October 2023

# METHOD AND SYSTEM FOR METADATA MAINTENANCE IN A DATA LAKE

Harleen Kaur
*Visa*

Durga Kala
*Visa*

# METHOD AND SYSTEM FOR METADATA MAINTENANCE IN A DATA LAKE

**VISA**

**INVENTORS:**

**Harleen Kaur**

**Durga Kala**

## TECHNICAL FIELD

**[0001]** The present subject matter in general relates to a method and system for metadata optimization/maintenance in a data lake and particularly, a method and system for identifying stale data sets in the data lake for deprecation.

## BACKGROUND

**[0002]** Modern data architectures support an increasing number and variety of business use cases. Examples include product development, tailored customer experiences, fraud detection, regulatory compliance, and data monetization. To enable such use cases, a data-driven organisation needs modern solutions for accessing, managing, and processing data. One of them is a data lake, a central repository for storing, processing, and securing large amounts of structured, semi-structured and unstructured data.

**[0003]** As data is growing in terms of volume, variety and velocity organisations are moving towards different data technologies, investing heavily in tooling around managing and mining the data. Often these tools work very well in the contained setup but in bigger schema of things, like data lake these have resulted in silos. In many organizations, teams keep data in separate buckets, and those buckets are partitioned according to the path to the data. The same applies to metadata. Teams often lack file system capabilities or hierarchy that enable efficient usage of metadata. Keeping data in separate silos that do not align has serious implications for the organization’s ability to leverage metadata to support its most important use cases. Thus, these silos lead to siloed metadata, making it hard for an organization to truly derive the competitive and comprehensive intelligence. It is not only important to connect the dots with complete metadata but with complete alive metadata.

**[0004]** Most of the organizations rely on metadata management tools that focus on crawling or pushing the metadata to a central HUB with no ability to really track how alive is the metadata. The presence of a physical dataset reflected by its logical metadata is not enough in today’s world to qualify the metadata to be “alive”.

**[0005]** Therefore, there is a need for a method and a system for identifying whether the metadata is alive in the data lake and effectively deprecate stale metadata from the data lake.

## SUMMARY

**[0006]** In an aspect, the embodiment of the present disclosure relates to a method and a system for metadata maintenance in a data lake by identifying stale data sets in the data lake for deprecation.

**[0007]** The present disclosure comprises a metadata Hub, having a processor and a memory, and may acts as a central repository. The metadata hub may also include a deprecator system or a deprecator, that uses an AI (artificial intelligence) model to deprecate unused datasets. The metadata hub may be communicatively connected to a large-scale data repository, also known as a data lake. The metadata hub may be configured to control the flow of ingest data from any relational database into the data lake. The data lake may include any data storage technology for offering storage for all data types. The metadata hub may be configured to send meta pulse queries to the data lake, where the meta pulse queries are used to pull usage stats of datasets, which includes identifying one or more actions performed on the datasets stored in the data lake. In response to the meta pulse queries the data lake is configured to provide a response pulse or metadata pulses indicating, the one or more actions performed on the datasets stored in the data lake. Upon receiving the response pulses, the metadata hub may be configured to perform one or more response actions on the datasets stored in the data lake, based on the response pulses. The one or more response actions may include deprecation of unused datasets by the deprecator system in the metadata hub, using a decision tree model algorithm.

**[0008]** The present disclosure provides an advantage of automatic deprecation of inactive datasets. Using meta pulse queries organisations may derive insights on true and active datasets. Further, automatic deprecation of inactive datasets may reduce storage and may help organisations to manage and mine only alive datasets.

**[0009]** These and other features and characteristics of the present invention, as well as the methods of operation and functions of the related elements of structures and the combination of parts and economies of manufacture, will become more apparent upon consideration of the following description with reference to the accompanying drawings, all of which form a part of this specification, wherein like reference numerals designate corresponding parts in the various figures. It is to be expressly understood, however, that the drawings are for the purpose of illustration and description only and are not intended as a definition of the limits of the invention. As used in the specification, the singular form of “a,” “an,” and “the” include plural referents unless the context clearly dictates otherwise.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0010]** The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, explain the disclosed principles. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same numbers are used throughout the figures to reference features and components. Some embodiments of device or system and/or methods in accordance with embodiments of the present subject matter are now described, by way of example only, and with reference to the accompanying figures, in which:

