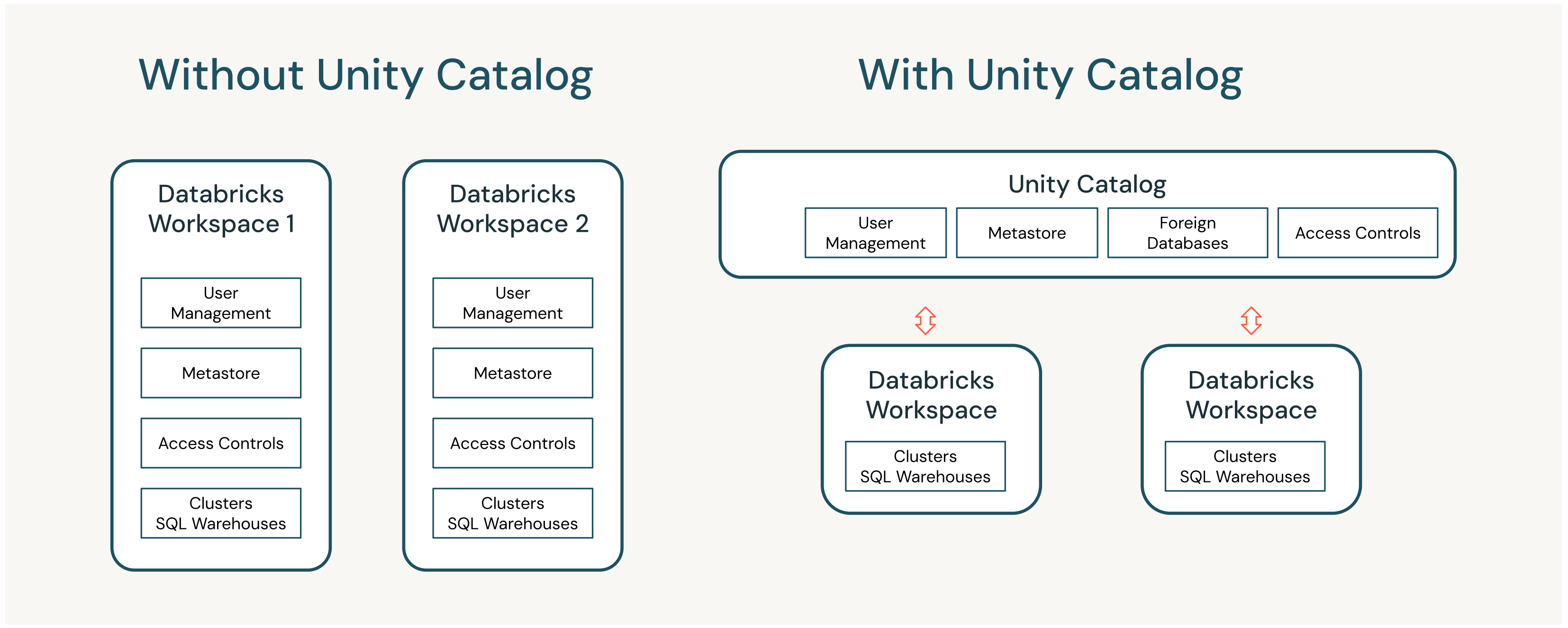
The Databricks Data Intelligence Platform is a cloud-native data platform available on all three major cloud providers: AWS, Azure and GCP. Customers have the flexibility to choose their preferred cloud provider when deploying Databricks, with a consistent experience across each of them. They can also opt for a multicloud strategy, seamlessly executing and integrating workloads across different providers.

In this context, effective data governance is essential for organizations relying on data, analytics and AI, yet challenges persist in implementation due to inadequate organizational processes and resources. These challenges are often exacerbated by the evolving role of chief data officers (CDOs). As a result, governance responsibilities often remain decentralized, leading to policy variations and multiple governing bodies across organizational divisions. This lack of centralized data governance function is termed *distributed governance*. Within the Data Intelligence Platform, we facilitate the implementation of a distributed governance model by leveraging Unity Catalog.

Unity Catalog enables the management of structured and unstructured data, as well as AI models within Databricks and external sources, under a single governance framework. Unity Catalog creates a consistent experience across cloud providers, abstracting many specifics across access, governance and identity. In this initial chapter, we delve into the specific aspects of each major cloud provider — AWS, Azure and GCP — and explore how Unity Catalog aligns with them.

Where Unity Catalog fits within each cloud provider

Before Unity Catalog, each Databricks workspace functioned monolithically, with support for dedicated users, pipelines, clusters, SQL warehouses and the concept of a local-to-workspace metastore. With Unity Catalog, the Databricks estate is unified for data and users, allowing these concepts to be shared across workspaces. Workspaces, in turn, continue to support the organization and isolation of nondata, workloads and compute.

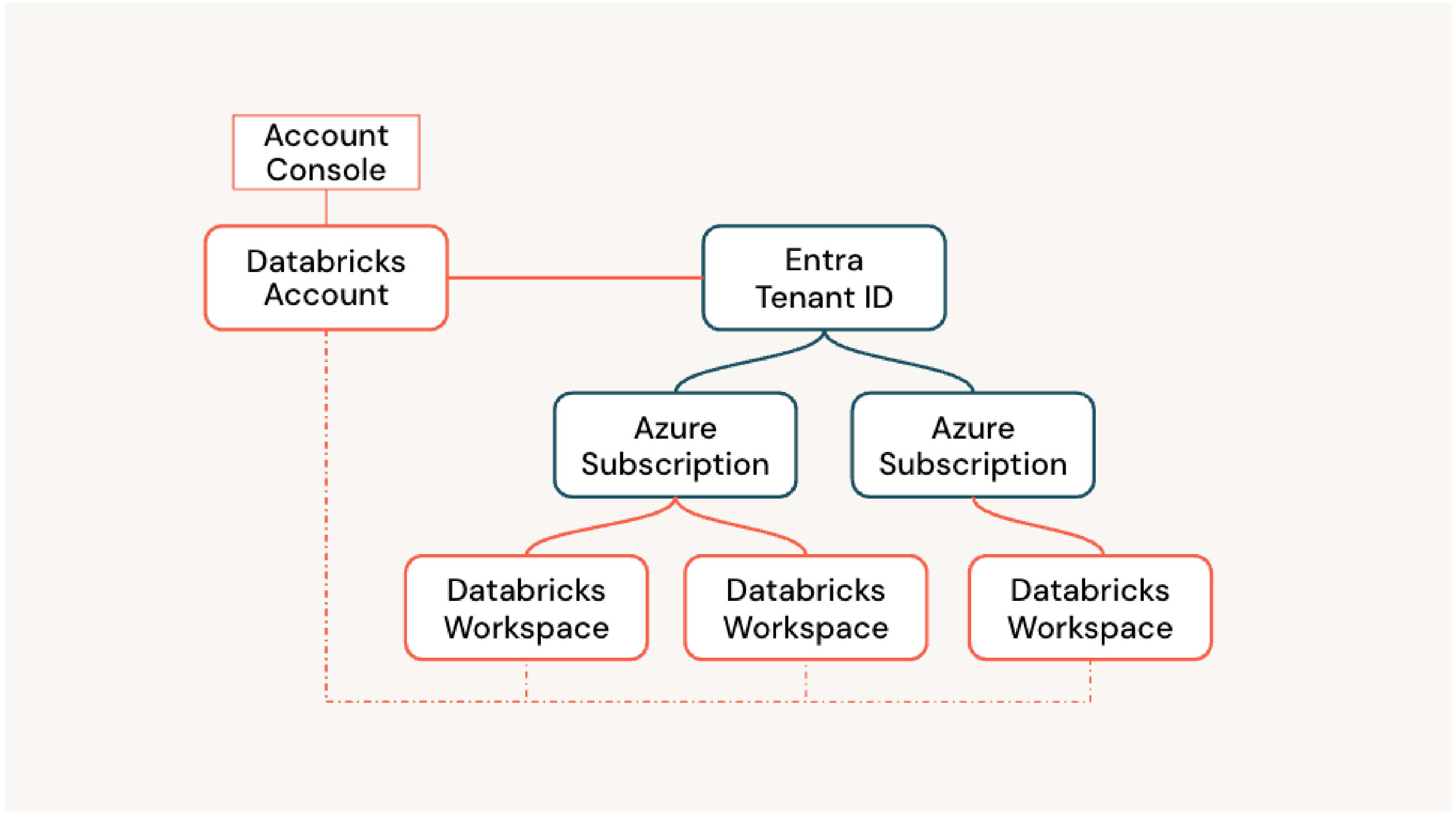


With Unity Catalog, the concept of the Databricks account gains increased significance compared to before. Now, users and metastore reside within the Databricks account, positioned one level above the workspaces.

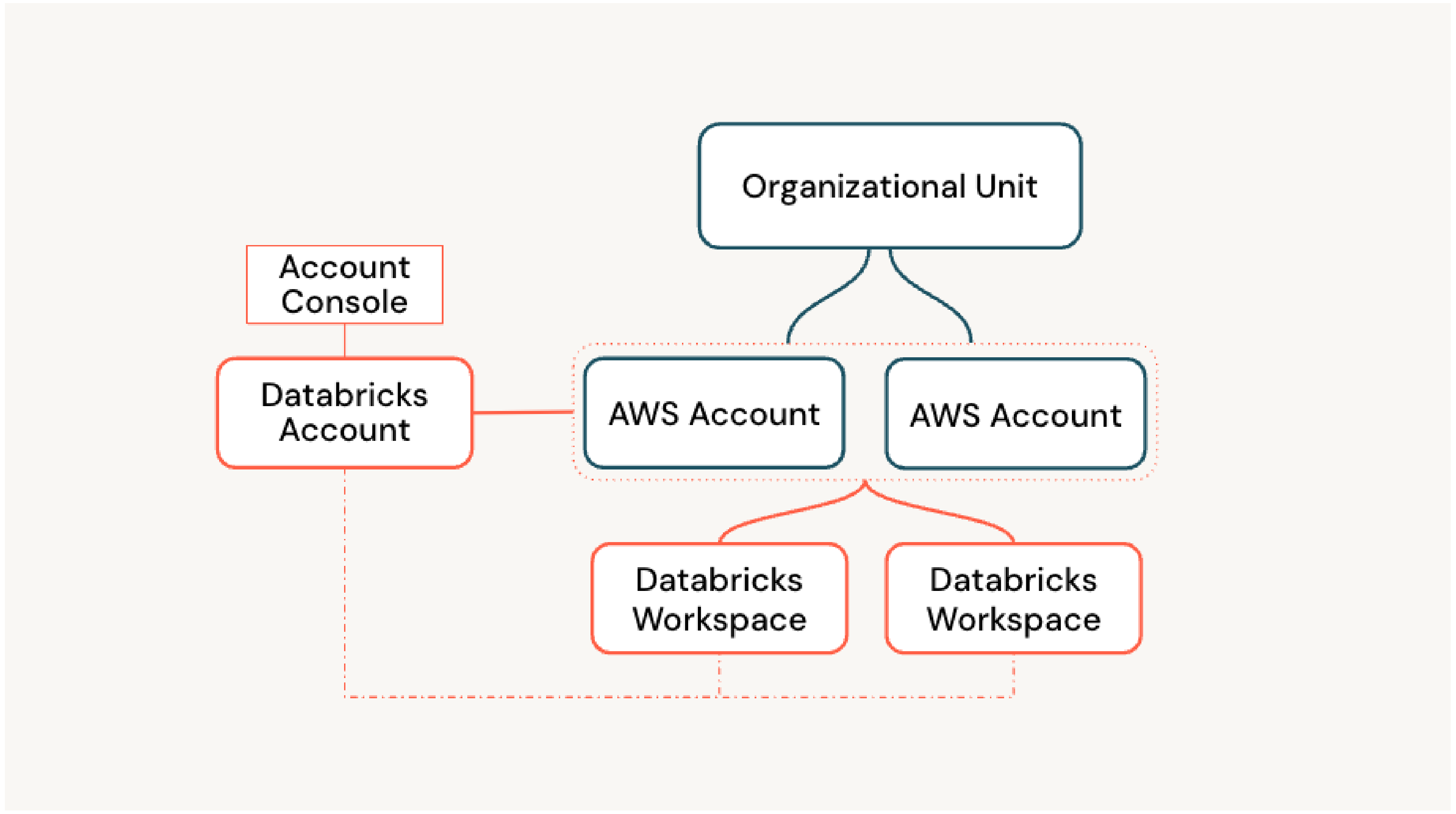
This new architecture also alters the distribution of responsibilities within Databricks. While the *workspace admin* previously held the sole administrative role, the introduction of the *account admin* role adds a new layer. The workspace admins retain control over individual workspaces, including user assignments and compute resource management. In contrast, the account admin oversees permissions related to data within Unity Catalog across all metastores, as well as other broader capabilities such as network connectivity for serverless compute that extend beyond a single workspace. Additionally, each metastore features a *metastore admin* role responsible for managing the data within it.

