Handling Sensitive Data with Generative AI and other Databricks Workloads

Dr. Alan L. Dennis, Avanade

Overview of the Document

This document aims to document and address common challenges with sensitive data using the Databricks platform. We provide an overview of the problem, discuss the challenges of using synthetic data, and provide definitions to remove ambiguity in this context. We discuss the possible solutions and present a recommended solution. The pillars of the solution are discussed, along with a detailed discussion of the solution's elements. A visual representation of the outline of the paper is presented in Figure 1.

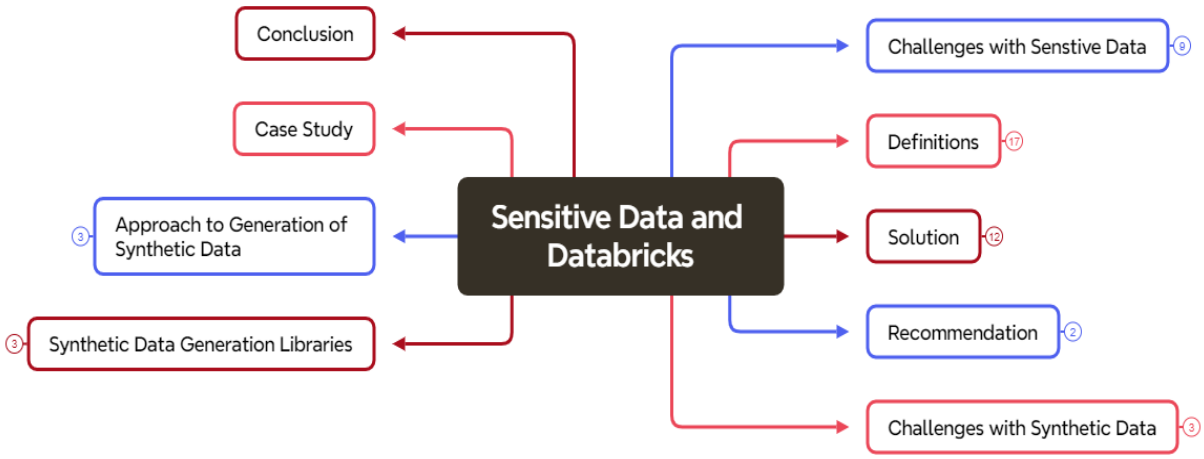


Figure 1: Mind Map of Sensitive Data Approaches with Databricks (read clockwise)

Challenges with Sensitive Data

There was a time when a student's grades were posted outside their instructor's office by listing the student's identifier and grade. Often, that student identifier was also their social security number. As you know, the world has moved on. Today, we approach privacy from a very different perspective. Organizations must balance the risk of data exfiltration with the desire to perform Lakehouse-based analytics.

Overview of the Problem

There are significant penalties for failure to control data adequately. They range from punitive financial penalties to brand devaluation. The basic problem is that some data requires specialized handling. The nature of sensitive data is explored in this section, followed by the challenges associated with utilizing offshore resources, common challenges, and typical solutions.

Sensitive Data

Sensitive data requires special architectures, processing, and tools. When dealing with sensitive data, the cost of failure is very high. While sensitive data fundamentally differs from other data types, it must have additional safeguards.

Offshore Resources

Offshore resources are attractive for many projects because of their lower cost and because they operate in a different time zone, enabling work to continue around the clock. However, onboarding offshore resources in highly restricted environments can be challenging. This friction is caused by data handling restrictions and statutory compliance.

In larger software construction efforts, it is common to have a mixture of on and offshore team members. This blend allows for more favorable billing rates, potentially increasing productivity and enabling follow-the-sun development. However, many organizations often struggle to onboard offshore resources due to regulatory concerns. Often, data movement must be restricted to ensure that the data does not leave a geographic location.

Common Challenges

The challenges associated with processing sensitive data span multiple industries. Healthcare is often at the top of people's minds due to high-profile regulations, such as the Health Insurance Portability and Accountability Act (HIPAA). Other governance controls exist, such as the California Confidentiality of Medical Information Act (CMIA) and the California Privacy Rights Act (CPRA). Other industries, such as higher education, have governance with similar ends, such as the Family Educational Rights and Privacy Act (FERPA). These controls protect sensitive data and carry significant penalties to ensure regulated organizations remain compliant.

Data has a lifecycle. It starts with the data's creation and typically ends at deletion. Some regulations, such as the General Data Protection Regulation (GDPR), may require the propagated deletion of data in response to an individual's request. Historically, this type of processing has been a challenge for data platforms and systems.

Identification of sensitive data may be challenging. Certain types of data are explicitly named as sensitive data in regulations. Often, the field names in a database may make it challenging to determine if a particular field contains sensitive data. On an ongoing basis, examination of the contents is often required to identify and address sensitive information.

Certain types of data, such as unstructured data (such as healthcare provider notes), are particularly difficult to handle. Later in this paper, we discuss approaches to reducing data exfiltration risk. Free-form text is challenging in that identifying sensitive data is difficult, as are techniques to enable the data to remain helpful while making it difficult to identify the individual.

Cross-organization collaboration and data sharing is often difficult. It is common for organizations to engage with partners to perform data processing. The need to work securely with external organizations is important enough that Databricks created Clean Rooms, which will be discussed later in this document.

Generative Artificial Intelligence (GenAI) brings its own set of sensitive data issues. For GenAI to be effective (as with other types of learning), it must be exposed to Actual data. This means that the security model governing Actual data must allow for the use of the data while ensuring that a GenAI model is not exposed to sensitive data. Considerable research is being performed to address the goal of

benefiting from the revolutionary innovation that GenAI brings while minimizing the introduction of novel vulnerabilities [1].

We have discussed many of the challenges associated with sensitive data. Next, we will discuss some common approaches to address these impediments.

Common Solutions

Software engineering has long used a multiple-environment approach. The idea of this approach is to have environments for purposes. Production environments are where tested and validated activities are executed while closely monitored. The goal of software engineering activities is to reach production and then execute as expected without failure once having reached production. We will talk more about environments later, but for now, it is common to see a three-environment configuration comprising development (Dev), quality assurance (QA), and production (Prod).

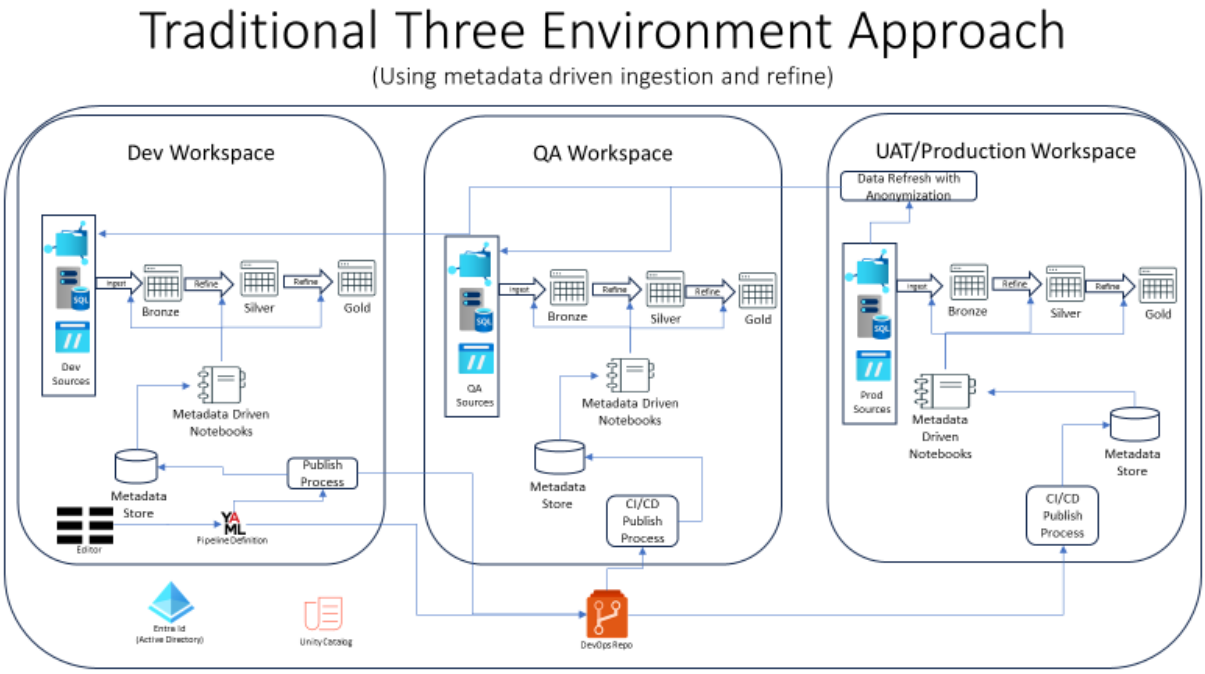


Figure 2: Traditional Three-Environment Approach

Ideally, each environment has its distinct version of source data. However, this is often not the case. Organizations often copy data from production to development and QA to address this gap, as demonstrated in Figure 2. While this copy occurs, operations are often performed to remove or modify sensitive data. For the purposes of this paper, we call this Sensitive Data Protection Mechanism (see definition). Additionally, it is common for sampling to occur, reducing the number of records in QA and Dev to a fraction of the records in the production environment. This aims to reduce cost and processing time.

Generally, access to an environment is based on the role an individual has been assigned and is performing. For example, a tester will likely have access to QA and Prod but not Dev. Likewise, a

developer will have access to Dev but likely not to QA or Prod. Ideally, access control should be managed by group membership; unfortunately, teams sometimes manage access at an individual level.