**[0011]** **FIG. 1** shows an exemplary architecture of a data lake, in accordance with some embodiments of the present disclosure;

**[0012]** **FIG. 2** shows an exemplary architecture of a system for identifying stale data sets in the data lake for deprecation, in accordance with some embodiments of the present disclosure;

**[0013]** **FIG.3** illustrates an exemplary database table, in accordance with some embodiments of the present disclosure;

**[0014]** **FIG. 4** shows an exemplary API of a system for identifying stale data sets in the data lake for deprecation;

**[0015]** **FIG. 5** shows an exemplary flow diagram of execution of deprecation of a table, in accordance with some embodiments of the present disclosure; and

**[0016]** **FIG. 6,** illustrates an exemplary flow diagram of a method for identifying stale data sets in the data lake for deprecation, in accordance with some embodiments of the present disclosure.

**[0017]** The figures depict embodiments of the disclosure for purposes of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the disclosure described herein.

## DESCRIPTION OF THE DISCLOSURE

**[0018]** In the present document, the word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any embodiment or implementation of the present subject matter described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments.

**[0019]** While the disclosure is susceptible to various modifications and alternative forms, specific embodiment thereof has been shown by way of example in the drawings and will be described in detail below. It should be understood, however that it is not intended to limit the disclosure to the particular forms disclosed, but on the contrary, the disclosure is to cover all modifications, equivalents, and alternative falling within the spirit and the scope of the disclosure.

**[0020]** The terms “comprises”, “comprising”, or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, device, or method that comprises a list of components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or device or method. In other words, one or more elements in a device or system or apparatus proceeded by “comprises… a” does not, without more constraints, preclude the existence of other elements or additional elements in the device or system or apparatus.

**[0021]** The terms "an embodiment", "embodiment", "embodiments", "the embodiment", "the embodiments", "one or more embodiments", "some embodiments", and "one embodiment" mean "one or more (but not all) embodiments of the invention(s)" unless expressly specified otherwise.

**[0022]** The terms "including", "comprising", “having” and variations thereof mean "including but not limited to", unless expressly specified otherwise. The terms “user” and “customer” have been used interchangeably throughout the disclosure. In the present disclosure, the term “offers” “promotions” and “promotional offers” have been used interchangeably throughout the disclosure. The offers/promotions may include discounts, incentives, rewards, rebates, gifts, cashbacks, coupons, reward points, or any such benefit which can be availed/redeemed upon satisfaction of certain conditions.

**[0023]** The term organization, business, company, or enterprise as used herein may be related to any private, public, government or private-public partnership (PPP) enterprise.

**[0024]** The term Application programming interface (API) may indicate a computing interface that defines interactions between multiple software intermediaries. An API can define the types of calls and/or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. An API can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees

**[0025]** It will be apparent that systems and/or methods, described herein, can be implemented in different forms of hardware, software, or a combination of hardware and software. The actual specialized control hardware or software code used to implement these systems and/or methods is not limiting of the implementations. Thus, the operation and behaviour of the systems and/or methods are described herein without reference to specific software code, it being understood that software and hardware can be designed to implement the systems and/or methods based on the description herein.

**[0026]** Some non-limiting embodiments or aspects are described herein in connection with thresholds. As used herein, satisfying a threshold may refer to a value being greater than the threshold, more than the threshold, higher than the threshold, greater than or equal to the threshold, less than the threshold, fewer than the threshold, lower than the threshold, less than or equal to the threshold, equal to the threshold, etc.