Depending on the cloud provider, the Databricks account maps differently, as shown in the following figures.

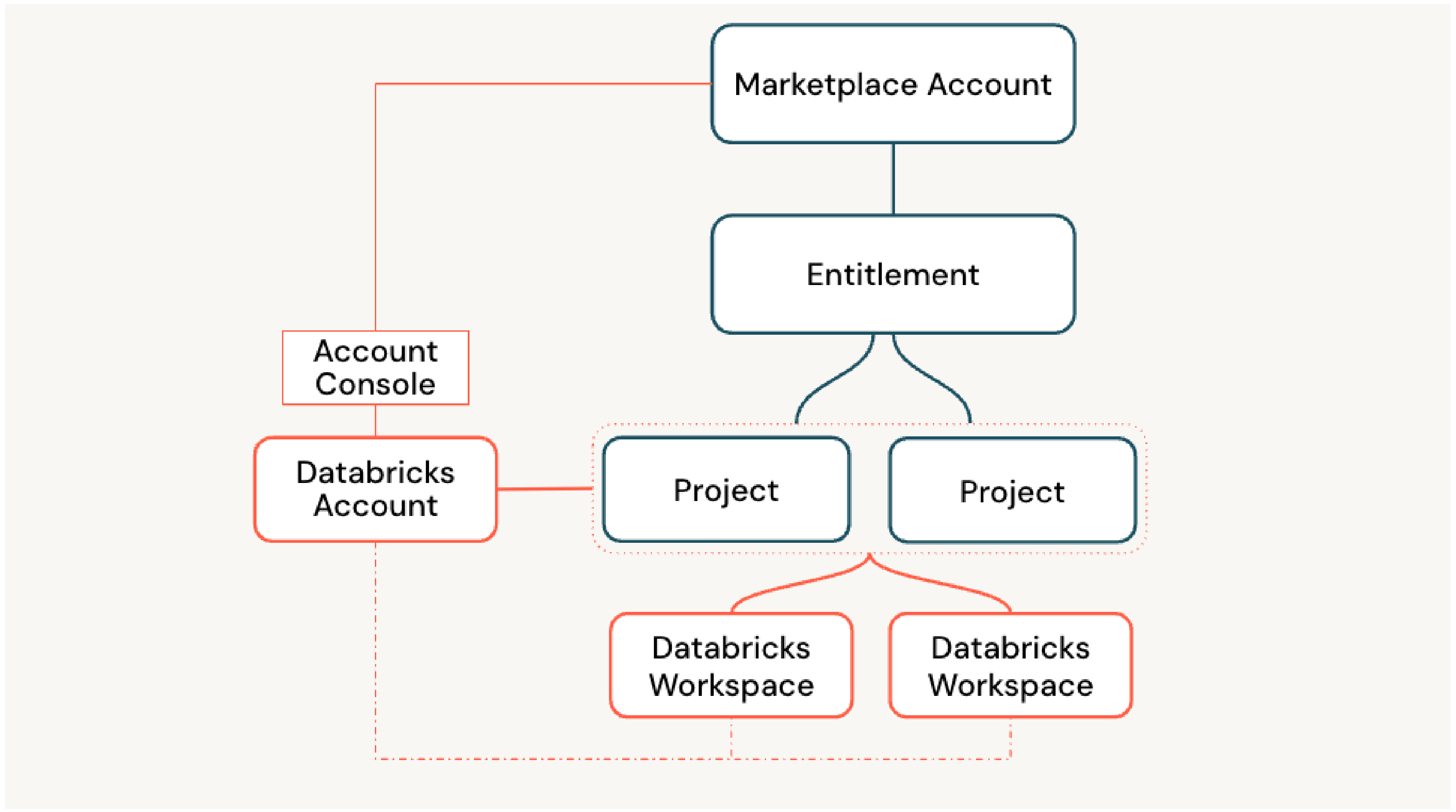
On Azure, the Databricks account is linked to the Microsoft Entra tenant ID. Any subscription inside the tenant is mapped to the same Databricks account. Consequently, any workspace in any subscription and resource group will be able to access the same metastore in the respective regions if admins allow this.



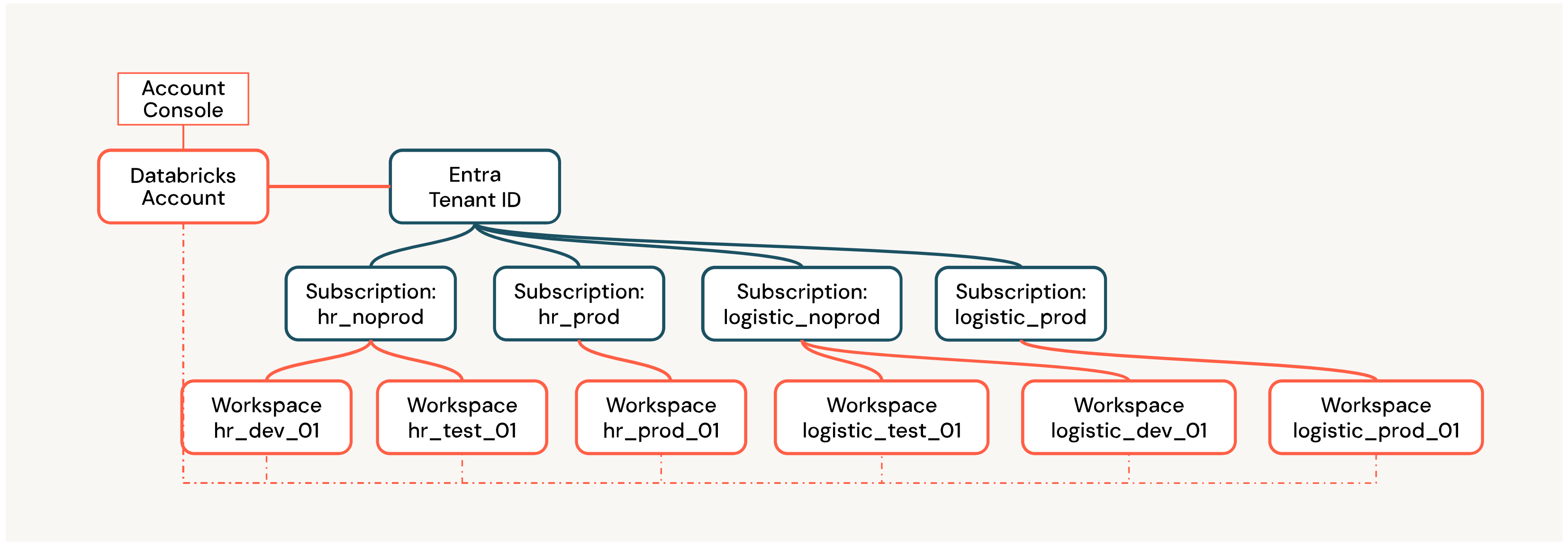
On AWS, the Databricks account maps to the organizational unit. It's common on AWS to organize the cloud environment in different accounts, and you can link one or more of them to your Databricks account so that all workspaces can access the same metastore.



On GCP, a Databricks account maps to a Marketplace account. Similarly to AWS, multiple projects on GCP can be linked to Databricks and all workspaces in them will be aggregated.



In the context provided, various strategies exist for organizing cloud environments within Azure, AWS or GCP, typically involving subscriptions, accounts and projects. Factors influencing these strategies often include business units, divisions and environments (such as dev/test/prod or production/non-production). Consider a scenario where a company organizes Azure subscriptions based on business units and environments. In this example, business units like HR and Logistics utilize Databricks, each mandated to maintain three environments categorized in two subscriptions as *prod* and *non-prod*: development, test and production. The Databricks account alignment for this is illustrated in the figure below.



How Unity Catalog abstracts the specifics of each cloud provider

Each cloud provider employs distinct underlying and proprietary concepts and features that Databricks utilizes, ranging from authentication methods like identity and access management (IAM) roles and service principals to storage accounts such as GCS, S3 and ADLS. While Databricks has always functioned as a multicloud solution, the specifics were previously exposed directly to end users rather than abstracted. For instance, in AWS, users had to utilize IAM roles configured within clusters to enable access to data in an S3 bucket at a specified path (s3://some-bucket). Similarly, on Azure, configuration of a service principal was necessary to read from an ADLS Gen2 storage, while on GCP, a service account was required to read from GCS. In the context of a multicloud data strategy, it becomes evident that such scenarios can quickly become complicated and confusing for end users. For example, if a user creates a notebook to read from S3, the same notebook would not function when attempting to access data from ADLS or GCS if moved to a workspace in another cloud.

With the unified governance facilitated by Unity Catalog, Databricks has also unified the approach to accessing data and services across different cloud providers, utilizing abstractions that are agnostic to the chosen provider. These abstractions include credentials for services and storage and external locations. Administrators only need to engage with the specifics of cloud providers when creating these objects; thereafter, they can operate within Databricks exclusively.

A service credential is used to access cloud-specific services other than storage such as SQS for AWS or Key Vault for Azure, via the respective SDKs. Service credentials give a uniformly managed solution to authenticate via Unity Catalog that is no longer cluster-bound via IAM roles, service principals or service accounts.

In a similar way, a storage credential functions like service credentials but is specific to storage, acting as a secure container that embeds the authentication mechanism for Unity Catalog when accessing it. This authentication is implemented as an IAM role on AWS, a managed identity on Azure and a service account on GCP.

Once authentication means are established, administrators require another secure element, the external location, to map the storage credential to a specific storage account. Administrators assign the storage credential to a particular S3 bucket, ADLS Gen2 or GCS when creating an external location.