Access to data assets, such as files and tables, may be governed by group membership. The idea is that an individual can see a certain table but should not be able to see all tables unless there is a business need. Often, changes to access are relatively static. Once an individual has gained access to a data asset, it may be rare for review if that individual or group should continue to have access. Restrictions that are more granular than table-level are not common in many of these situations. In part because big data processing platforms historically lacked fine-grain controls (FGAC).

Some organizations may track the access of data assets by their employees. The idea is to monitor the data assets being accessed and determine if the individual has a business reason for the activity. For example, does the employee have a business reason to look at the medical records of a specific individual? This approach's challenge is that it may be difficult to determine if an activity has a business purpose during access, so the logs are only useful during remediation, not providing prevention.

Some organizations utilize remote desktop connectivity to a virtual desktop infrastructure (VDI). The goal is to restrict behaviors closer to the data, such as accessing email, cloud storage, and other egress points. The challenge with these approaches is that as the environment becomes more secure, it often makes the users of the environment less productive. For example, many data engineers rely on websites for code snippets. While not allowing access to the internet may seem like a way to secure data, it may result in significant decreases in productivity and software and, in turn, data quality. Sometimes, this approach includes physical controls, resulting in an environment reminiscent of a sensitive compartmented information facility (SCIF). These facilities do not allow employees to bring cell phones and other electronics to reduce an employee's ability to mishandle information. Related to this approach is Databricks' Clean Rooms [2], which provides a security-first approach to data handling.

We have discussed the problem of handling secure data. Next, we will introduce concise definitions of terms. Having clear definitions is important when discussing concepts.

Definitions

Words often have overloaded meaning. For example, if someone says, "We have a production issue," it may be unclear exactly what that means. Is the problem related to a production environment or production data? We introduce specific terminology for the remainder of this paper to address term overloading.

Actual Data

The term Actual is used to refer to data that came from a production system. Actual data may contain sensitive data (PHI, PII, etc.). Data derived from Actual data is still Actual, even if sampled, masked, anonymized, or pseudonymized.

Sensitive Data

For the purposes of this paper, sensitive data refers to data that is regulated and deemed identifying. This includes Personally Identifiable Information (PII) - Information that can be used to identify an individual, such as their name, address, phone number, email address, social security number, or other

unique identifier [3]. Sensitive data will always be Actual data, while Actual data may or may not be Sensitive.

TYPE OF FIELD	DESCRIPTION
NAME	The first and last name of an individual
PERSONAL IDENTIFICATION NUMBERS	Social Security Number, passport, Driver's License, taxpayer ID number, patient identification number, financial account number, or credit card number
PERSONAL ADDRESS INFORMATION	Street or email address
PERSONAL TELEPHONE NUMBERS	Phone numbers that can be used to reach the individual
PERSONAL CHARACTERISTICS	Photographic images, fingerprints, handwriting
BIOMETRIC DATA	Retina Scans, voice signatures, facial geometry
INFORMATION IDENTIFYING PERSONALLY OWNED PROPERTY	Vehicle Identification Number (VIN) or title number
ASSET INFORMATION	Static Internet Protocol (IP) or Media Access Control (MAC) addresses

Table 1: Types of PII data

Additionally, Protected Health Information (PHI) is also referred to as sensitive data [4]. PHI is information about an individual's health status, provision of health care, or payment for health care. Examples of PHI data are in Table 2.

TYPE OF FIELD	DESCRIPTION
NAME	The first and last name of an individual
ADDRESS	Geographic regions that are smaller than a state, street address, city, county, and zip code
DATES	Birth, death, admission, and discharge
TELEPHONE AND FAX NUMBERS	Phone numbers that can be used to reach the individual
EMAIL ADDRESS	An individual's email address
SOCIAL SECURITY NUMBER	Identifier issued by the Social Security Administration
MEDICAL RECORD NUMBER	A key used by some medical record administration systems
HEALTH PLAN BENEFICIARY NUMBER	A key used by health insurance or similar services
ACCOUNT NUMBER	A key often used for financial purposes
CERTIFICATE OR LICENSE NUMBER	
VEHICLE IDENTIFIERS AND SERIAL NUMBERS	Includes License Plates

DEVICE IDENTIFIERS AND SERIAL NUMBERS	Includes Medium Access Control (MAC) Address and Electronic Serial Number (ESN) number
WEB URL	A link to an individual's web page or resources
INTERNET PROTOCOL (IP) ADDRESS	A potentially static unique number related to an individual's web access
FINGER OR VOICE PRINT	Includes other biometric elements
PHOTOGRAPHIC IMAGES	Not restricted to images of the face
ANY OTHER CHARACTERISTIC THAT COULD UNIQUELY IDENTIFY THE INDIVIDUAL	

Table 2: Types of PHI data

There are many other ways of discussing sensitive data [5]. For our purposes, we will simplify the taxonomy to data that is sensitive and data that is not.

Sensitive Data Protection Mechanism

Having identified sensitive data, we need a way to handle it. The purpose of the mechanism is to handle sensitive data for compliance and to reduce potential exposure. There is a high-level decision: do you want to retrieve the original value after applying the mechanism? If you do, then you want to use pseudonymization techniques. Otherwise, use anonymization.

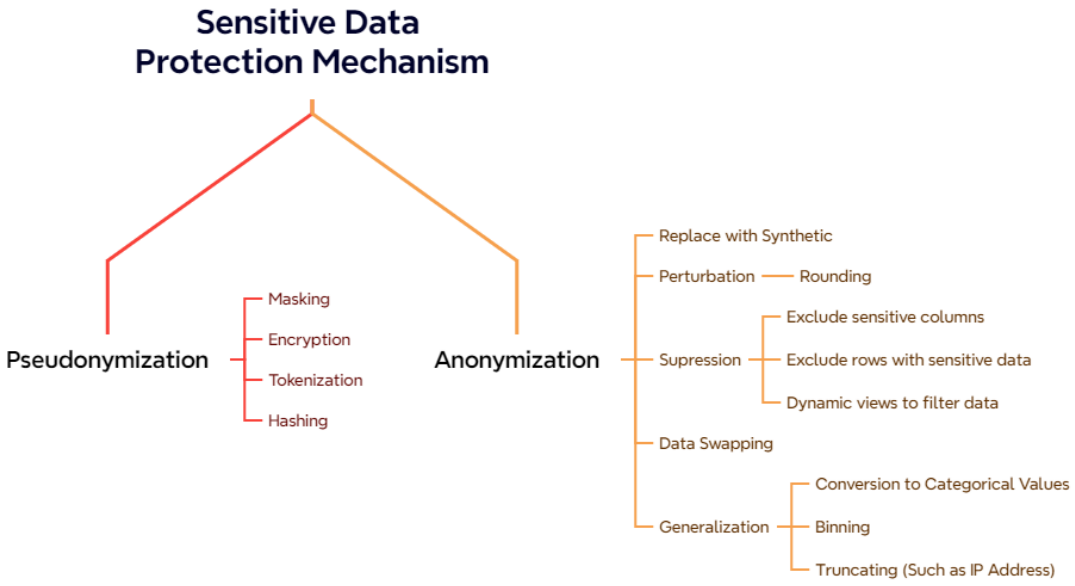


Figure 3: Sensitive Data Protection Mechanism

We will cover many of these approaches in the following sections. The contents of Figure 3 and the associated ontology are purposeful simplifications of the field. The goal is to keep things simple.

Synthetic

Synthetic data is created algorithmically without using Actual data. It may be based on characteristics of Actual, such as range, standard deviation, number of categories or distinct values, etc. Because of the randomness of creating synthetic data, it is unsuitable for data analysis, such as Machine Learning (ML) or Artificial intelligence (AI) workloads.

When dealing with synthetic data, it is important that it is never combined with actual data unless it is being used for anonymization. Two simple rules can reduce the chance of unintentional commingling: Synthetic data should never enter an Actual environment, and Actual data should never enter a Synthetic environment.

There are many methods to generate Synthetic data [1]. In Databricks, a tried-and-true approach is to use the Databricks Labs Data Generator [6]. This is an active research area and a topic worthy of its own paper.

Pseudonymization

Pseudonymization is the process of changing sensitive data so that it is not identifiable but can be re-identified later. The process is performed at a record level. The data is still considered personal data by the General Data Protection Regulation (GDPR) [7]. Common approaches to pseudonymization include tokenization, hashing, and masking.

Hashing

A hash function maps data from one set of values to another. It often uses techniques such as a Secure Hashing Algorithm (SHA) or some other hash function. Often, salt values are added and stored in a secret store (such as a key vault). There are several open-source systems available [8].

Tokenization

The idea of tokenization is to replace sensitive values with a key used in a token vault that houses the sensitive data. This approach is useful for Artificial Intelligence or Machine Learning.

Masking

Data masking involves replacing the results of a query with a value that obstructs the Actual value. It may be a complete mask, meaning that none of the actual data is displayed, or partial. An example of a partial mask would be to show the last four digits of a Social Security number, such as ***-**-1234. Note that partial masking may enable deidentification through triangulation and other big data processing techniques by combining the masked dataset with additional datasets. In the section entitled *Fine Grained Permissions via Tags*, we discuss how to leverage Databricks' ability to mask columns and provide high level pseudocode to implement a simplistic version of attribute-based access control.

Anonymized

Actual data that has been manipulated to be unidentifiable. Data is irreversibly modified so that the identifying information cannot be recovered.

Anonymization mechanisms include:

- Synthetic data generation
- Data suppression

- Exclude sensitive columns from the return of a dataset
- Exclude rows with sensitive data
- Use dynamic views to access data

- Generalization
  - Categorical
  - Binning
  - Truncating IP addresses
  - Rounding

This is not intended to be an exhaustive list. Likely there are far more mechanisms to anonymize data.

