Data Taxonomy

Taxonomy is the scientific name used predominantly to describe and classify organisms.

Traditionally taxonomy uses hierarchical classifications to help scientists understand and organize the diversity of life on our planet.

A **data taxonomy** typically includes a hierarchical structure. In this case it is of categories and subcategories that describe different types of data. These categories may include data types, data sources, data sensitivity and data ownership.

By creating a clear and consistent taxonomy, organizations can ensure that data is properly classified, and that access to it is granted based on the appropriate level of security and permissions.

Data taxonomy can also help to identify data that is redundant or obsolete and help organisations prioritise data for backup and disaster recovery purposes.

There are **various types of data taxonomies** that can be used to categorise and organise data in an organisation.

Here are some common types of data taxonomies:

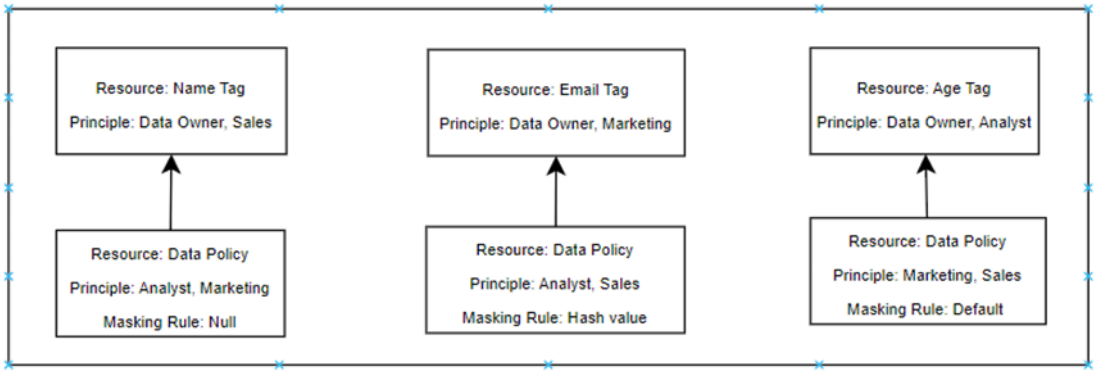
- **Functional Taxonomy:** Organises data based on its functional use, such as sales data, marketing data, or finance data.
- **Information Taxonomy:** Classifies data based on the type of information it contains, such as customer data, product data, or financial data.
- **Technical Taxonomy:** Technical characteristics of data, such as the format, storage location, or data source.
- **Organisational Taxonomy:** Categorises data based on the business unit or department that owns or uses the data, such as HR data, IT data, or sales data.
- **Regulatory Taxonomy:** Classifies data based on regulatory requirements, such as data subject to HIPAA or GDPR compliance.
- **Time-Based Taxonomy:** Organizes data based on the time in which it was collected, such as historical data or real-time data.

The type of data taxonomy that an organisation uses will depend on the specific needs and goals of the organisation, as well as the types of data being managed.

Often, multiple taxonomies will be used in combination to create a comprehensive framework for managing and organizing data.

1. Data owners should have access for all data in the raw format.
2. Marketing teams can access only raw data from email column
3. Analysts team access only raw data from age column
4. Sales team should see only raw data from name column
5. Merchandising team should not see any data

This is summarised in the table below:



General Data Protection Regulation (GDPR)

The General Data Protection Regulation (**GDPR**), introduced by the European Union, took effect on May 25, 2018. With the introduction of GDPR, organisations became compelled to understand data and take steps to protect data. These drives organisational needs to collaborate with enterprise data architecture to classify data. One of the ways they have partially achieved the requirements includes Data Masking.

Data Masking is technique for protecting sensitive data and ensuring compliance with data privacy regulations. This is typically done by replacing the original data with a modified version that appears to be the same but contains no actual sensitive information, while still allowing authorised users to access and use the data.