With these two secure elements in place, end users can interact with their data independently from the cloud provider. Administrators grant end users permission to utilize external locations, enabling them to create tables and volumes for accessing structured and unstructured data, respectively.

In summary, Unity Catalog aligns with various cloud provider models and further provides mapping concepts from cloud provider storage and access abstractions into its hierarchy. This eases the burden of administration by allowing users to manage access and data in one place in a governed, secure and scalable way without requiring cloud-specific knowledge.

Designing Your Three-Level Namespace

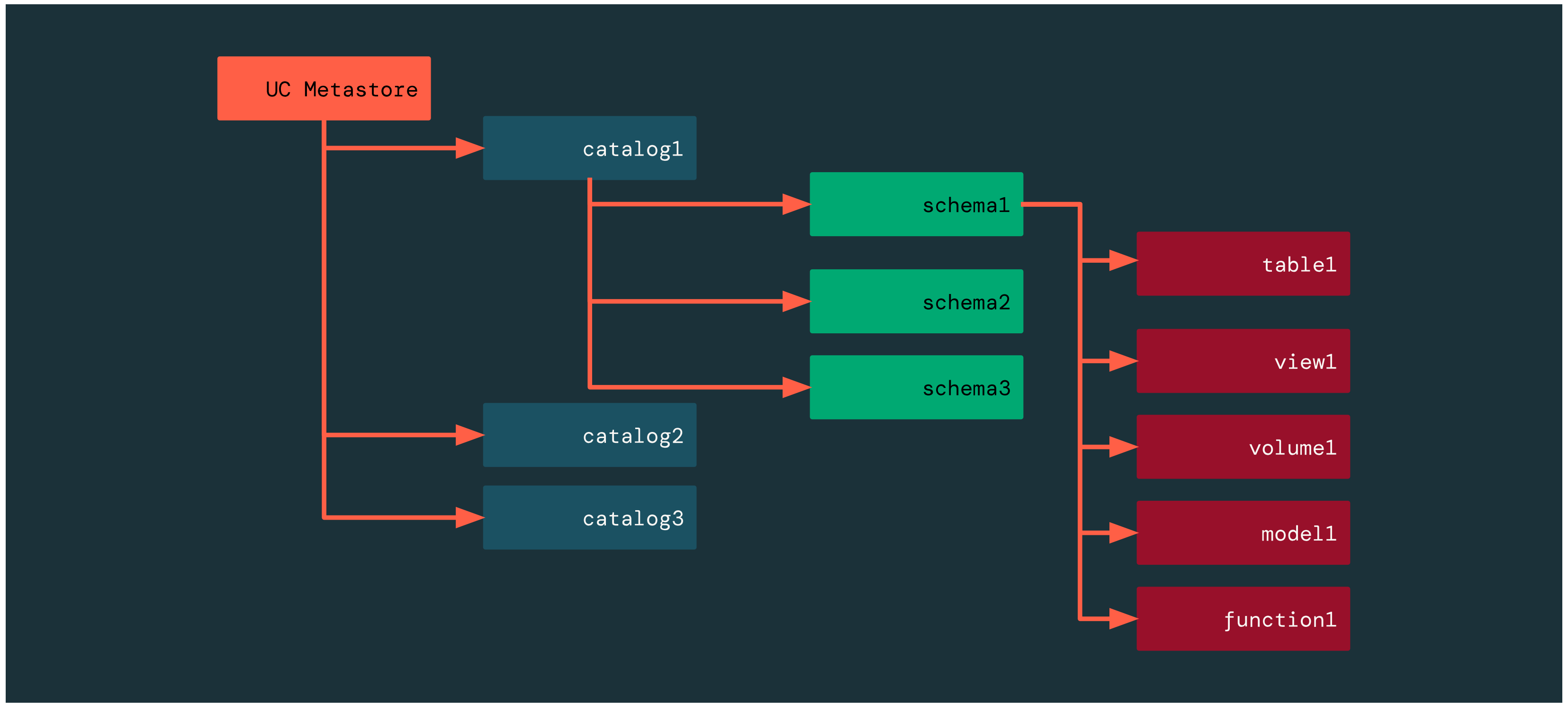
Unity Catalog provides a three-level namespace that can be used to represent a logical layout of how data is accessible and managed in your organization. The three levels are catalogs, schemas and assets.

Catalogs represent the top level of the structure. They can be made available for selective workspaces, which allows different operating units to enforce that their data is only available within their environments. This also allows teams to enforce the availability of data across software development lifecycle (SDLC) scopes (e.g., prod data in prod, dev data in dev).

Schemas are defined inside the catalogs. They can serve as a grouping for per-domain data assets.

Assets are inside schemas. They include the following objects:

- **Data assets:** Tables, views, materialized views, streaming tables
- **Machine learning models**
- **Unity Catalog Volumes:** These are logically assigned to a specific catalog and schema and represent folders with files (e.g., images or documents)
- **Functions:** Reusable procedures that can be written in SQL and Python



Catalogs can also be *internal* and *foreign*, depending on the data source.

Internal catalogs are created, owned and registered within the metastore based on data stored in relevant cloud storage. Foreign catalogs can originate from federated SQL databases or other metastores.

Unity Catalog supports two types of foreign catalogs: *Delta shares* registered as catalogs and foreign catalogs registered via Lakehouse Federation.

Naming conventions

When building the logical structure of the catalogs, consider the following axes of potential separation.

- **Region:** The region in which the data of a catalog is *physically* stored. This indication is quite useful when a multicloud (or multiregional) organization requires registering foreign catalogs from different regions.
- **Provisioner:** The division responsible for *providing* this asset (e.g., is it a *business unit* or a *central* data team of the platform).
- **Environment:** Indicates separate dev, staging and prod environments.
- **Logical data domain:** Identifies inside business units or cross-unit teams.

Using these rules, we can combine catalog names into the following naming convention:

{region}_
{provisioner}_
{environment}.{team_name or domain_name}.{asset_name}

Common scenarios

Following are some scenarios that allow us to conceptualize these naming conventions.

- **Naming by Business Unit:** An internal catalog with data provided from a business unit (BU) named BU1, with data related to marketing domain, inside the prod environment

  Since the table is in the internal catalog, we can omit the region name to avoid creating a long name. Applying our naming convention logic, the name for this catalog would look like this:

  Bu1_prod.marketing.some_table_name

- **Naming by Region:** A foreign catalog, sourced from region R1, distributed by the central team, with data related to the finance topic for dev environment

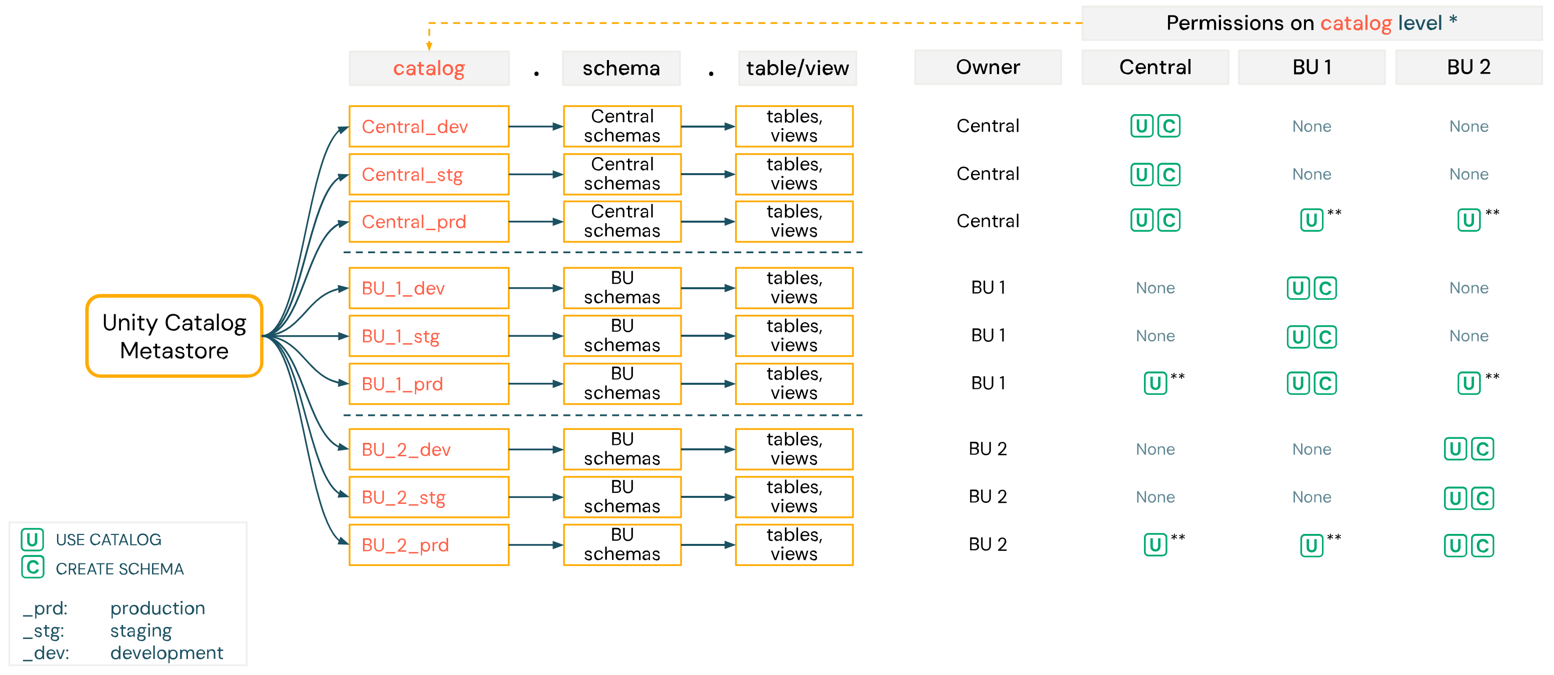
  r1_central_dev.finance.some_table_name

- **Naming by Multicloud or Multiregion:** A multicloud and multiregional organization, which has several separate data platforms, and each part of the platform is represented inside each region

  The generic structure of such a scenario would look like this:

  {cloud}_{region}_{team}_{env}.{domain}.{asset_name}

It's also important to understand how the logical blocks should be distributed in terms of ownership. This topic is extensively covered in the following chapters, but the overall layout is shown in the figure below.



catalog	schema	table/view	Owner	Central	BU 1	BU 2
Central_dev	Central schemas	tables, views	Central	U C	None	None
Central_stg	Central schemas	tables, views	Central	U C	None	None
Central_prd	Central schemas	tables, views	Central	U C	U**	U**
BU_1_dev	BU schemas	tables, views	BU 1	None	U C	None
BU_1_stg	BU schemas	tables, views	BU 1	None	U C	None
BU_1_prd	BU schemas	tables, views	BU 1	U**	U C	U**
BU_2_dev	BU schemas	tables, views	BU 2	None	None	U C
BU_2_stg	BU schemas	tables, views	BU 2	None	None	U C
BU_2_prd	BU schemas	tables, views	BU 2	U**	U**	U C

General recommendations

When designing the catalog/schema structure, there are a few important aspects to keep in mind.