Software Engineering Environments

We must distinguish between data and software engineering environments when considering environments. Data includes Actual data, including data that might have been sampled to reduce the overall size. Software engineering environments are locations where work is performed. A single software engineering environment may use multiple types of data, depending on the workload. Databricks provides robust continuous integration and continuous deployment (CI/CD) capabilities, enabling the promotion of artifacts across environments. Additionally, MLflow can be used to manage the machine learning (ML) lifecycle, and other types of activities across environments.

We often refer to higher and lower environments when discussing software engineering environments. This comes from the fact that we promote code from one environment to another. For example, we often promote code from Development to Quality Assurance. In this scenario, Quality Assurance is a higher environment than Development. Production is viewed as the highest environment in the ordering, as shown in Figure 4.

Figure 4: Ordering of Software Engineering Environments

Development

Development environments (Dev) are used to construct data engineering activities and access controls. For example, a column filtering function should be developed and tested in Dev before being applied in high environments. Dev Environments are used to perform data engineering development. As a best practice, Dev should never utilize Sensitive data. Ideally, Synthetic data should be used in Dev as it reduces unnecessary risk. As long as the Synthetic data is relatively representative of Actual, data engineering workloads should be able to be performed. Other workloads like machine learning and artificial intelligence development often require pseudonymized data.

Quality Assurance

Quality Assurance (QA) can utilize Synthetic or Actual (anonymized) data. It is used to validate development efforts and to ensure unit and functional tests pass.

System Integration Test

System Integration Test (SIT) ensures that various solution parts communicate correctly. Depending on the type and scale of the development being performed, it may not be required. Like QA, it may contain Synthetic Actual (anonymized) data. SIT reduces the chance that system elements do not work correctly when combined in production.

User Acceptance Test

User Acceptance Test (UAT) often contains Actual (anonymized) data to validate that the solution meets the technical and functional requirements. It may be a distinct environment or an activity that is performed in an environment.

Production

Production (Prod) uses Actual data captured from source systems. It is an environment with controls to ensure changes are not made without proper processing being followed. Generally, the only time software engineering changes are made directly in a Prod environment is when that environment is experiencing a defect, often called a fire-fighting experience. Often, software engineering teams do not have access to Prod environments.

Data Science and Artificial Intelligence Sandbox

Data Science (DS) and Artificial Intelligence (AI) bring unique challenges when dealing with sensitive data. Generally, they require Actual data. One approach to this challenge is to create a workspace that is not considered production but is used to access actual data. This workspace is where a DS or AI practitioner will create their experiments, models, and so on. The output of some of these efforts may enter MLOps and LLMOps workflows. Additionally, DS or AI users may need to create and persist tables for their purpose, which often will contain sensitive data. To address these challenges, we present an approach in Figure 5. The idea is that all users of this workspace would have a unique schema in a Sandbox catalog. Their schema is private, with only administrators having additional access.

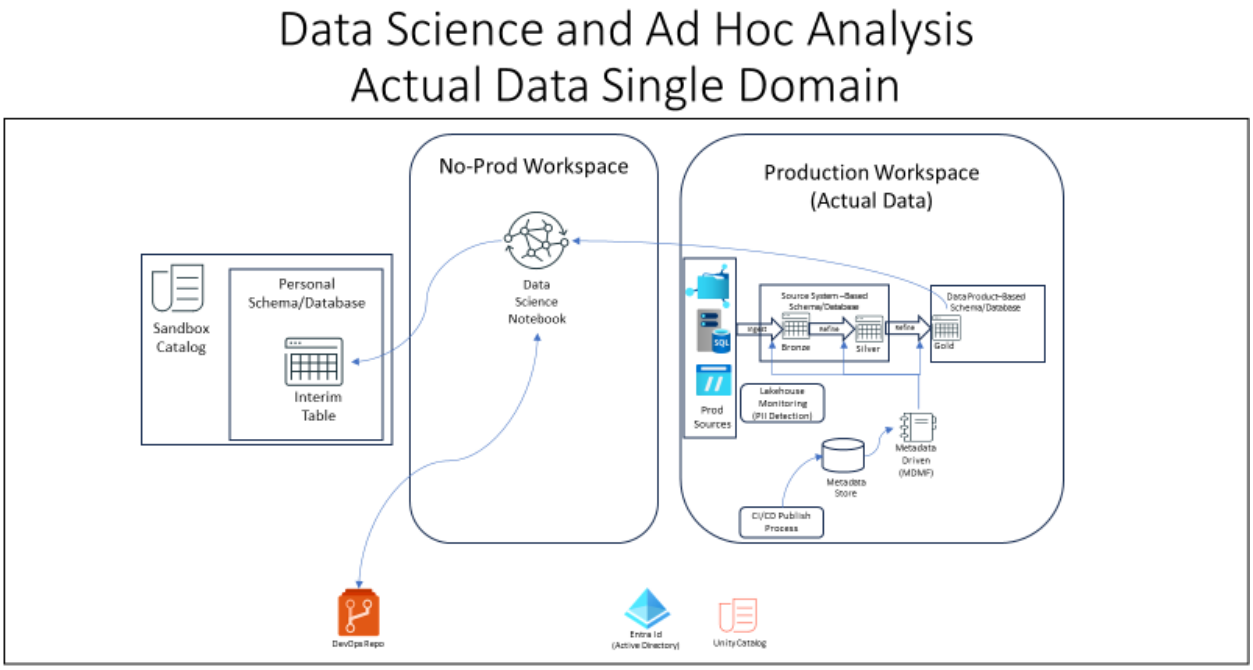


Figure 5: DS, AI, and Ad Hoc Analysis Workspace

To address the challenges associated with Ai and ML, Databricks has published a whitepaper [9]. The paper helps organizations assess and address risks associated with each of the steps in AI and ML workstreams.

In this section, we have discussed places where individuals work. We discussed traditional data engineering workload activities, AI, DS, and ad hoc analysis. In the next section, we discuss the challenges associated with using synthetic data.

In this section, we presented a set of definitions that will be used throughout this paper. In the next section, we propose a solution to the challenges associated with sensitive data.

Sensitive Data Handling Decision

We have discussed the various approaches to handling sensitive data. A fundamental question relates to how and where sensitive data is stored. Figure 6 contains a representation of a high-level decision making process.

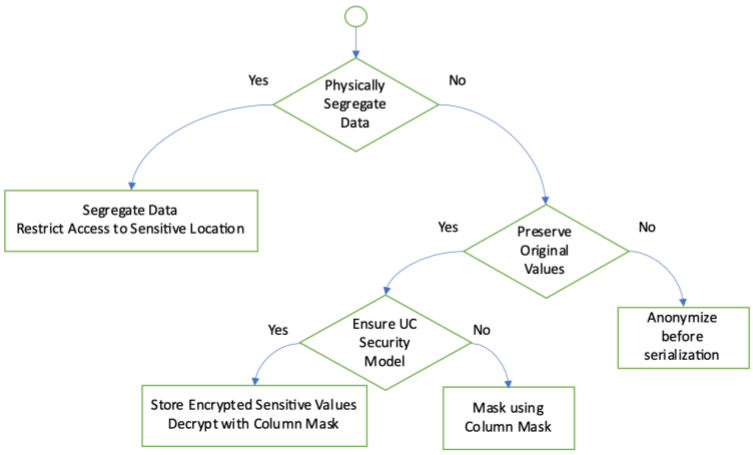


Figure 6: Sensitive data decisions

One choice is to store all sensitive fields in a different set of tables. The idea is to deconstruct a table containing sensitive data into table two or more tables. One of the tables would contain non-sensitive data and a key to join with the sensitive-data-containing tables. The benefit of this approach is that the sensitive data can be physically stored in a different location. If that approach is not desired, the next key decision is to preserve the original values. If preservation of those sensitive values is required, then a form of pseudonymization is required (such as hashing, tokenization, etc.). The final decision presented is if we want to store the sensitive values in such a way that the Unity Catalog security model is enforced, regardless of if Databricks is used to access the data. This is relevant when interacting with other data platforms or applications. It is not a best practice to enable access directly to the physical files.

Solution

There are many possible solutions to the problem we have outlined. We will present two variations of a solution. The variation accommodates the situation where shore development is being done in a distinct and segregated environment. We start this discussion by considering the pillars of the solution. Without these technologies, these solutions would not be possible. Then, we discuss the alternatives.

Pillars of the Solution

This solution relies on Databricks Unity Catalog, Azure's Entra Id (formerly Azure Active Directory), and Azure DevOps repositories. We will discuss the contributions of each element to our solution and provide details on the features and functions being utilized.

Databricks Unity Catalog

What appears to be a minor change sometimes is revolutionary. The adoption of Unity Catalog is one of those changes. While introducing a catalog level to Databrick's organizational structure seems like a minor change, Unity Catalog offers considerably more benefits to organizations. Not only does it provide better organizational capabilities, but it can also be leveraged to enable previously impossible things.

Groups

It has been consistently shown that using a group-based access control mechanism is a good way to manage access and permissions [10, 11]. Using System for Cross-domain Identity Management (SCIM), users and groups for Entra Id (Azure Active Directory) can be used to secure data assets.

Tags

Unity Catalog supports applying tags to catalogs, schemas, tables, volumes, views, columns, and registered models [12]. Following a familiar pattern, tags have a key and value. The key can be up to 255 characters, while the value can be 1000 characters. A taggable item can have up to 20 tags associated with it. We will discuss tags in greater detail in the section on Fine Grained Permissions via Tags.

Three Level Organization

Databricks' Unity Catalog brings three levels of organization. This replaces the Hive Metastore's two-level system. The addition of the catalog level enables more ways to organize data. There are multiple elements to organize, as shown in Figure 7. We present an approach to organizing data assets, but we concede there are multiple strategies, with the best approach differing by situation.