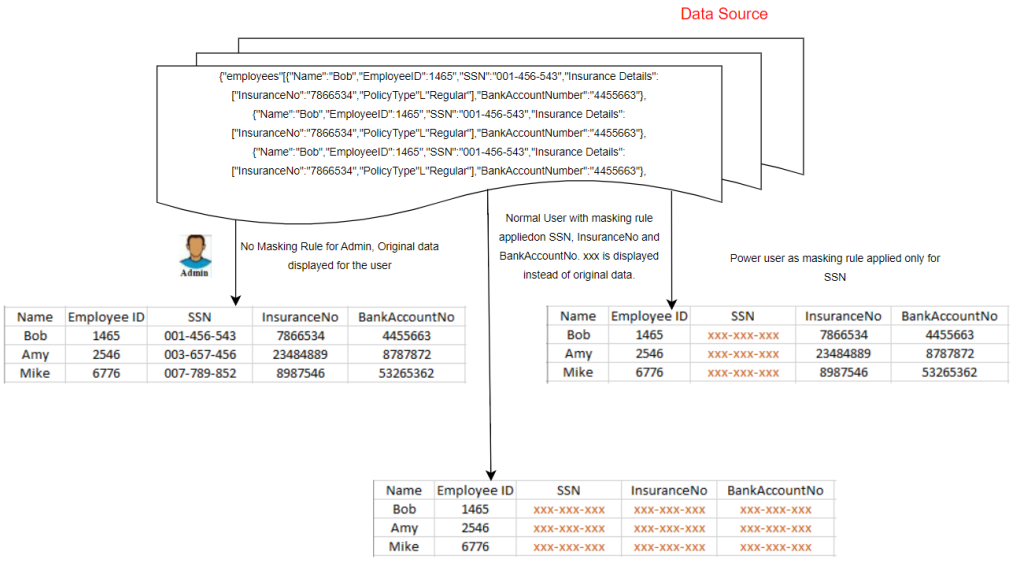
How does data masking work?

The data masking process is simple; however, it has different techniques and types.

In general, organisations start with identifying all sensitive data then use algorithms to mask the data and or replace it with structurally identical but numerically different data.

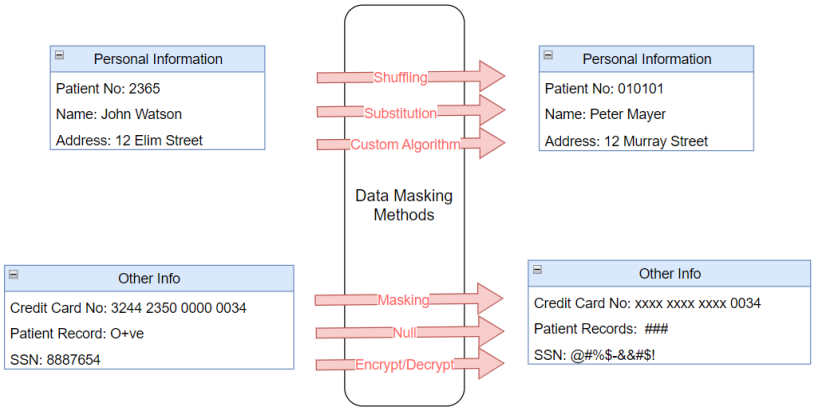
What do we mean by structurally identical? For instance, passport numbers are 9 digits in the US and individuals usually must share their passport information with airline companies. When an airline company builds a model to analyse and test the business environment, they create a different 9-digit long passport ID or replace some digits with characters.

Below is an example of how data masking works:



Techniques of data masking:

- Substitution: Replacing sensitive data with a different value that has no meaning, such as replacing a credit card number with a random set of digits.
- Shuffling: Rearranging the order of data elements so that they are still recognisable, but the actual values are hidden.
    - For example, shuffling the letters in a person's name.
- Encryption: Converting sensitive data into an unreadable form using an encryption algorithm. The data can only be decrypted using a specific key or password.
- Number and Date Variance: Applying the same variance to create a new dataset which will not change the accuracy of the dataset.
- Character Scrambling: Randomly rearranging the character orders.
- Nulling out or deletion: Replace sensitive data to null value
- Masking Out: Some part of original data to be masked.



Types of Data Masking:

- Static data masking (SDM): In SDM, data is masked in the original data then duplicated to a test data so that businesses can share the test data to the third-party vendors.
- Dynamic data masking (DDM): In DDM the original sensitive data remains in the repository and accessible to an application when authorized by the system.

- Data is never exposed to unauthorized users; contents are shuffled in real-time on-demand to make the contents masked. Only authorised users can see authentic data.

How can data masking be implemented in Databricks using Python?

In this example, the mask_credit_card function is defined to replace all but the last 6 digits of a credit card number with X's.

A user-defined function (UDF) is then created for this function. The data to be masked is loaded from a CSV file using Spark, and the masking function is applied to the credit_card_number column using the withColumn method.

Finally, the masked data is written to a new CSV file.