- **Strike a balance:** It's better to have a balance between the amount of catalogs and the number of schemas inside them. For example, choosing between 10 to 20 catalogs with thousands of schemas inside them or 200 to 300 catalogs with 100 to 200 schemas. This will increase the manageability of the setup.
- **Work with catalogs:** Catalogs serve as a very convenient boundary for teams using a Data Mesh architecture. Self-service teams can have their own catalogs and set up their own logical grouping inside these catalogs without having to reach out to account and metastore admins for changes.
- **Use automation tooling:** When it comes to managing the catalog structure at scale, we strongly recommend you use automation tooling. Databricks provides capabilities to create, control and manage catalogs and schemas via Databricks Terraform Provider. For advanced cases and integrations with third-party systems, you should use Databricks Rest API, and in particular, Databricks Python SDK.

Manage Your Data

In this chapter, we'll review various types of data and the relevant functionality provided by Unity Catalog to access and govern them in a secure fashion.

Supported storage types

For data sources that can be accessed and governed via Unity Catalog, we can classify the potential storage types into the following categories:

- Cloud storages with structured data
- Cloud storages with unstructured data
- Relational databases with structured data

Given these types, we can easily map to different technologies provided within Unity Catalog:

1. **Cloud storages with structured data:** Delta tables, as well as structured file formats like Parquet, are usually stored inside the cloud storage, in some kind of path-like layout, for instance:

    /org-root/teamA/events/ contains a Delta table with _delta_log folder and Parquet files

    To access and govern such data, use external locations in combination with storage credentials.
2. **Cloud storages with unstructured data:** Unstructured data (e.g., images or PDF files) is also stored in the cloud storage:

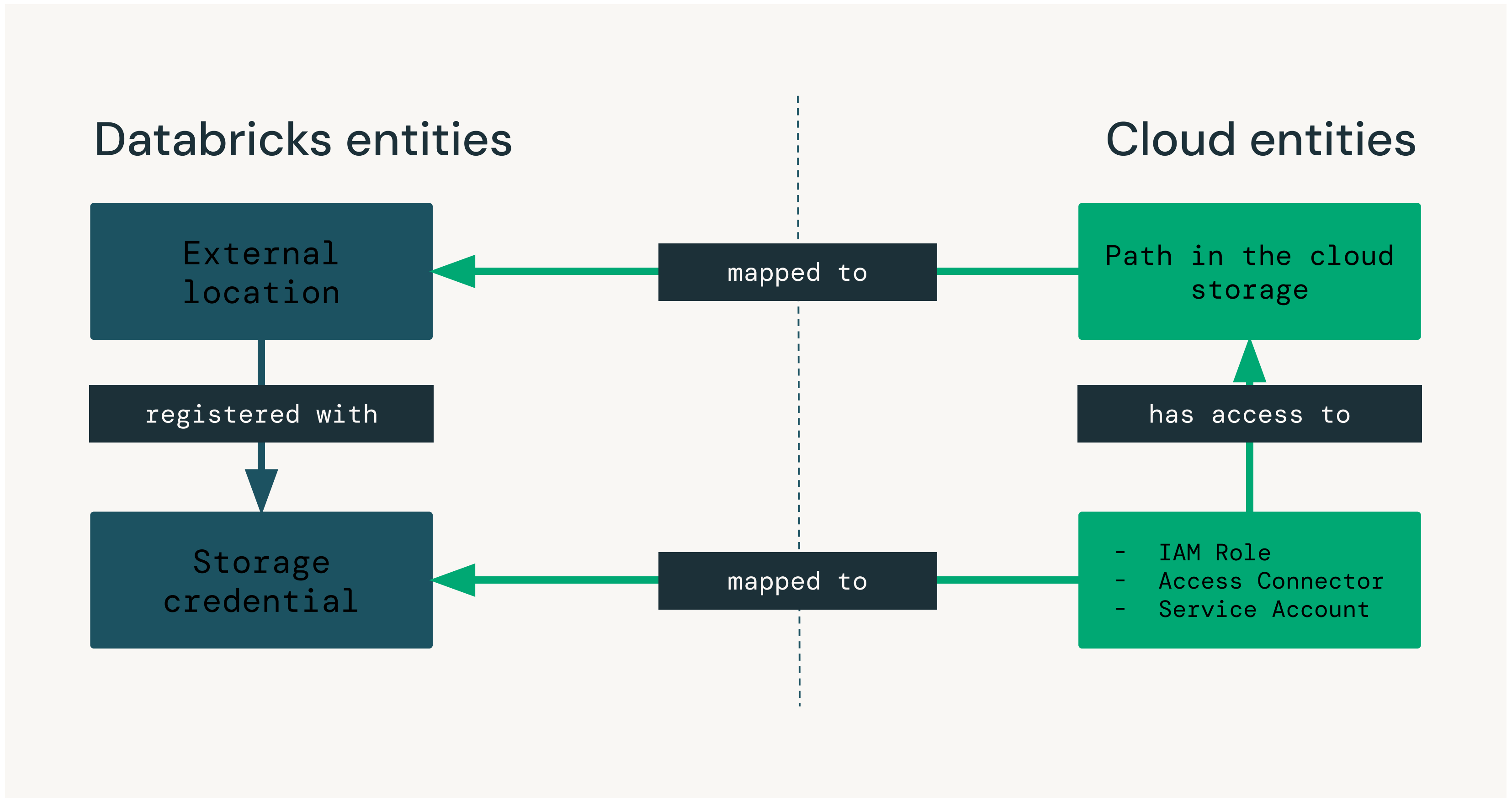
    /org-root/teamB/docs contains hundreds of PDF files

    Unstructured data is a core object in Unity Catalog. Access and governance for such folders (or subpaths) is governed via Unity Catalog Volumes.
3. **Relational databases with structured data:** Relational databases such as MySQL can store schemas with tables inside them. Usually such databases store structured information. To access, govern and control data from relational databases such as MySQL, Postgresql, SQL Server and more, Unity Catalog provides the Lakehouse Federation feature.

EXTERNAL LOCATIONS AND STORAGE CREDENTIALS

To access the data inside the same cloud region as the Unity Catalog metastore, the following is required:

- **Storage credentials:** These represent relevant cloud-level entities that are entitled to read the data on behalf of the users. For example, on AWS this role is dedicated to IAM roles, and on Azure these entities are managed identities (and service principals in rare cases).
- **External locations:** These are cloud storage objects (e.g., buckets, containers or subpaths inside these buckets or containers) that can be accessed by users through Unity Catalog. External locations can't be registered without a storage credential.



A combination of these entities is typically how teams access their existing data on the cloud storage in a secure way. For better manageability, multiple external locations can use the same storage credential to access the data as soon as the storage credential is entitled for the relevant operations on the underlying cloud storage objects.

To provide teams with self-service capabilities, they can be granted permissions to create storage credentials and external locations. Such a setup allows teams to easily plug in their existing cloud storages without being dependent on the central platform team.

MANAGED AND EXTERNAL TABLES

Unity Catalog metastore also comes with the capability to register managed storage. Managed storage is still storage located on the data plane (in the customer object storage), but it doesn't require users to create folder-level structures to separately store the data.

Unity Catalog supports setting managed storage at several levels:

- **Metastore:** Can have a root managed storage, which needs to be registered during the metastore creation process. Though this is a simpler approach, enterprise-grade organizations usually require the data to be stored in logically, per-environment separated cloud object storages.
- **Catalog-level storage:** Can be set when the catalog is created by using the LOCATION clause. This is the most common way to logically separate the managed storage across the environments and teams, which also provides a lot of flexibility for their operations.
- **Schema-level storage:** Is used in cases when there's a strict organizational requirement to store data even at a schema level inside separated cloud containers or buckets. This may be useful for cases when teams perform so many IO operations that they may hit the underlying "per-container/per-bucket" limits of the cloud object storage. To add a managed location to a schema-level storage, the LOCATION clause should be used when creating a schema.

In practice, **catalog-level storage by location** is the most commonly used setting for enforcing enough logical control without introducing too much management overhead.

Since tables can be both managed and external, some logical questions arise: Which tables shall be used in which cases, and what are the benefits of using managed tables?

Managed tables shall be used in most cases, specifically when:

- Performance of accessing data in these tables is critical for business continuity (e.g., BI and analytics). In such case, a managed table with powerful features like Predictive Optimization can deliver lightning-fast analytics at scale.
- Overall management of the cloud storage is already (or is becoming) a burden for the platform team. In this case it would be a question of improving the efficiency of internal processes and decreasing the dependency on the underlying storage layout.

One use case for an external table is when the table is accessed from a system that's outside of the Unity Catalog control and users can't read data via standard open protocols such as Delta Sharing and ODBC or JDBC.

GOVERNANCE FOR UNSTRUCTURED DATA USING UNITY CATALOG VOLUMES

Given that a lot of modern valuable data can't easily fit into a table-like format, Unity Catalog comes with a tool for governing access to unstructured data called a *volume*.

In short, volumes come on top of the external locations, adding a capability to govern access to the unstructured data. They also provide FUSE-like access to the data stored in them when they're accessed via Databricks clusters.

Volumes are perfect for storing unstructured data and data-related assets such as:

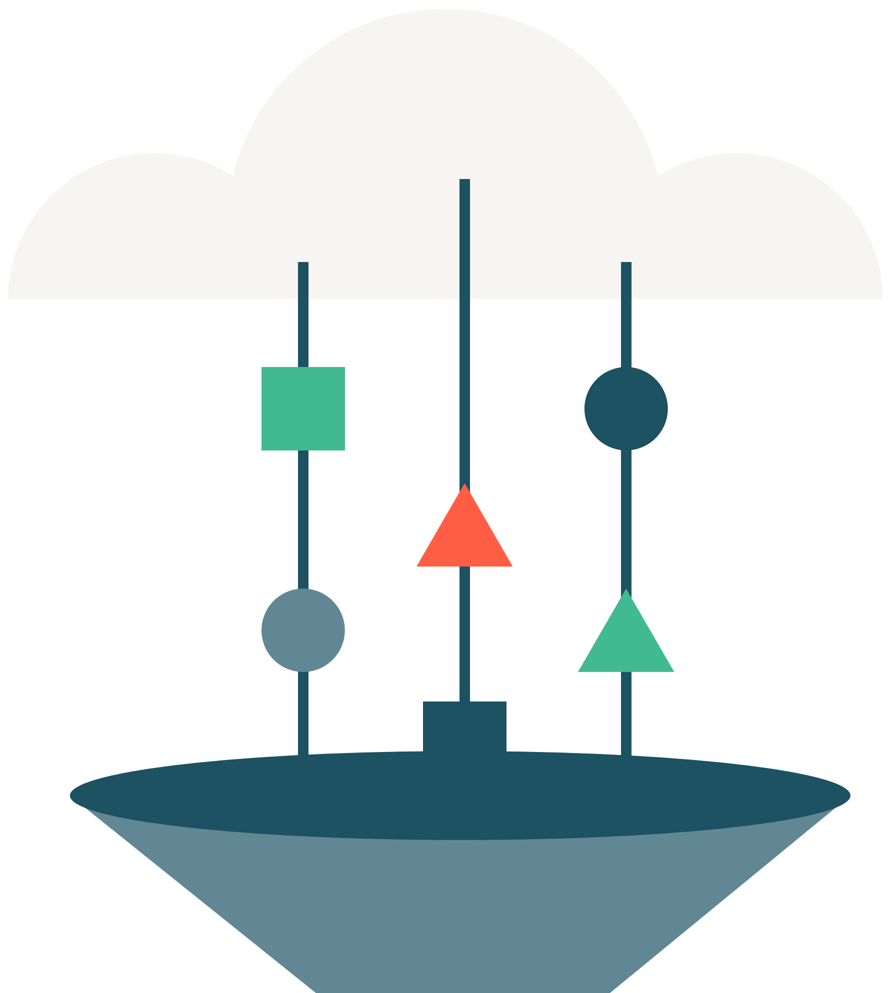
- Spark Streaming Checkpoints
- Dependent libraries (e.g., jars)
- A distribution location for Python packages that are used in the workflows and on the all-purpose clusters

Lakehouse Federation

Lakehouse Federation provides an interface to easily access data inside relational databases. Databricks offers federation into external databases like MySQL, PostgreSQL, SQL Server, Snowflake, BigQuery and more.