One key element to use during organization is the source of the data. This is a high-level value item and should be used to identify synthetic data instead of Actual data. There are alternatives to including this classification in the organization, such as using Unity Catalog Tags. Still, given the impact of incorrectly denoting data, we include it in the organizational approach. The values for this are Actual or Synthetic. There are additional variations, such as Actual data that has been pseudorandomized before storage.

Previously, we discussed software engineering environments, such as production and development. As these environments might have purpose-built data, we include the designation in the organization. Other workloads, such as Data Science exploration or Machine Learning experiments, may have their environment designation.

The designation of the Medallion zone (Bronze/Silver/Gold) is important as it tells us what has been done to data for it to be in that zone. For example, we know that Silver data has had duplicates removed and business rules applied and can be used for business purposes without worry.

The source system is an important designation before data is in the Gold zone. It tells us about the high-level original system from which the data came. Once we begin performing Silver-to-Silver and Silver-to-Gold transformations, we often have data from multiple systems, requiring us to utilize a different naming convention.

The most specific information we have is the name of the digital asset. Often, this is a Table name or a detailed name related to the origin of the data. For example, we often have Accounts and Orders tables.

Knowing the source Table name is important when dealing with Bronze and Silver items, as we often use them to validate ingestion and refinement. When we move to Silver-to-silver-produced data items (intermediate transformations), the name is more important as a way to reduce the chances of duplicate tables being produced. When dealing with Gold items, the name should relate to the business purpose for which they were constructed.

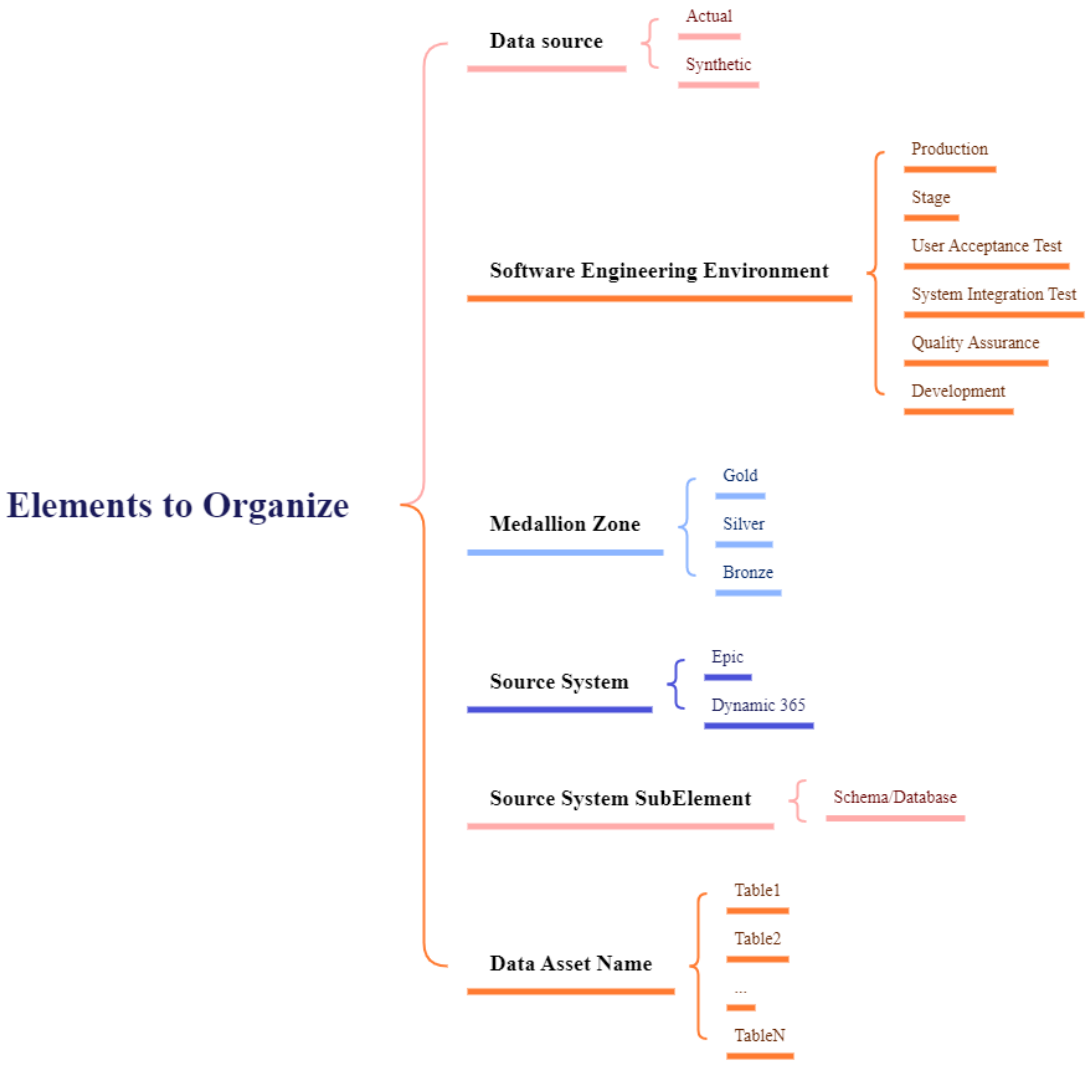


Figure 7: Elements used during data asset organization

We have discussed the various elements that need to be part of an organizational approach, and now we will present one way.

Organization Approaches

There are multiple ways to utilize a three-level namespace. Multiple chapters, if not books have been written on the topic. What we present here should serve as a starting point for your organization's journey, not something to be viewed as the only solution. We present a simple approach in Figure 8.

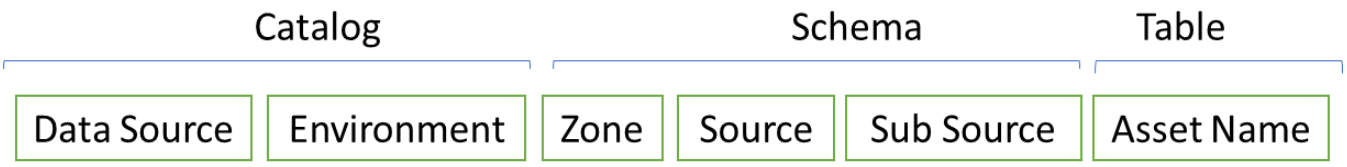


Figure 8: Simple Ingestion-based Organizational Mapping

We combine together the elements that make up Catalog, Schema, and Table using an underscore, as shown in Figure 9. Then, we can combine using the traditional format, `Catalog`.`Schema`.`Table`. Note that this approach is appropriate when handling Bronze data and Silver data derived directly from Bronze data (not intermediate silver to silver transformations).

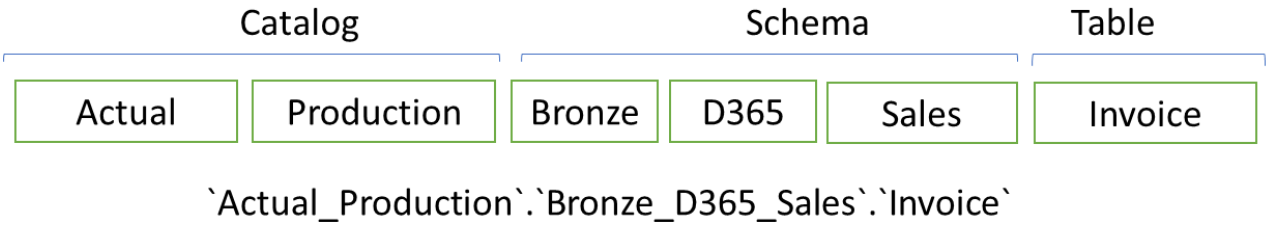


Figure 9: Example of using mapping

Once data has been cleansed and de-duplicated, the next step of processing often involves using intermediate tables. These tables are often populated using Silver to Silver transformations. For intermediate Silver and Gold data we recommend a slightly different organizational approach, as presented in Figure 10.

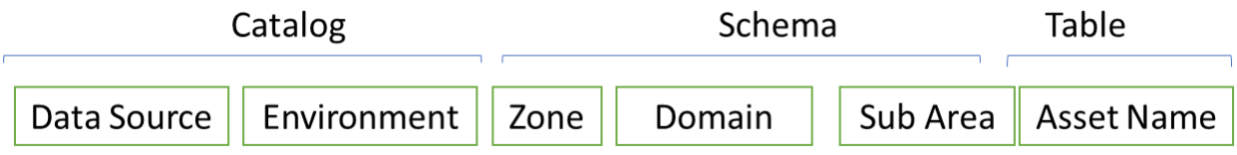


Figure 10: Organization based on Data Products

In this case, we replace Source and Sub Source elements with Domain and Sub Area. Examples of Domain include intermediate or data product. For example, we may want to group together intermediate tables related to the customer entity, as shown in Figure 11.

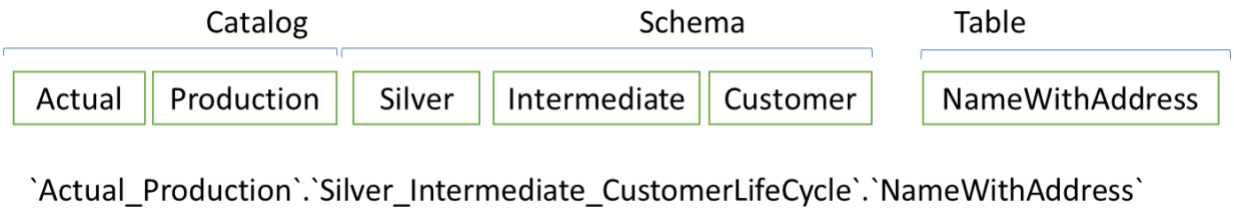


Figure 11: Organization based on data product creation

Revisiting the Data Science and Artificial Intelligence Sandbox discussed earlier, we can organize cross-organizational groups in a similar fashion, as shown in Figure 12.