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

# Define masking function
def mask_credit_card(card_number):
    # Replace all but the last 6 digits with X's
    return 'X'*(len(card_number)-6) + card_number[-6:]

# Create a user-defined function (UDF) for the masking function
mask_credit_card_udf = udf(mask_credit_card, StringType())

# Load the data to be masked
df = spark.read.csv('/path/creditdata.csv', header=True)

# Apply the masking function to the credit card number column
df = df.withColumn('mask_credit_card',
mask_credit_card_udf(df['credit_card_number']))

# Write the masked data to a new file
df.write.csv('/path/mask_data.csv', header=True)
```

How can data masking be implemented in Azure using SQL Server?

In this example, a table called Customers is created to hold the sensitive data, including a credit_card_number column.

A sample of data is inserted into the table. A data masking function called mask_credit_card is created to mask the credit card number by replacing all the last 6 digits with X's.

A data masking policy is then applied to the credit_card_number column using the ALTER TABLE statement, with the FUNCTION parameter set to the mask_credit_card function.

Finally, the data is queried using a simple SELECT statement to verify that the credit card numbers have been masked.


```
-- Create a table to hold the sensitive data
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(50),
    credit_card_number VARCHAR(16)
);

-- Insert some sample data
INSERT INTO Customers (customer_id, first_name, last_name, email, credit_card_number)
VALUES (1, ‘Amy’, 'Doe', 'amy.doe@example.com', ‘12398755989’),
       (2, ‘Bob’, 'Doe', 'bob.doe@example.com', ‘98938498932’);

-- Create a data masking function to mask the credit card number
CREATE FUNCTION mask_credit_card(@credit_card_number VARCHAR(16))
RETURNS VARCHAR (16)
WITH SCHEMABINDING
AS
BEGIN
    -- Replace all the last 6 digits with X's
    RETURN 'XXXXXXXXXXX' + RIGHT(@credit_card_number, 6);
END;

-- Create a data masking policy to apply the masking function
ALTER TABLE Customers
ALTER COLUMN credit_card_number ADD MASKED WITH (FUNCTION = dbo.mask_credit_card);

-- Query the data with the masked credit card numbers
SELECT * FROM Customers;
```

How can data masking be implemented in AWS using Amazon RDS with PostgreSQL?

In this example, a table called Customers is created to hold the sensitive data, including a credit_card_number column.

A sample of data is inserted into the table. A data masking function called mask_credit_card is created to mask the credit card number by replacing all but the last 6 digits with X's.

A data masking policy is then applied to the credit_card_number column using the ALTER TABLE statement, with the FUNCTION parameter set to the mask_credit_card function.

Finally, the data is queried with SELECT statement to verify that the credit card numbers have been masked.


```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(50),
    credit_card_number VARCHAR(16)
);

-- Insert some sample data
INSERT INTO Customers (customer_id, first_name, last_name, email, credit_card_number)
VALUES (1, ‘Amy’, 'Doe', 'amy.doe@example.com', ‘12398755989’),
       (2, ‘Bob’, 'Doe', 'bob.doe@example.com', ‘98938498932’);

-- Create a data masking function to mask the credit card number
CREATE OR REPLACE FUNCTION mask_credit_card(card_number VARCHAR)
RETURNS VARCHAR
AS $$
BEGIN
    -- Replace all the last 6 digits with X's
    RETURN 'XXXXXXXXXXXX' || RIGHT(card_number, 6);
END;
$$ LANGUAGE plpgsql;

-- Create a data masking policy to apply the masking function
ALTER TABLE Customers
ALTER COLUMN credit_card_number SET MASKED WITH FUNCTION mask_credit_card();

-- Query the data with the masked credit card numbers
SELECT * FROM Customers;
```

Types of data that require data masking

- Personally identifiable information (PII): Any data that could potentially be used to identify a particular person.
    - For example, full name, social security number, driver’s license number, and passport number.
- Protected health information (PHI): PHI includes demographic information, medical histories, test and laboratory results, mental health conditions, insurance information, and other data that a healthcare professional collects to identify appropriate care.
- Payment card information (PCI): There is an information security standard for organizations to follow while handling branded credit cards from the major card schemes.
- Intellectual property (IP): IP refers to creations, such as inventions; literary and artistic works; designs; and symbols, names and images used in commerce.

In conclusion Data Taxonomy is critical for compliance.

The creation of a taxonomy is just the beginning.

After this broad classification, it is important to assign data owners and map these high-level categories down to actual data points and ensures the critical data is under control. This drives alignment between legal, compliance, privacy, and enterprise data management. So, the data can be processed and stored in a way that maintains confidentiality.

If your business needs help with their Data Taxonomy, please contact a Fujitsu Data & AI specialist now.

Contact
Fujitsu Data & AI
+61 3 9924 3000