Foreign catalogs registered inside the Unity Catalog metastore can be further governed on the Unity Catalog side, which makes access control seamless and integrated with other Unity Catalog tooling.

Lakehouse Federation reduces data fragmentation and access management overhead. Therefore it's crucial to incorporate existing sources that can't be replicated to the data lakehouse into the overall data access pattern enforced by Unity Catalog.



Securing Your Data

In this chapter, we'll look into how Unity Catalog secures and governs data and AI assets, the various access patterns provided by Unity Catalog and some of the key best practices for securing and governing your data with Unity Catalog.

As we've seen in the previous chapters, Unity Catalog provides:

- Fine-grained access control to data and AI assets cataloged under the standard catalog
  - **Tables:** Structured data in various different formats such as Delta and Parquet
  - **Volumes:** Unstructured data formats such as CSV, XLSX, MP3, JSON, etc.
  - **Models:** Machine learning models
- Read-only access to popular database solutions cataloged under the foreign catalog

The data and AI assets under the standard catalog actually live in the cloud provider's object store. Unity Catalog manages access to these assets using the cloud provider–specific mechanisms registered as storage credentials within the Unity Catalog metastore:

- Service principal or managed identity in Azure
- IAM role in AWS
- Service account in GCP

Storage credentials are configured at the Unity Catalog metastore level, usually by the platform team or the workspace admins. The most important thing to note here is that the Unity Catalog service must be the only way to access the object store. If there's a way tvo bypass Unity Catalog and access the object store by any other means, then this would also bypass the entire governance structure of the data platform.

Permissions model

Unity Catalog provides a logical governance layer that abstracts out the file-level access controls. In other words, regardless of the nature of the datasets, Databricks users no longer need to deal with the data in terms of files and folders but only as Unity Catalog objects. Moreover, for all the assets cataloged under Unity Catalog, a single standard governance model applies.

Elaborating on the Unity Catalog governance model, let's look at the important aspects of the Unity Catalog permissions model:

- **Unity Catalog permissions are at the metastore level:** Given that Unity Catalog objects are defined at the metastore level, it's obvious that the permissions associated with these objects are also defined at the metastore level. This means that a user will be able to access the objects from any workspace (associated with that particular metastore) as long as they have permission.
- **Unity Catalog permissions apply to account-level identities:** Permissions on the Unity Catalog objects can be granted only to the account-level identities. So the first step for granting permission is to define or provision the identities at the Databricks account level and assign them workspaces through which users can access the objects.
- **Privilege inheritance:** Privileges are inherited down the Unity Catalog object hierarchy. For instance, if you provide ALL PRIVILEGES for a user at the catalog level, then the user will have ALL PRIVILEGES for all the objects within that catalog.
- **Permission assignment:** Granting permissions on an object can only be granted by the identities with the following permissions.
  - Owner of the object
  - Owner of the catalog/schema containing the object
  - Metastore admin
- **Dropping an object:** Only the object's owner or the owner of the catalog/schema that contains it can drop an object.

Advanced access controls

In addition to granting permissions on objects to individuals and groups, Unity Catalog provides some advanced controls for managing access.

- **Row filtering and column masking:** Unity Catalog allows for fine-grained access controls in the form of dynamic row-level security (RLS) and column-level masking (CLM). These can be applied directly to the tables, preventing the need for data duplication and creating and maintaining views.
- **Attribute-based access control (ABAC):** ABAC allows access control to be conditional based on broader properties of the user, resource and request. It builds on top of and coexists with the Unity Catalog core security model, including all privileges. The core concepts of ABAC are *attributes*, *rules* and *inheritance*. Attributes are derived from the context of the object such as identity, tag and time. Rules use these attributes to decide access. Rules are inherited down the object hierarchy and coexist with existing object grants. ABAC provides enhanced capabilities that enable organizations to fulfill complex governance requirements at scale.
- **Workspace-catalog binding:** A catalog object can be bound to a subset of workspaces in order to restrict access (read-write/read-only) to that catalog from only those workspaces. This provides an additional layer of access control. For example, this feature can help prevent users from accidentally accessing production data from the development workspace — even if they have permission to do so — by binding production catalogs to production workspaces. Unity Catalog now allows binding storage credentials and external locations to specific workspaces for enhanced security.

Access patterns

Unity Catalog object hierarchy, permissions model and advanced access control mechanisms pave the way to various access patterns that simplify the overall governance of data and AI assets. Let's look at some of the most common access patterns.

1. **Isolating assets at the business unit (BU) level:** In this pattern, each BU is assigned a set of catalogs that are independently owned and operated by the business unit. This means the BUs are allowed to create schemas, tables and other Unity Catalog objects within the assigned catalogs and manage access to everything under these catalogs. This pattern implements a federated governance approach, keeping certain aspects of it still centralized. The catalogs can be associated with the object store that belongs to the BUs by setting the managed locations. Furthermore, the catalogs can be bound to the corresponding workspaces of the BUs. These measures facilitate strict isolation of assets at the storage level.

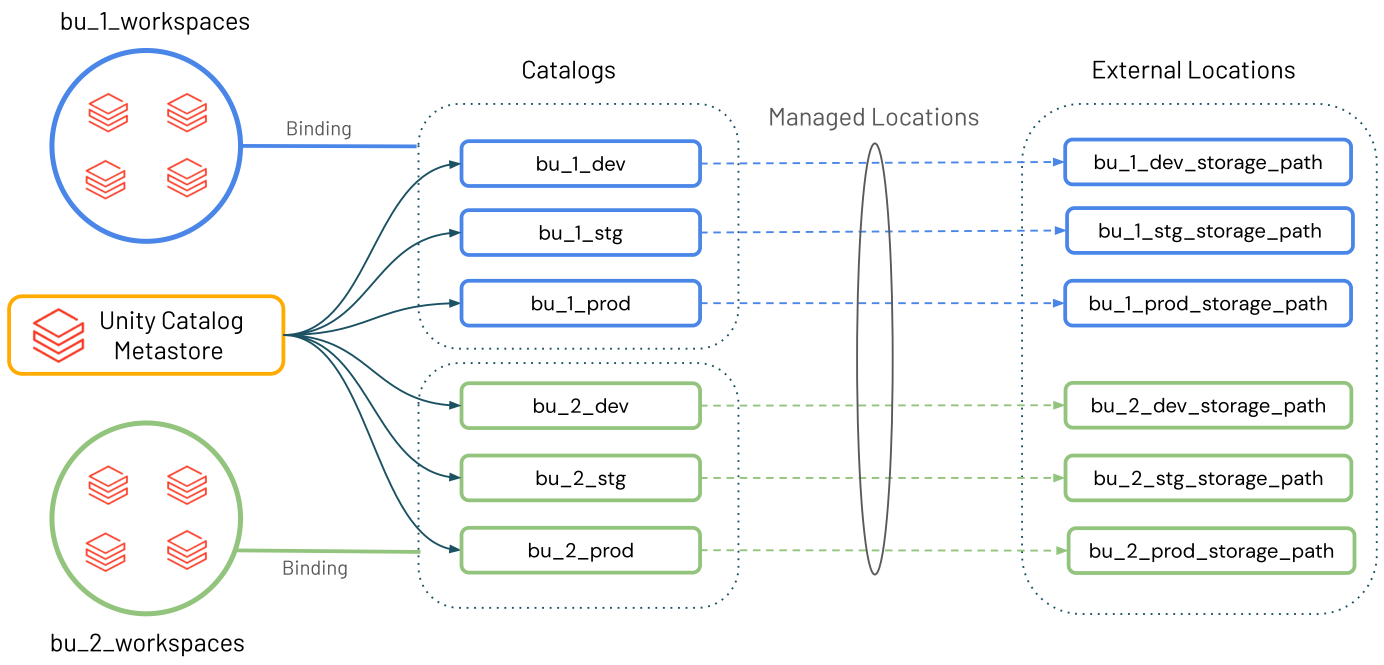


Figure: Business unit–specific catalogs with managed locations and workspace-catalog binding

2. **Cross-functional access:** The power of data lies in how best it can be leveraged. The value of data increases as more and more use cases are powered by it. This means it's crucial to provide data to the relevant teams that can make use of the data, even if they're not a part of the project or the business unit that owns the data. Unity Catalog makes it much easier to share data across teams and business units. For instance, consider a situation where data science or machine learning teams might need access to the production datasets from another business unit. With Unity Catalog, access to the datasets can be granted with simple GRANT statements, and the production catalogs can be bound to the recipient workspaces as read-only catalogs. This creates two layers of protection against accidental writes while providing secure and easy access to the datasets.

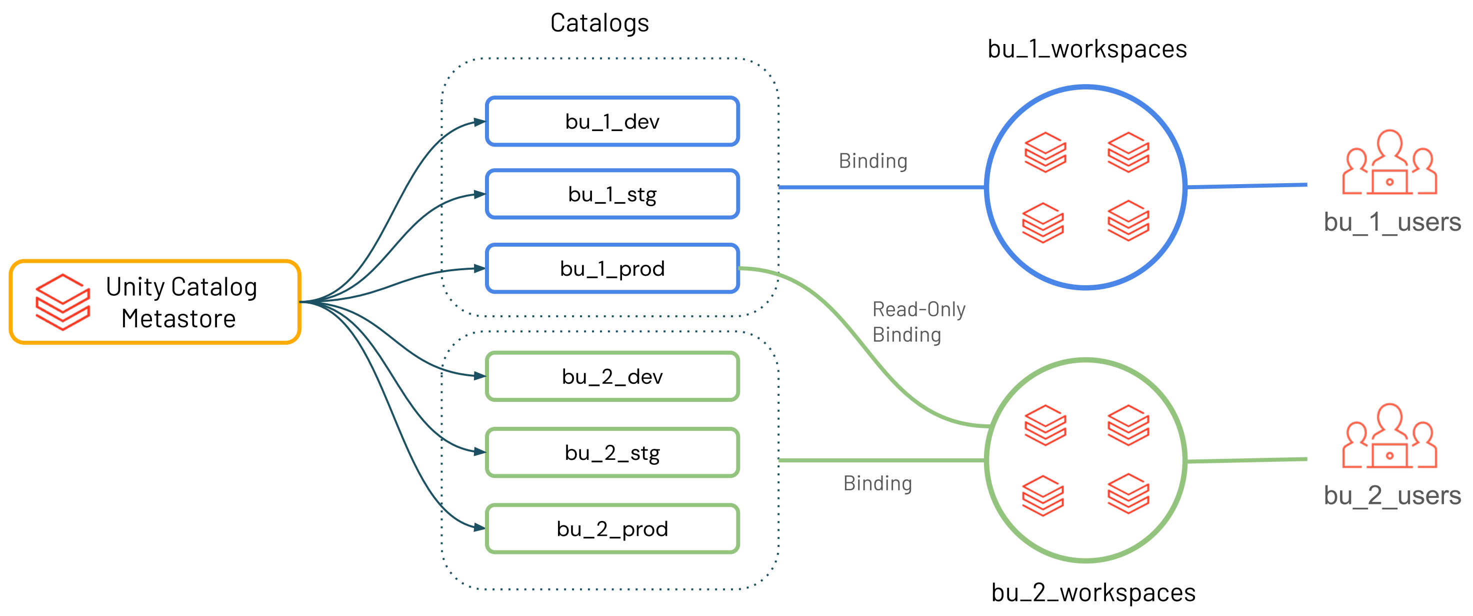


Figure: BU 2 users with access to BU 1 prod catalog with read-only workspace-catalog binding

3. **Data as products:** Data and AI artifacts have emerged as valuable assets that have value beyond departmental or even organizational boundaries. As a result, the concept of data as a product (DaaP) is becoming more and more popular, and many organizations are trying to develop data products and make them available to others. This poses several challenges, including data product cataloging, discovery, access mechanisms, governance and observability. Unity Catalog can help address some of these challenges. For details about implementing data products with Databricks, please refer to this blog post.