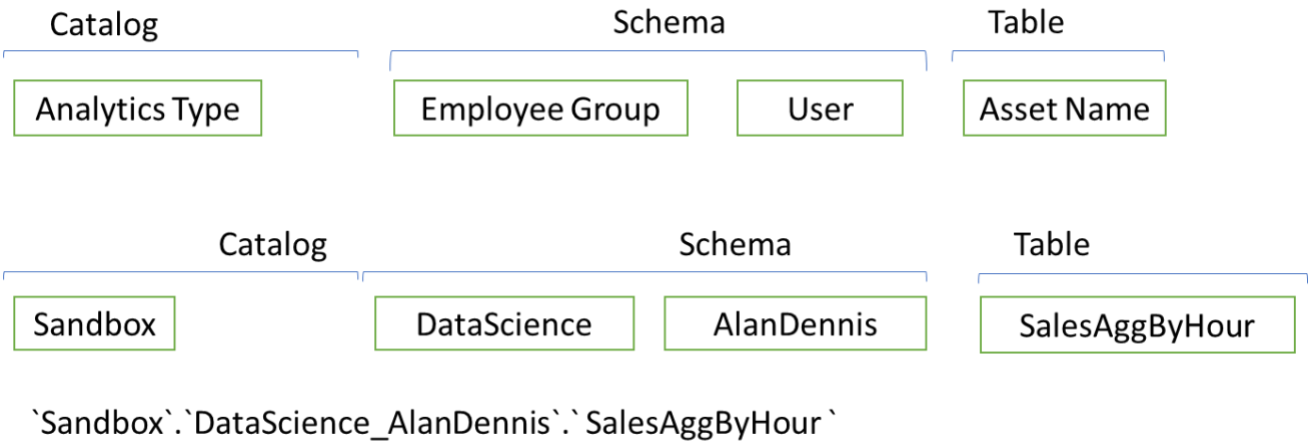


Figure 12: Cross-organization groups

While there are many opinions on how things should be organized, it is important that your organization establishes a standard that is strictly adhered to. While there are other means to find the purview and lineage of data, it is best that information is transparently and consistently communicated.

System and Environment Organization

It is important to understand that while we include the environment in the catalog name, it does not mean that only production systems/workspaces can access that data. The general approach should be that the owning workspace should create and maintain the data while other environments can read the data. For example, it is not uncommon for a data engineer in a development workspace to need to validate a value in a production catalog as part of a defect investigation.

Permission via Grants on Catalogs, Schemas, and Tables

The permissions model in Unity Catalog relies on groups and users being granted access. These rules can be used to create complex access control patterns. An important step in this process is to plan your organization's data isolation model [13]. The basic idea is to ensure that users only access data via a role grant. Ideally, these systems should have a form of automated request and approval via a workflow-based system. For example, a user who needs access to a particular table should create a request routed to that item's data steward for approval. This process should also ensure that there is a time period associated with the access. Typically, grants should be based on roles that users are performing and should be adjusted when a user takes on a new role or leaves the organization. Access control at a table level is adequate for many situations; however, there are times when a finer-grained approach is required. For that, we will discuss a simplified attribute-based access control solution in the next section.

Fine Grained Permissions via Tags

While Databricks works to complete an attribute-based access control (ABAC) solution, we can use Unity Catalog's Tags to lay the groundwork for the eventual solution. The approach is to apply tags to columns that contain sensitive data. For now, we treat PHI and PII as generic sensitive items. The tag's value is the name of the group the current user must be in to see the sensitive values. Keep in mind that tagged items are determined using INFORMATION_SCHEMA tables. Tags can be combined with Column masks using a code generation and programmatic application approach. The content presented here should be viewed as a starting point to implement your solution.

Columns containing Sensitive (PII, PHI, etc.) have the following tags:

- `s_groupname` – the name of the AD Group that the current member must be in to see the column
- If a tag exists on CATALOG or SCHEMA with the name "environment", it is prefixed to `s_groupname`
- `s_filter` – if user is not part of expanded `s_groupname`, then if NULL, column filter returns null instead of value; otherwise, the contents of `s_filter` are used (ex: `***-**-****`)
- Once Tags are applied or changed, code is executed that processes all column mask operations

Pseudocode

- Determine if the environment is set (check Catalog and Schema)
- For each table in catalog & schema of interest that has column tags
    - Check if a Mask already exists in `Information_Schema.column_masks` for the table and column
        - If so, remove it and remove the associated SQL UDF Function
    - Create an SQL UDF function –
        - based on environment, catalog, schema, table, and column that checks for membership of the user in the group named by (environment + `s_groupname`'s value)
        - If the user is not in the group
            - return NULL if `s_filter` is empty, not present, or contains "NULL".

- Otherwise, convert the value of `s_filter` to the data type of the column and return it.
  - Apply the SQL UDF Function as a Column Filter
  - Optionally, create a view that excludes the columns that are tagged with `s_groupname` and the tags contain values.

Microsoft Entra Id (Azure Active Directory)

Microsoft Entra Id, formerly known as Azure Active Directory, is a significant part of the solution discussed here. It provides identity and access management for Azure Databricks' solutions. While Entra ID provides many features and functions, we are primarily concerned with its user and group management capabilities.

As discussed earlier, users should never be granted access to data assets directly. Groups should be nested to reduce the number of grants required. Organize groups to provide granular controls, while minimizing group management. This will require a detailed design and take considerable effort to reach consensus within an organization.

Often, the group organization follows the organization's functional organization. There are challenges with this approach, such as the effort required to adopt new organizational structures when a reorganization occurs.

Azure DevOps Repositories

One of the key elements of modern software engineering is the utilization of automation during development and code promotion. We recommend automating your continuous integration and continuous delivery (CI/CD) processes [14]. Ensure metadata, security related code, and code related to fine grained security control are developed and promoted following SDLC principles.

Databricks Git folders, an updated way of interacting with git repositories, allows for flexible code organization [15]. Previously, links to repositories were required to be placed in a certain location within the workspace. The updated feature simplifies code management and provides flexibility or code organization.

Solution Alternatives

There are multiple ways to organize software development environments, unity catalog configurations, and group memberships to address the challenges presented by managing sensitive data.

Flow of Actual Data

The fundamental question to answer is: "Should data of any kind flow from an Actual environment to a development environment?" If it is acceptable, then the next step is to select the appropriate sensitive data protection mechanism. If not, then use Synthetic in Development, never flowing data from Actual environments to Synthetic, and vice versa.

Production and Pre-Production Environment

Starting with two workspaces (Production and Pre-Production – sometimes called Stage), we can discuss the elements of the environments. Data processing related workspaces and environments will contain

mechanisms to ingest and refine data using the Medallion architecture [16]. We recommend using a metadata-driven ingestion and refinement asset, similar to Avanade's Metadata Driven Management Framework (MDMF). Such assets store information required to perform an operation in a data store, rather than using hand-coded notebooks. You can see the metastore, metadata-driven notebook, and the CI/CD publish process in Figure 13.

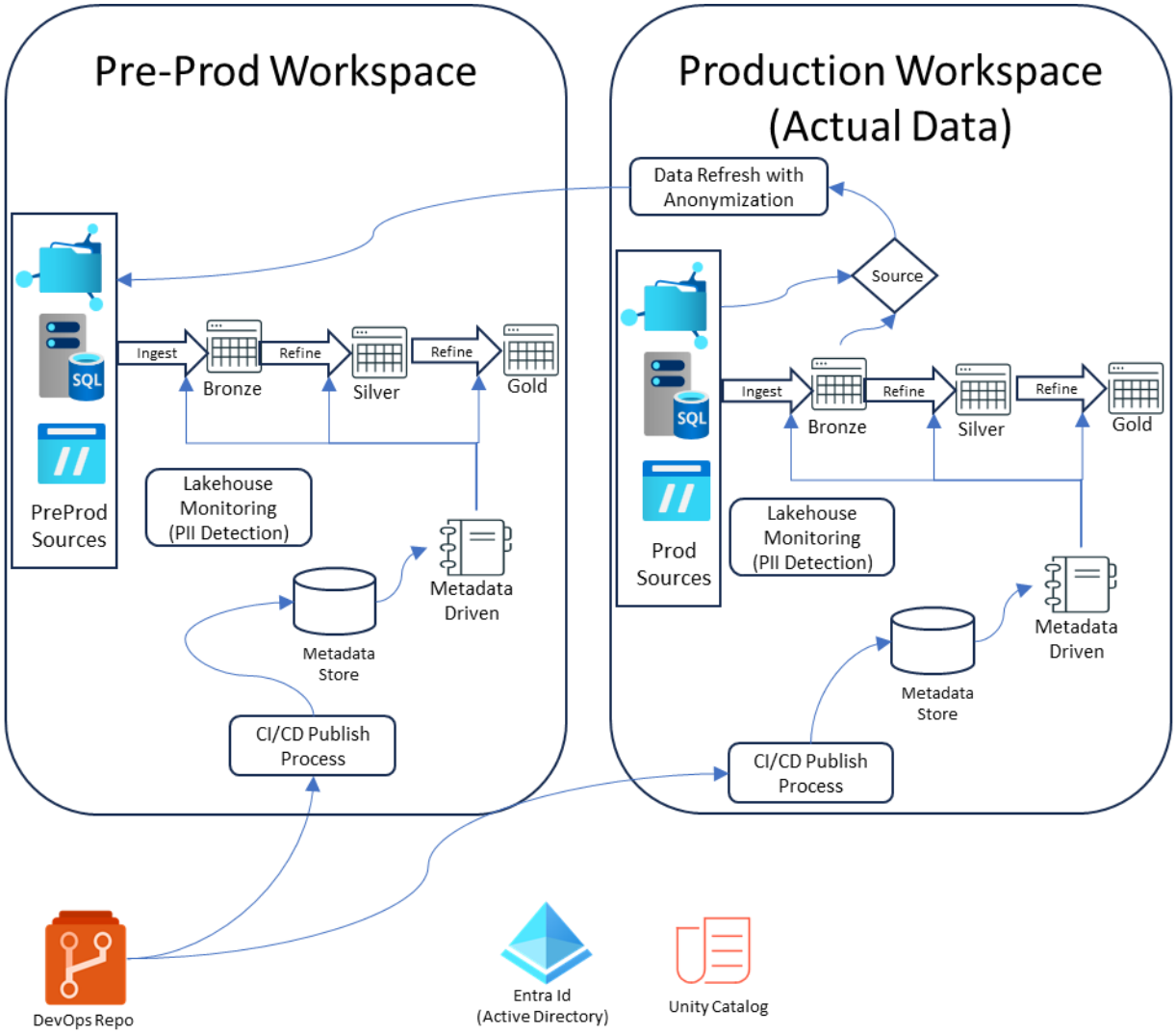


Figure 13: Production and Pre-Production

We also recommend levering Lakehouse Monitoring, in particular the PII Detection capabilities are relevant to this discussion. You may consider using multiple tools, such as Dynamo.AI. This will help ensure that sensitive data did not escape tagging or segregation.