These are just a few examples of how the Unity Catalog features and permissions model can be leveraged to set up different access patterns for easy and secure access to all your data and AI assets. With Unity Catalog, it's easy to develop other patterns to address the unique requirements of a given organization.

Best practices

Following are some of the best practices for keeping your data and AI assets secure with Unity Catalog.

- Lock down the object store and prevent external data access. The data in the object store associated with Databricks must be accessible only through Unity Catalog.
- Grant access to groups instead of individuals. This practice simplifies permissions management, especially when onboarding and offboarding users.
- Powerful roles such as *metastore admin* should be granted to an empty group. As needed, a person or service principal can temporarily be elevated to a metastore admin role. Since metastore admin is a role with very high privileges, this practice will avoid any accidental incidents and enforce the least privilege principle. Privileges in Unity Catalog are inherited, so it's better to provide specific access based on necessity rather than providing additional privileges to those who don't need it. This is known as *the least privilege principle*. When granting access to Unity Catalog objects, follow the least privilege principle.

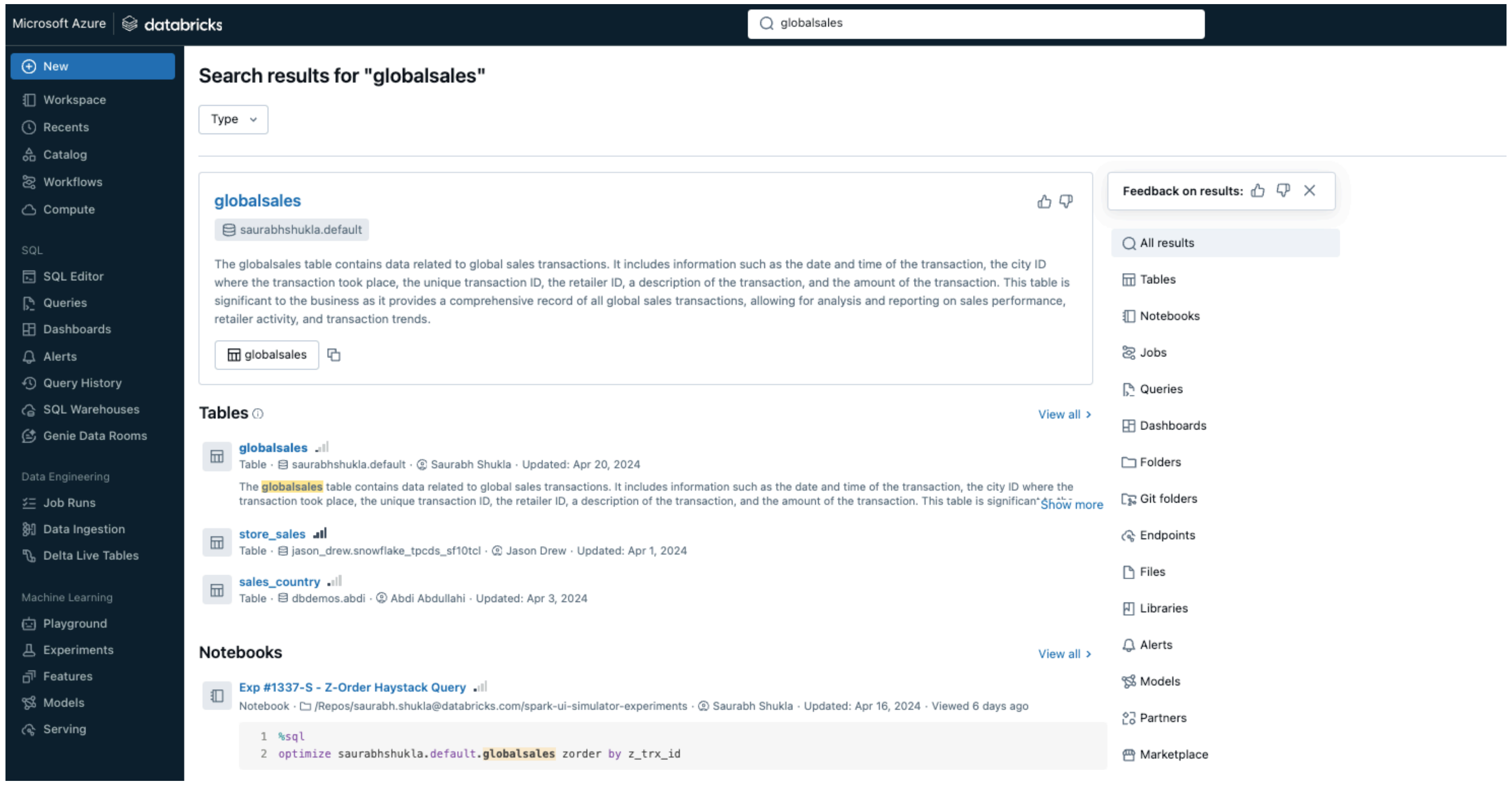
Searching for Data

Intelligent search is critical for working effectively with the data and AI assets in the Databricks Data Intelligence Platform. When considering search capabilities, it's important to address two critical use cases:

1. **Navigation:** Helping users find what they're looking for quickly and efficiently. This usually implies the user already knows precisely what they want to find.
2. **Discovery:** Helping users who might have a general notion of what they want but don't know what specific things to search for

Databricks search

Databricks search allows users to search for tables registered in Unity Catalog, notebooks, queries, dashboards, alerts, files, folders, libraries, jobs, repositories, endpoints, models, partners and Databricks Marketplace listings in your Databricks workspace.



Search results for "globalsales"

globalsales

saurabhshukla.default

The globalsales table contains data related to global sales transactions. It includes information such as the date and time of the transaction, the city ID where the transaction took place, the unique transaction ID, the retailer ID, a description of the transaction, and the amount of the transaction. This table is significant to the business as it provides a comprehensive record of all global sales transactions, allowing for analysis and reporting on sales performance, retailer activity, and transaction trends.

globalsales

Tables — View all ›

globalsales
Table · saurabhshukla.default · Saurabh Shukla · Updated: Apr 20, 2024
The globalsales table contains data related to global sales transactions. It includes information such as the date and time of the transaction, the city ID where the transaction took place, the unique transaction ID, the retailer ID, a description of the transaction, and the amount of the transaction. This table is significan Show more

store_sales
Table · jason_drew.snowflake_tpcds_sf10tcl · Jason Drew · Updated: Apr 1, 2024

sales_country
Table · dbdemos.abdi · Abdi Abdullahi · Updated: Apr 3, 2024

Notebooks — View all ›

Exp #1337-S - Z-Order Haystack Query
Notebook · /Repos/saurabh.shukla@databricks.com/spark-ui-simulator-experiments · Saurabh Shukla · Updated: Apr 16, 2024 · Viewed 6 days ago

```
%sql
optimize saurabhshukla.default.globalsales zorder by z_trx_id
```

Databricks search leverages DatabricksIQ — the Data Intelligence Engine for Databricks — to provide a more intelligent AI-powered search experience. AI-generated comments use large language models (LLMs) learned from multiple sources to automatically add descriptions to tables and columns managed by Unity Catalog. These comments equip the search engine with an understanding of unique company jargon, metrics and semantics, providing the context required to make search results more relevant, accurate and actionable.

Databricks search offers *semantic search*, which interprets the meaning of words and allows natural language queries. It retrieves semantically similar assets and combines them with keyword searches. For instance, if a user searched for "What should I use for geographies," the search engine would focus on "geographies" and find related terms containing geographic attributes such as cities, countries, territories, geolocations, etc.

Databricks search separates search terms from filters, enhancing natural language queries and *search query understanding*. For example, the query "Show me tables about inspections" would yield "inspections" (key term) and "table" (object type).

Using popularity signals based on user interactions to rank objects, Databricks search delivers improved *relevance with popularity*. This method ranks the most popular assets higher, reducing trial and error.

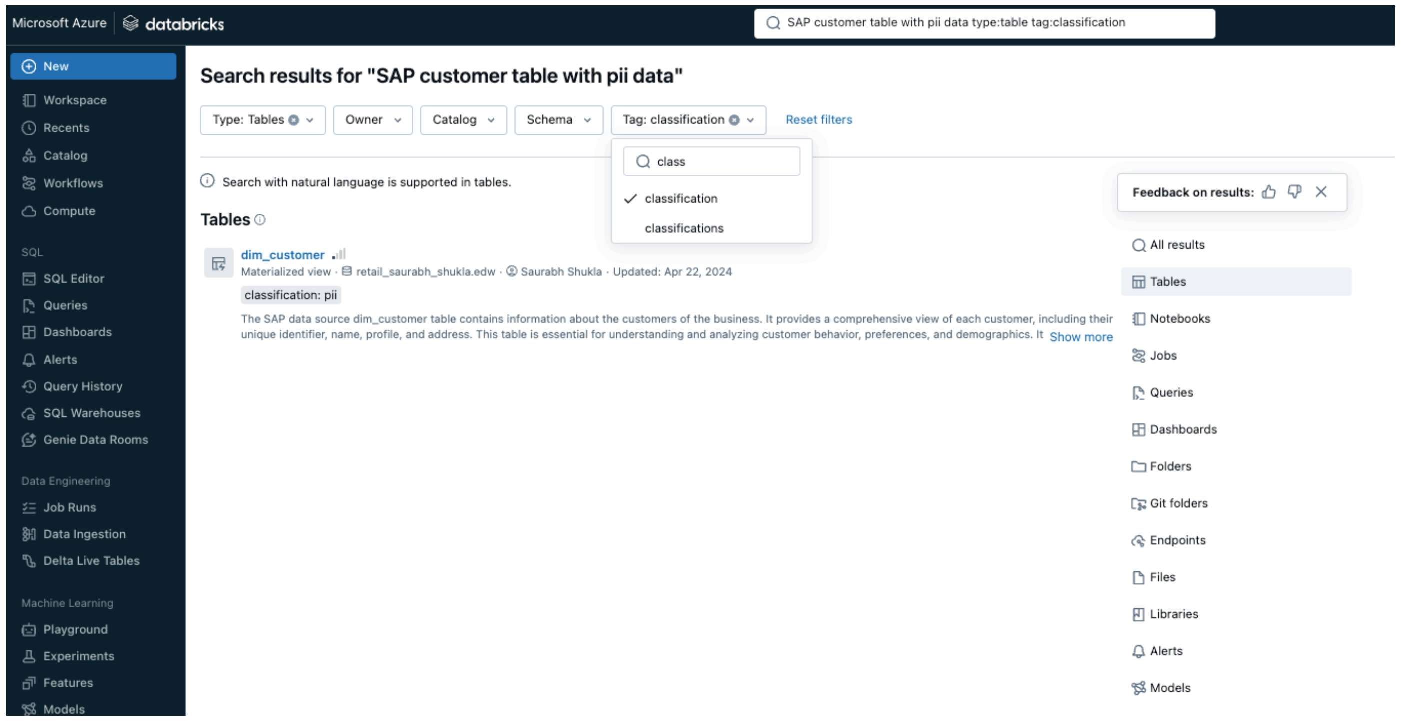
Databricks search also incorporates *knowledge cards*. When search identifies what you're looking for with high confidence, the top search result turns into a knowledge card. These cards provide additional asset metadata for easier identification of relevant resources.