Production environments combine production catalogs with production Databricks workspaces. Production environments utilize production sources. Many organizations lack representative data

sources for all environments, as such, we are certain that production sources will exist while other environments (such as dev) may not have source systems. Instead, data is often copied from a production catalog to a non-production catalog, often applying sampling and sensitive data handling. When making that copy, we need to decide if we will copy from the Bronze tables or go to the source systems directly. Copying from the source systems is likely better, however, it may introduce additional load on those systems. The reason a development environment exists is to provide a location to perform data engineering development (such as populating metadata stores), and then promote that work product to an environment to test it for accuracy and to test the promotion and deployment process.

Pure Synthetic Data in Lower-Level Environments

The proposed solution has multiple environments. We name those environments based on the types of data they typically process. The Synthetic environment contains generated data while the Actual Environment contains Actual data. The Actual data may include sensitive information. All environments are linked via Unity Catalog, sharing the same Unity Catalog metastore.

The data in the Synthetic environment is created based on the table definitions from the Actual environment. Ideally, the synthetic data generation approach takes into account information about the data it is emulating. For example, if the data contains time-series data, the generated Synthetic data should follow similar patterns as the Actual data. Alternatively, systems such as Tonic.AI can be used to create Synthetic data that is similar to Actual data.

A key concept with Synthetic data is that it should not be used in an Actual environment. Ideally, Synthetic and Actual data never leave their respective environments.

Unity Catalog controls access in all instances based on environments, user role, and permissions. Lastly, all environments leverage the same DevOps Repositories, enabling infrastructure as code, access related software development, and software and data engineering code to be promoted through the software development lifecycle. As you can see in Figure 14, the architecture is similar to the Actual environments except for having a Data Generator.

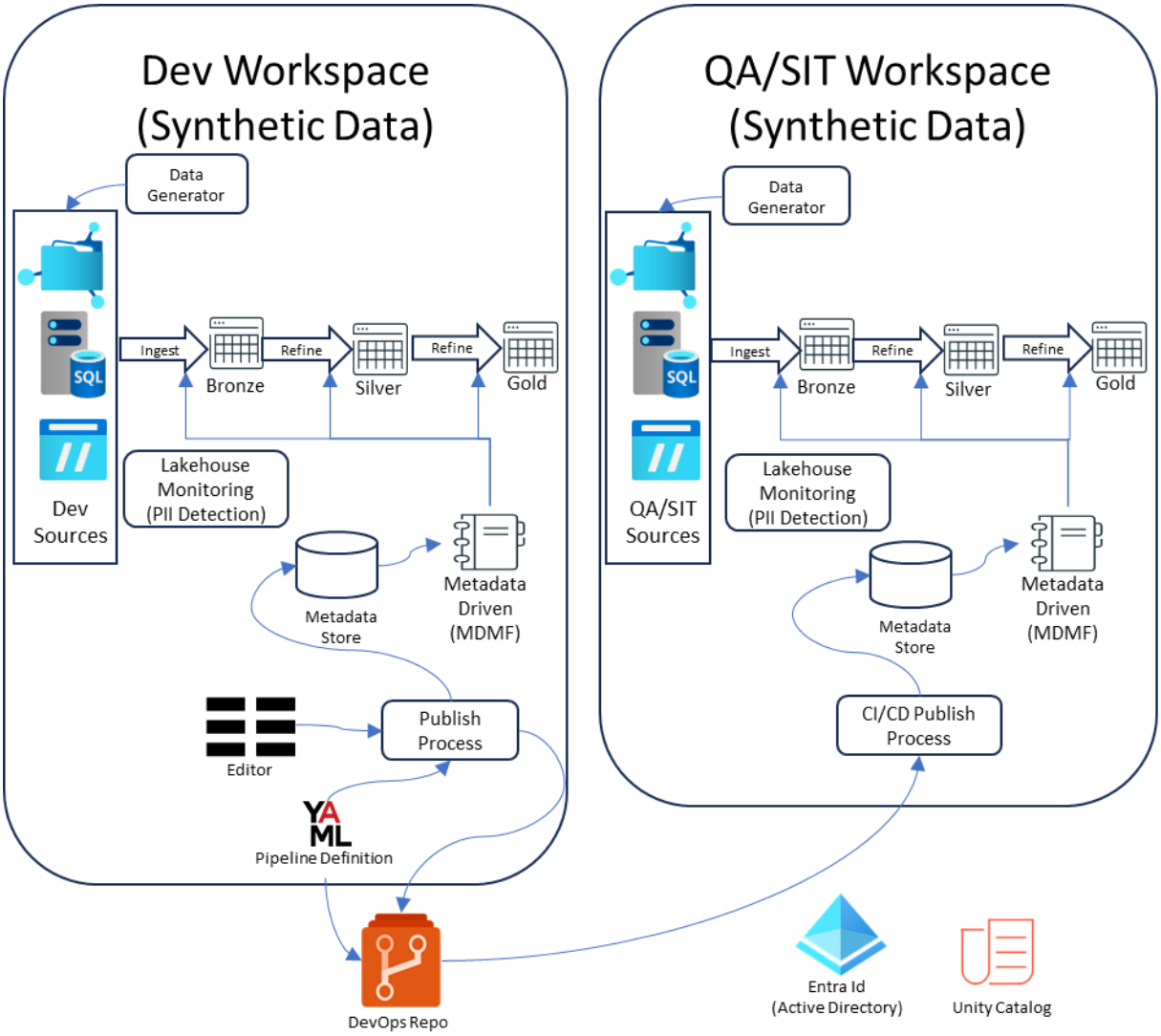


Figure 14: Synthetic Data Environments

We have the building blocks for our proposed solution. Next, we will discuss how they can be combined.

Architectural Variations

There are two variations of this Synthetic Lower Environment and Actual Higher Environment approach. The core issue is that sometimes the Synthetic environment is physically located in a different Azure region than the Actual environment, as shown in Figure 15. For example, it is not uncommon for offshore developers performing data engineering activities to access environments containing Synthetic data. Once those activities have passed testing, they are deployed to the Actual environment. Alternatively, you may have all resources within the same Azure Region.

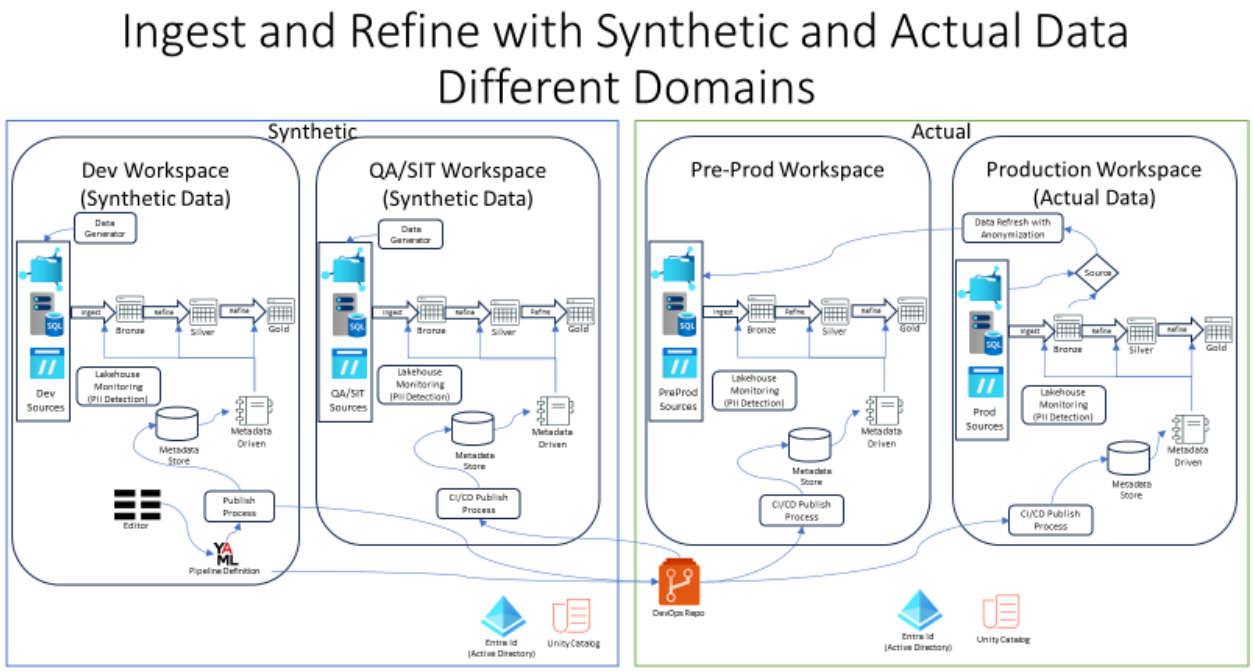


Figure 15: Different Azure Regions

A slightly different approach occurs if we combine the traditional three environment Actual environment with a Synthetic environment with two areas, as Shown in Figure 16. This can occur when adding a Synthetic environment to a legacy three stage environment. Utilizing CI/CD and other forms of automation can reduce the time and effort to promote across the various environments.