Tags as business taxonomy

Datasets can be challenging to discover and categorize. Tags are additional metadata that can be attached to securable Unity Catalog entities to improve various use cases, including governance, search and discovery of datasets.

To establish a business taxonomy using discovery tags in Databricks, follow these guidelines:

- **Define tagging standards:** Collaborate with stakeholders to define a standardized set of tags that align with business objectives and regulatory requirements
- **Apply tags:** Use the Databricks tagging feature to label data assets manually or automatically based on data ingestion workflows
- **Train users:** Educate data users about how to use tags for searching and managing data, ensuring that everyone understands the tagging system and its benefits
- **Monitor and refine:** Regularly review the tagging system to ensure it meets the evolving needs of the business. Adjust and add new tags to keep the taxonomy relevant and valuable.



SAP customer table with pii data type:table tag:classification

Search results for "SAP customer table with pii data"

Search with natural language is supported in tables.

Tables

dim_customer

Materialized view · retail_saurabh_shukla.edw · Saurabh Shukla · Updated: Apr 22, 2024

classification: pii

The SAP data source dim_customer table contains information about the customers of the business. It provides a comprehensive view of each customer, including their unique identifier, name, profile, and address. This table is essential for understanding and analyzing customer behavior, preferences, and demographics. It Show more

Data lineage as a search enabler

Data lineage is essential to a pragmatic data management and governance strategy. Following are important key points for leveraging lineage as a data search enabler.

- **Tracking data evolution** — Lineage allows users to visualize the journey of data from its origin through various transformations to its final form. This visibility into data evolution helps users search and identify the datasets relevant to their analysis or business needs.
- **Supporting data compliance and audit** — By providing a clear map of where data comes from and how it's processed, lineage helps organizations comply with regulatory requirements. Users can search for specific datasets and understand their provenance, which is crucial for audit trails and compliance reporting.
- **Enabling observability for the data stack** — Lineage contributes to the observability of the data stack by showing how data is used and where it flows. This makes it easier for users to search for data assets and assess their impact on different parts of the system, such as ML models, dashboards and reports.
- **Facilitating impact analysis** — Before making changes to data assets, users can search and review the lineage to understand the downstream effects of those changes. This helps manage risks associated with data modifications and ensures that dependent processes aren't adversely affected.
- **Improving data quality assurance** — Understanding the lineage of data helps in identifying the sources of data quality issues. Users can search for and trace errors back to their origins, which is essential for maintaining the integrity of data pipelines and ensuring the accuracy of analytics.
- **Enhancing collaboration and knowledge sharing** — Lineage provides a shared understanding of data flows, which is valuable for collaboration across data teams. Users can search for data assets and share insights about their lineage, promoting knowledge transfer and alignment on data practices.
- **Integration with other data governance tools** — Lineage information can be exported via REST API and integrated with other data catalogs and governance tools. This allows for a unified view of data lineage across the entire data ecosystem, enhancing searchability and governance across multiple platforms.

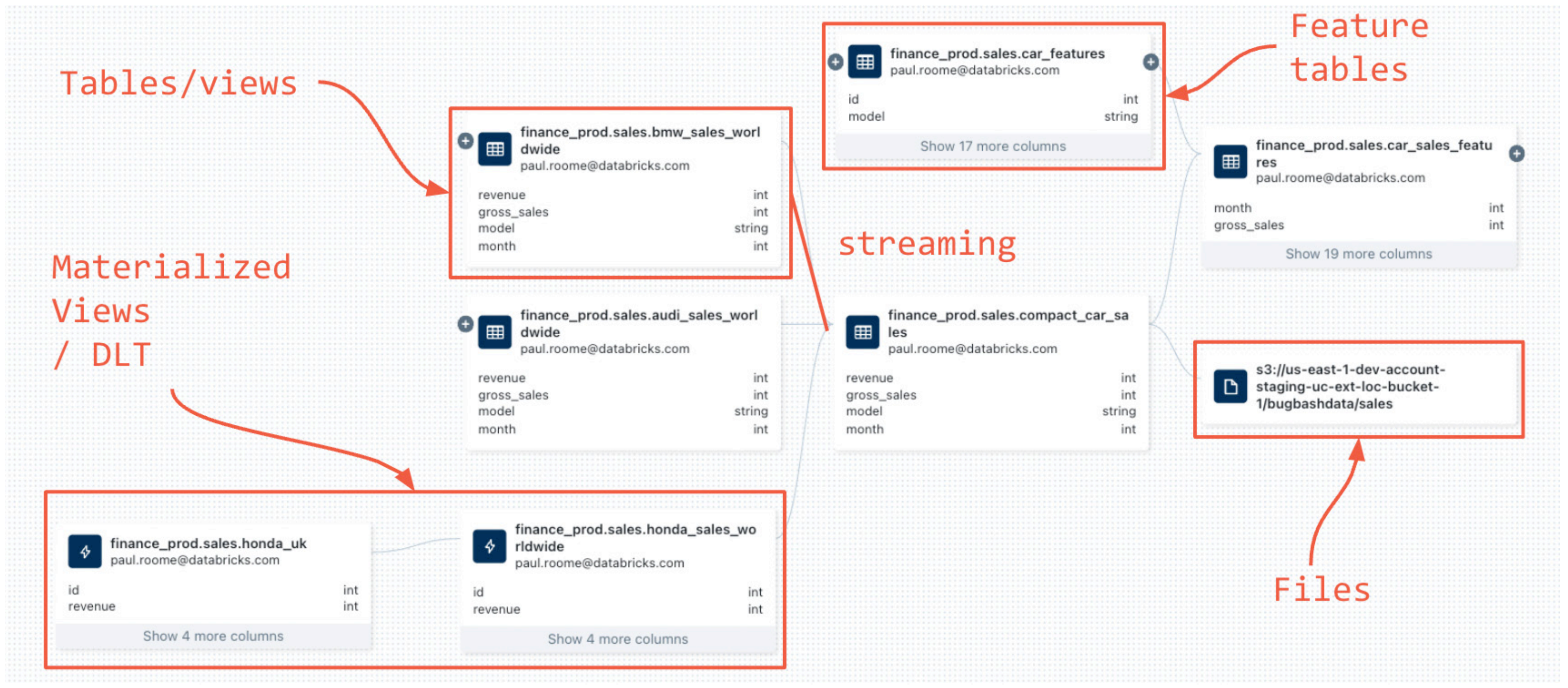

Tables/views
Materialized Views / DLT
streaming
Feature tables
Files
finance_prod.sales.car_features
finance_prod.sales.bmw_sales_worldwide
finance_prod.sales.car_sales_features
finance_prod.sales.audi_sales_worldwide
finance_prod.sales.compact_car_sales
s3://us-east-1-dev-account-staging-uc-ext-loc-bucket-1/bugbashdata/sales
finance_prod.sales.honda_uk
finance_prod.sales.honda_sales_worldwide
paul.roome@databricks.com
Show 17 more columns
Show 19 more columns
Show 4 more columns

Figure: Data lineage for all data and AI assets

GRANTING BROWSE PERMISSION ACROSS THE DATA ESTATE

With browse permission, users can discover data without having read access to the data. A user can view an object's metadata using Catalog Explorer, the schema browser, search results, the lineage graph, information_schema and the REST API.

Tracking and Promoting Data Quality

Tracking and monitoring data quality is essential for all kinds of data pipelines. It's an integral part of a modern and mature data and AI architecture.

Data quality as a practice isn't a one-time check but a continuum across your data and AI pipelines. It's also crucial for machine learning and generative AI in addition to more traditional ETL motions. There's a known maxim in the AI world that goes "Garbage in, garbage out," which refers to the idea that the quality of the data used to develop a model has a direct impact on the quality of the model output. The same principle applies to reporting and other downstream decision support systems.

Databricks Unity Catalog provides a host of features dedicated to ensuring the quality of your pipelines can be accurately measured, monitored, tracked and maintained for any use, from data to features to models.