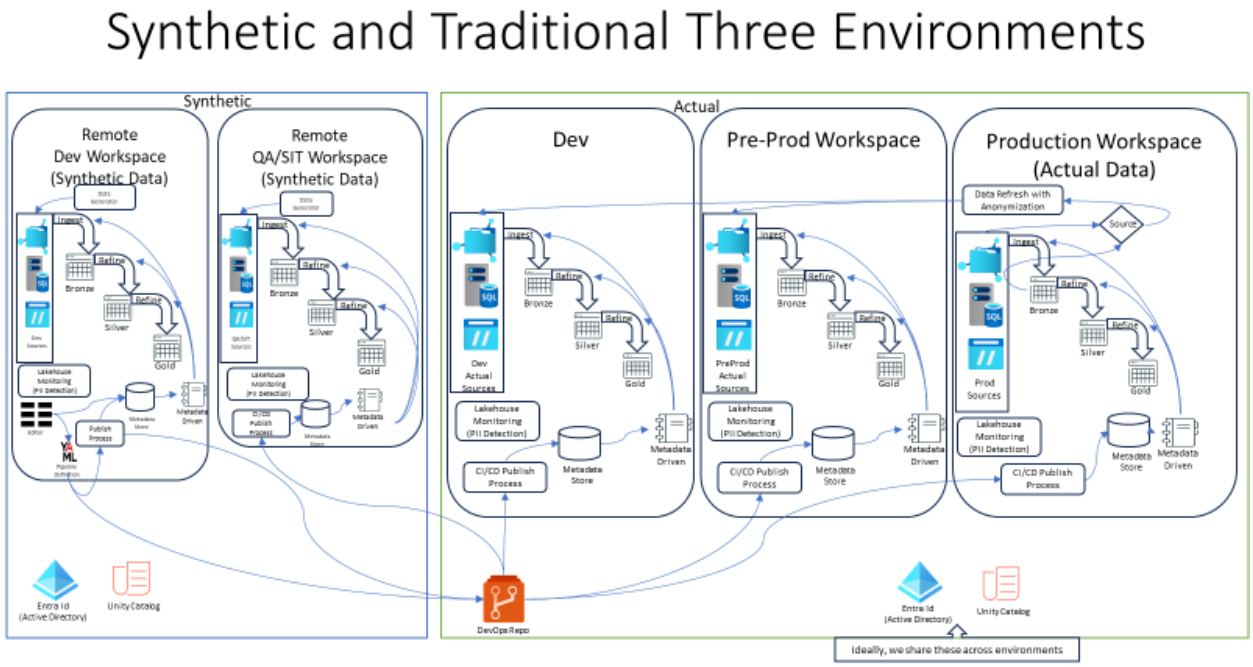


Figure 16: Two Synthetic and Three Actual Environments

In this section we have introduced an approach for combining Synthetic data with Actual in multiple regions. In the next section, we will discuss the approach and evaluate the recommendation in detail.

Recommendation

The recommendation is to use Synthetic data in lower environments. Avoid commingling Actual and Synthetic data unless it is part of a sensitive data protection mechanism or you are creating additional data during AI-related activities.

For data engineering activities, only use Synthetic data in Dev. This will minimize exposure. Additionally, synthetic data created with the intent of validation can simplify unit testing. Focusing on the creation of synthetic data so that it causes the data engineering flows to test the edge cases is an additional potential benefit of synthetic data. Additionally, you should use a separate workspace for development activities. Ideally, that workspace would share the same Entra ID, DevOps Repository, and Unity Catalog as your production workspace. All workspaces should be linked to the same DevOps Repository. This will ensure consistency during development, testing, and deployment to production.

Recommended Solution Benefits

The obvious benefit of this approach is that Actual data will not be used during development. While there may be data-specific issues that only manifest with Actual data, the likelihood of those instances only appearing in production can be minimized by creating Synthetic data that emulates the problematic data. This solution enables onboarding resources in geographic regions that might be problematic otherwise.

Unity Catalog Contributions

Unity Catalog contributes significantly to this solution. In particular, Unity Catalog's course and fine-grained access controls underpin the solution. Unity Catalog' centralized governance model ensures consistent and holistic security.

Another significant contribution of Unity Catalog is the ability to access data from alternative environments, such as a Data Scientist's sandbox. The model enables us to move past how our data is constructed and focus on realizing business value.

In this section we discussed the recommended solution. Next, we revisit synthetic data and discuss the challenges associated with constructing and using it.

Challenges with Synthetic Data

There are several challenges associated with the construction and utilization of synthetic data. When constructing relational data, the synthetic data must adhere to the constraints associated with the data model. For example, if a column has a unique constraint, we must ensure that duplicate data is not created. Often, tables have a foreign key relationship. In this case, we must populate the tables in the appropriate order and ensure that the keys match. These challenges are easily addressed but should be considered when constructing or selecting a synthetic data generation system.

Workloads that attempt to detect patterns in data are challenging with synthetic data. For example, if we attempt to predict a value based on several features, pseudo-randomly generated data will likely result in no convergence or unreliable outcomes. Likewise, generalization is not very likely if we train a model using synthetic data.

However, if we use an approach where training occurs in each environment after promotion, synthetic data is an appropriate choice. In that case, we perform software engineering activities and then perform the training using Actual data in a higher environment.

As previously mentioned, creating synthetic data can be challenging. In the next section, we discuss a framework for creating synthetic data.

Synthetic Data Generation Libraries

The generation of synthetic data can be a challenging endeavor. This section discusses the tools available to create synthetic data, both open-source and paid. We provide a brief discussion of each.

There are multiple open-source projects associated with generating synthetic data. In this section, we discuss a few. We start with a commonly used library, Faker.

Faker

The Python Faker library was first released on December 23, 2010. It has become the go-to for creating synthetic or fake data; you can find the documentation at https://faker.readthedocs.io/en/master/. Faker uses a provider architecture, enabling robust extensibility. It integrates with Factory Boy (https://factoryboy.readthedocs.io/), rapidly creating synthetic data within a domain. For example, if a column that should contain email addresses is identified, there is an out-of-the-box capability to generate them.

Databricks Labs Data Generator

Given the topic and audience of this paper, it would be remiss not to discuss the Databricks Labs Data Generator (DBLDataGen), found at https://github.com/databrickslabs/dbldatagen. DBLDataGen is a Python library that enables synthetic data generation at scale using Spark. One experimental capability of DBLDataGen is that it can generate data based on a schema or data. Additionally, there is a documented process for populating multiple tables with relationships.

YData Synthetic

Pandas Profiling is a well-known way of evaluating the nature of datasets. It has recently been rebranded YData-profiling. YData also has a synthetic data offering, available from https://github.com/ydataai/ydata-synthetic. YData-synthetic supports various generative models, including Generative Adversarial Network (GAN) approaches.

This section discussed some of the available open-source synthetic data generators. In the next section, we discuss using one of these alternatives as part of a holistic approach to generate data.

Approach to Generation of Synthetic Data

We now have the building blocks for a synthetic data solution. This section discusses the high-level approach to synthetic data generation. Then, we share a detailed pseudocode to determine the order of processing.

High-Level Approach

Documenting simple steps is important to ensure they are not overlooked. In this section, we outline the high-level steps needed to generate synthetic data.

Based upon a supplied source system

1. Extract the schema for source system (SQL DDL) and relationships (INFORMATION_SCHEMA) and determine the order of processing based on relationships.
2. Profile source data system against production data, save profile data for generation.
3. Manually specify (if not able to determine programmatically)
    - Truncate and regenerate or Incremental Load
    - Ongoing arrival data rate or the truncate and reload frequency
    - Data quality/validation rules for each table
4. Ensure the schema exists in the synthetic development source system
5. Generate data into Dev source systems using profile data.
    a. For foreign key relationships, for each record, randomly assign a related record of leverage that capability in your selected data generation tool or library
6. Validate generation

On a schedule, for specified tables

1. If set to truncate, delete all records
2. Create data for specified tables (incremental if not truncating, full load otherwise)
3. Validate generation, including primary and foreign key relationships, if present in the metadata.

Figure 17 contains a visual representation of this approach.