Databricks Lakehouse Monitoring

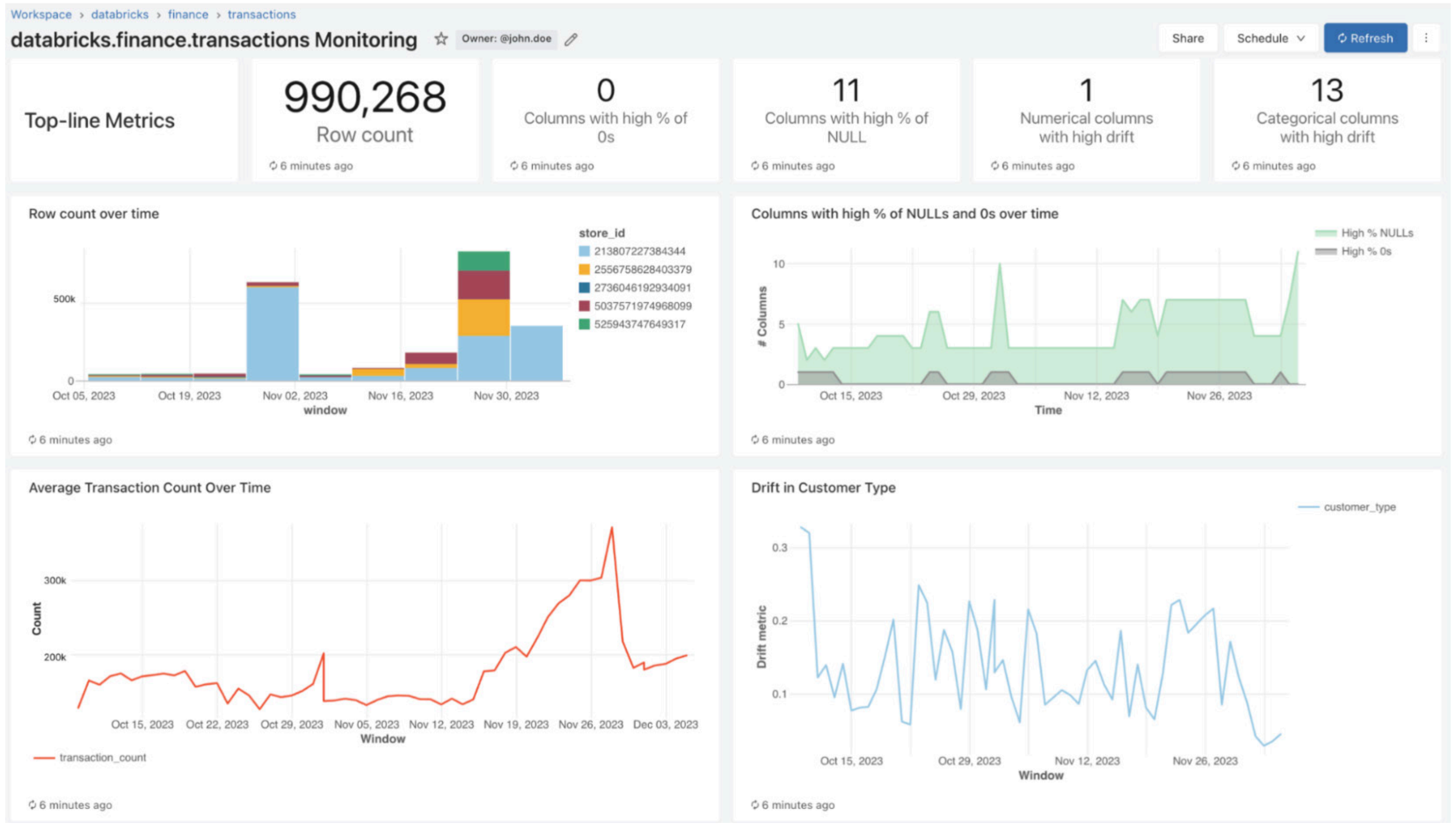
Unity Catalog facilitates data quality tracking and monitoring with Databricks Lakehouse Monitoring. This feature allows you to specify a monitoring endpoint on any given Unity Catalog table or Model Serving endpoint in your environment.

For example, once a monitoring endpoint is configured on a table, Lakehouse Monitoring calculates a set of out-of-the box metrics such as min, max and other distributional metrics, including statistical distribution tests like the Kolmogorov-Smirnov (K-S) test. For a complete list of metrics, you can refer to our Databricks product documentation. Custom metrics of your choice can also be added. All the information is aggregated in a dashboard that's automatically created. This dashboard can be used not only to visualize and track the evolution of these metrics but also to set alerts via the use of Databricks SQL Alerts. For instance, you could set an alert, coupled with a variety of webhook options from email, for Slack notifications to let you know when a certain column in your data is out of range.

Lakehouse Monitoring is a unified data monitoring solution that spans data engineering and ML/AI use cases. This can be particularly useful for tracking the performance of machine learning models and Model Serving endpoints by monitoring inference tables that contain model inputs and predictions.

When changes in your table's data distribution or the performance of the corresponding model are detected, the tables created by Databricks Lakehouse Monitoring can alert you to the change and help you identify the cause. This can be crucial in identifying and addressing *concept drift*, which arises when the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways.

Lakehouse Monitoring also allows you to control the time granularity of observations and set up custom metrics, providing flexibility in how you track and analyze your data and models. For instance, you can use it for tables that contain the request log for a model, with each row being a request, and columns for the timestamp, the model inputs, the corresponding prediction and an optional ground-truth label. Lakehouse Monitoring then compares model performance and data quality metrics across time-based windows of the request log.



Lakehouse Monitoring provides the following benefits for organizations looking to better track, enforce and monitor data quality best practices:

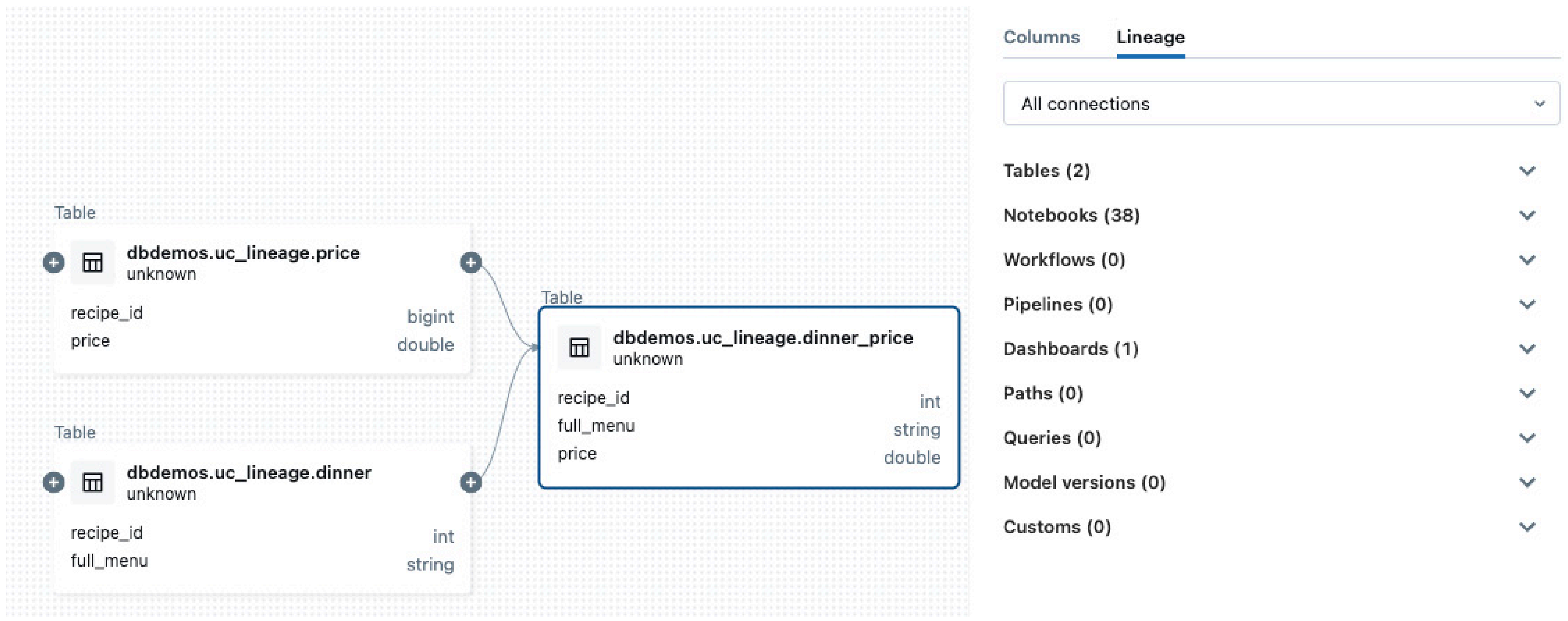
- **Comprehensive monitoring** — Allows you to monitor the statistical properties and quality of all the tables in your account. This includes tracking the performance of machine learning models and Model Serving endpoints by monitoring inference tables that contain model inputs and predictions.
- **Data confidence** — To draw useful insights from your data, you must have confidence in the quality of your data. Lakehouse Monitoring provides quantitative measures that help you track and confirm the quality and consistency of your data over time.
- **Alerts on data changes** — When changes in your table's data distribution or the performance of the corresponding models are detected, the tables created by Databricks Lakehouse Monitoring can alert you to the change and help you identify the cause.
- **Customization** — Databricks Lakehouse Monitoring lets you control the time granularity of observations and set up custom metrics. You can also set slicing expressions to monitor feature subsets of the table in addition to the table as a whole.
- **Quality visualization** — An automatically generated dashboard visualizes data quality for any Delta table in Unity Catalog. All metrics are stored in Delta tables to enable ad hoc analyses, custom visualizations and alerts.
- **Proactive issue detection** — You can proactively discover quality issues before downstream processes are impacted, ensuring that pipelines run smoothly and machine learning models remain effective over time.
- **Serverless** — Lakehouse Monitoring is fully serverless, so you never have to worry about infrastructure or tuning compute configuration.

Lineage

Another crucial feature that Unity Catalog provides to track data quality and facilitate its promotion is lineage. Any operation in Databricks that's performed on a Unity Catalog registered table (managed or external) using a Unity Catalog–enabled cluster is logged. This includes the lineage information.

Lineage within Unity Catalog includes information about upstream and downstream tables. For example, if a table is the product of a merge between two other tables, this information is captured and queryable, as is the information about additional artifacts where tables may be used like notebooks, workflows and dashboards. Lineage extends beyond the table level. Unity Catalog is actually able to provide lineage information up to the column level, which makes it extremely powerful in terms of identifying data quality issues with high granularity.

The lineage information captured by Unity Catalog is available via the Databricks UI, but it's also directly available through Databricks system tables, namely through the table and column lineage tables. For more information, you can refer to the system tables documentation. The information is also programmatically available via the API.



Lineage plays a crucial role in AI development, providing a comprehensive view of the data journey from its original source through all its transformations to the point it's used in a machine learning model. This visibility is essential for understanding the impact of data quality on model performance, as well as for troubleshooting issues that may arise during the model development process.

Lineage information within Databricks can help track different versions of a model, allowing for comparisons and evaluations of their performance over time. This can be particularly useful in identifying when and why a model's performance may have changed, which can inform future model development and refinement.

A powerful tool for auditing purposes, lineage provides a transparent record of the data's journey, which can be crucial for regulatory compliance, especially in industries where explainability and accountability are paramount.

Maintaining data quality is a continuous process, and lineage can support this by highlighting areas where data quality may be compromised, such as in the data ingestion or transformation stages. By identifying these areas, teams can take proactive steps to improve data quality, thereby enhancing the overall performance and reliability of their AI systems.

Lineage captured through Unity Catalog affords organizations the following benefits with regards to tracking and promoting data quality:

- **End-to-end lineage tracking** — Unity Catalog captures lineage data across all lakehouse entities, including notebooks, workflows, dashboards, queries, tables and machine learning models. This end-to-end visibility helps you understand how data assets are created and used across all languages.
- **Data governance and discovery** — Lineage data is crucial for managing change, ensuring data quality and implementing data governance in an organization. It aids in data discovery by providing a clear picture of the data's origin and its transformation journey.
- **Problem identification and change management** — By tracking the lineage of data, you can identify issues at their source and manage changes effectively. This allows for impact analysis, helping you understand the potential consequences of changes in your data.
- **Integration with other features** — Lineage data can be used to support intelligent actions in other product features, such as data insights or data quality alerting. It can also be integrated with partner catalog vendors for seamless lineage tracking.
- **Access and permissions** — Lineage data is aggregated across all workspaces attached to a Unity Catalog metastore, meaning that lineage captured in one workspace is visible in any other workspace sharing that metastore. However, users must still have the correct permissions to view the lineage data.
- **Data quality platform integration** — Catalog Explorer can display health/quality status and violations along the lineage chain to assist in root cause analysis. This requires all tables in the lineage chain to be monitored through Lakehouse Monitoring.