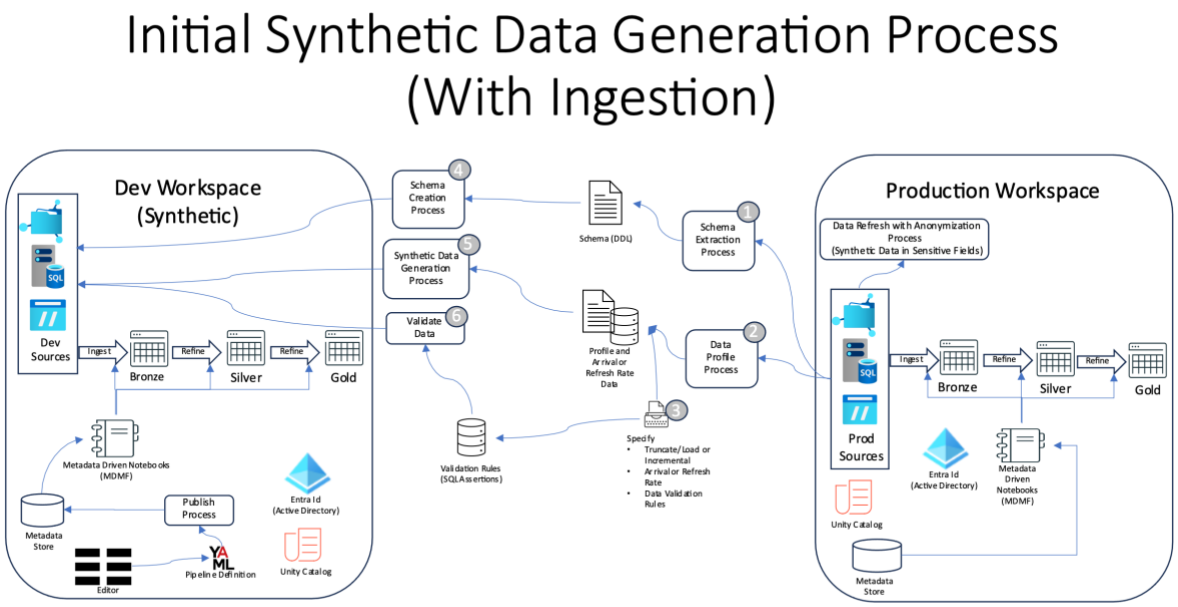
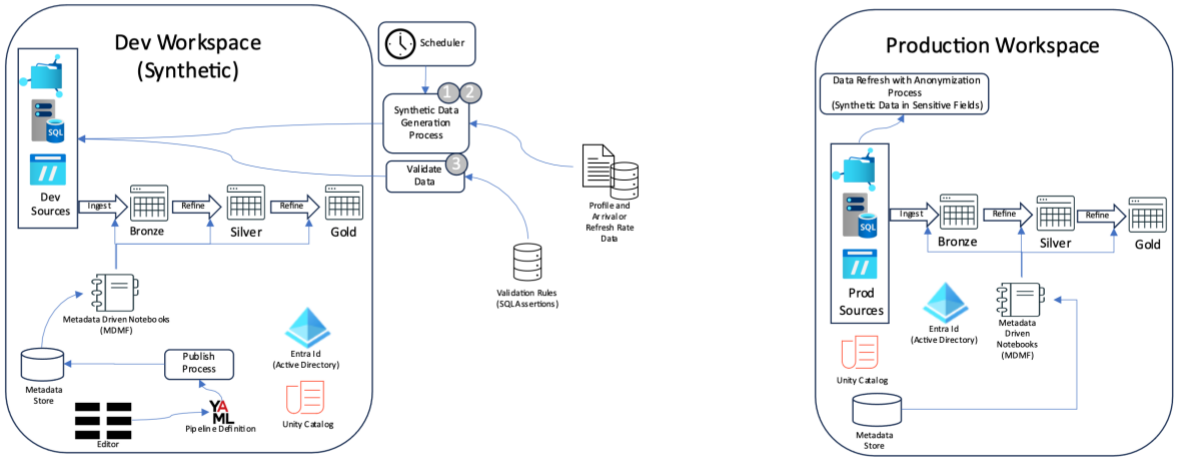


Figure 17: Data Generation Process

Often, we want to generate data in an ongoing fashion. This allows us to simulate patterns that occur in the Actual data. This might be as advanced as time series data or as simple as daily refreshes. Figure 18 holds a visual representation of the ongoing generation process.

Ongoing Synthetic Data Generation Process



Detailed Pseudocode for Order of Processing

Provided here is a high-level pseudocode to determine the processing order for data generation. If you are a non-technical person you can safely skip this section.

It considers foreign key relationships, traversing the graph dependency graph to ensure referenced tables are populated before referring tables. Note that this approach does not handle self-referencing relationships or longer cycles within the dependency graph. It is a depth-first search with minor tweaks.

- For each database
    - Use the INFORMATION_SCHEMA metadata to construct a graph containing the database's tables (nodes) and relationships (edges).
    - Add a property to all nodes to indicate the processing order. Initialize it to a negative one.
    - Process the graph, starting with nodes that are referenced by other nodes. Use the vertices outDegree to determine the processing order, starting with the highest degree. Start with a processing order of one. For each node:
        - If the node has been assigned a processing order, return that value
        - For each node that the node references
            - Recurse, supply the node and increase the processing order

        - Using the max returned processing order, set the processing order for the current node
    - For each table that has not been processed (no references)
        - Set the processing order to zero

After having assigned all tables a processing order, use that order when generating data.

Case Study

This case study is forward-looking and references a fictitious organization. It discusses a planned use of the approach presented here.

A large not-for-profit medical organization is creating an intelligent data platform using Databricks. The organization uses Epic to track medical information, Workday for human resource data, and various other systems. The organization wants to leverage offshore resources but is unwilling to onboard them into its systems.

Instead, Synthetic data is used in a separate Databricks workspace. Because of networking and other reasons, the organization's Unity Catalog metastore is not accessible. A new Azure subscription, including Azure Data Lake Gen2, Databricks, and required supporting technologies, has been provisioned.

Most data ingested by the organization is processed through Azure SQL Database tables, as this standardizes ingestion. The Actual schema was applied to Azure SQL Database servers and tables to emulate the Actual data source systems. Those tables were populated using an approach similar to that outlined in the previous section.

Avanade provided MetaData Management Framework (MDMF) as an accelerator to reduce the time to define ingestion. MDMF leverages Azure Data Factory and Azure Databricks to reduce the time to value for data. MDMF leverages YAML Ain't Markup Language™ (YAML) files to encode the desired module and operations desired. For example, a YAML file may contain a list of files to ingest and, in turn, standardize to the Bronze zone. The YAML files are versioned using Azure DevOps. When a pull request is approved, the YAML files are used to update the control metastore for MDMF, enabling the operations to be available.

The offshore resources could create and test YAML files for each table for a given data source, resulting in at least 50% improvement over hand coding. Programmatic generation of the YAML files from source metadata results in an even greater velocity of ingestion and refinement. Additionally, the promotion of metadata is simplified. Once the offshore test team approves a set of YAML files, they are promoted to the Actual environment, where they are deployed in the development environment. From that point, traditional testing and acceptance processes are followed.

Conclusion

Sensitive data is inevitable in any field that interacts with people. Organizations must have a documented plan for addressing sensitive data. In this paper, we have discussed the challenges sensitive data brings. We presented definitions with the hope of removing ambiguity while discussing this

challenging topic. We also presented a solution for dealing with sensitive data that relies on creating synthetic data. We discussed the benefits of the proposed solution. Then, we discussed several open-source libraries that generate synthetic data. A high-level approach was presented along with a detailed pseudocode to determine the order to process tables. Lastly, we presented a case study related to synthetic data.

This paper only scratches the surface of how Databricks, and Unity Catalog in particular, can address sensitive data challenges. When addressing a challenge, engaging with a firm with expertise in that domain is often necessary. Sensitive data is no different. You should partner with a firm with extensive experience and resources trained in Unity Catalog and other topics presented here. Given the high cost of failure, it is essential that sensitive data processing is a priority in your organization.

References

1. Bauer, A., et al., *Comprehensive exploration of synthetic data generation: A survey.* arXiv preprint arXiv:2401.02524, 2024.
2. Databricks. *Databricks Clean Rooms: Privacy-safe collaboration for data, analytics and AI*. 2024; Available from: https://www.databricks.com/product/clean-room.
3. U.S. Department of Labor. *Guidance on the Protection of Personal Identifiable Information*. n.d.; Available from: https://www.dol.gov/general/ppii.
4. Moore, W. and S. Frye, *Review of HIPAA, part 1: history, protected health information, and privacy and security rules.* Journal of nuclear medicine technology, 2019. **47**(4): p. 269-272.
5. Majeed, A. and S. Lee, *Anonymization techniques for privacy preserving data publishing: A comprehensive survey.* IEEE access, 2020. **9**: p. 8512-8545.
6. Databricks. *Databricks Labs Data Generator (dbldatagen)*. 2024; Available from: https://github.com/databrickslabs/dbldatagen.
7. Information Commisioners Office. *What is personal data?* n.d.; Available from: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/personal-information-what-is-it/what-is-personal-data/what-is-personal-data/.
8. Vazão, A.P., et al., *Implementing and evaluating a GDPR-compliant open-source SIEM solution.* Journal of Information Security and Applications, 2023. **75**: p. 103509.
9. Databricks. *Introducing the Databricks AI Security Framework (DASF): An actionable framework to manage AI security*. 2024; Available from: https://www.databricks.com/blog/introducing-databricks-ai-security-framework-dasf.
10. Databricks. *Identity best practices*. 2024; Available from: https://docs.databricks.com/en/admin/users-groups/best-practices.html.
11. Francis, D., *Mastering Active Directory: Design, Deploy, and Protect Active Directory Domain Services for Windows Server 2022*. 2021: Packt Publishing Ltd.
12. Microsoft. *Apply tags*. 2024; Available from: https://learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/tags.
13. Databricks. *Unity Catalog best practices*. 2024; Available from: https://docs.databricks.com/en/data-governance/unity-catalog/best-practices.html.
14. Microsoft. *Best practices for operational excellence*. 2024; Available from: https://learn.microsoft.com/en-us/azure/databricks/lakehouse-architecture/operational-excellence/best-practices.
15. Microsoft. *Git Integration with Databricks Git Folders*. 2024; Available from: https://learn.microsoft.com/en-us/azure/databricks/repos/.
16. Databricks. *Medallion Architecture*. n.d.; Available from: https://www.databricks.com/glossary/medallion-architecture.