**[0027]** **FIG. 1** shows an exemplary architecture **100** of a data lake, in accordance with some embodiments of the present disclosure. In an embodiment, the architecture may include a metadata hub (**FIG.2**), comprising a processor and a memory, and may act as a central repository. The metadata hub may be communicatively connected to a large-scale data repository, also known as a data lake **102**. In some embodiments, the metadata hub may be an authoritative source of metadata across an organisation. Metadata across the organisation may describe the data stored in the data lake **102**, providing details such as source of the data lake **102**, structure of the data lake **102**, meaning of the data lake **102**, relationships of the data lake **102** with other data, or usage of the data. Metadata facilitates organisations to discover relevant data in the vast amounts of data stored in the data lake **102**. The metadata hub may be configured to control, ingest data **104**, the process of collecting or absorbing data, from any database into the data lake **102**. The data lake **102** may absorb data from one or more data sources. The one or more data sources may be external or internal data sources of an organization. The one or more data sources may include a streaming source **106**, a batched ingestion **108**, or the like. In an embodiment, for the streaming source **106**, ingestion may be continuous, from different sources to the data lake **102**. In an embodiment, the batched ingestion **108** may occur when data collection and transfer occur at regular intervals in batches on a schedule or a trigger, by a user. The data lake **102** may include any data storage technology for offering storage for all data types. As an example, datasets may have different file types or data formats and may be structured (e.g., according to a relational table type structure with records made up of values for one or more columns), semi-structured (e.g., key value-item pairs), or unstructured (e.g., a collection of different files, freeform text, etc.). Because the types of data stored in a data lake **102** may be varied, structural data describing the data set may be determined and maintained for the data set (e.g., within a data catalogue) in order to access the data set. In some embodiments, the data lake **102** may be cloud-based and may provide rapid data access and analytics by having all necessary compute resources and storage objects internal to the data lake **102**. The data lake **102** may also isolate workloads and allocate resources to a prioritized jobs to avoid user concurrency issues from slowing down analyses.

**[0028]** In some embodiments, data stored in a data lake **102** may be available for data science and machine learning **110** and for analytics and business intelligence **112** tools. In some embodiments, data lake **102** may act as a single source for different use cases. In some embodiments, the meta hub may also include a deprecator system or a deprecator, that uses an AI (artificial intelligence) Model for deprecation of unused datasets. The metadata hub may be configured to send meta pulse queries to the data lake **102**, to pull usage metrics or statistics of datasets, which includes identifying one or more actions performed on the datasets stored in the data lake **102**. In response to the meta pulse queries the data lake **102** is configured to provide response pulses metadata pulses indicating the one or more actions performed on the datasets stored in the data lake **102**. Upon receiving the response pulses, the metadata hub may be configured to perform one or more response actions on the datasets stored in the data lake **102**, based on the response pulses. The one or more response actions may include deprecation of unused datasets by the deprecator system in the metadata hub, using a decision tree model algorithm.

**[0029]** **FIG. 2** shows an exemplary architecture **200** of a system for identifying stale data sets in the data lake **102** for deprecation. In an embodiment, the architecture **200** may include a metadata hub **202**, comprising a processor and a memory (not shown in figure), and may acts as a central repository. The metadata hub **202** may be communicatively connected to a data lake **102**. In some embodiments, without limitation, the metadata hub **202** may be within the data lake **102**. The metadata hub **202** may be configured to manage data by providing metadata search and discovery tools and may act as a centralized hub which presents a holistic view of the data ecosystem. The metadata hub **202** may include a data catalogue to organize data assets in the data lake **102**. The metadata hub **202** may also include a data quality system to define rules that assess and evaluate the accuracy of data. In some embodiment, the metadata hub **202** may be configured to trace and track how data flows around in the data lake **102**. The metadata hub **202** may be configured to ingest data from any relational database into the data lake **102**.

**[0030]** In some embodiments, as shown in **FIG.2,** the metadata hub **202** may be configured to send meta pulse queries or heart beats to the data lake **102**. The meta pulse queries may be sent at regular intervals or may be user triggered. The meta pulse queries may be inquiries that can run according to a specific schedule. The meta pulse queries may be messages sent to data lake **102** with a customized frequency. The meta pulse queries may be used to check data frequently, in order to see whether one or more actions performed on the datasets stored in the data lake **102**. The meta pulse queries may be used to pull usage statistics of datasets in the data lake **102**. In some embodiments, the data lake **102** may include a pulse service. The pulse service may be responsible for pulling usage statistics of datasets in the data lake **102**. The meta pulse queries may query the pulse service to pull usage metrics of datasets.

**[0031]** In some embodiments the data lake **102** may include a database that stores dataset usage metrics. As shown in **FIG.3**, an exemplary database table **300** may showcase statistics of the one or more actions performed on dataset, such as Read, Write, Execute, Legal Hold and the like. Upon receiving the meta pulse queries, the pulse service may get the usage metrics from the database. In some embodiments, the usage metrics may be collected through Ranger Logs, Hive meta store, yarn logs and the like. In response to the meta pulse queries, the pulse service in the data lake **102** is configured to provide response pulses or metadata pulses **204** indicating the one or more actions performed on the datasets stored in the data lake **102**. In some embodiments, the pulse service may return a PULSE 1 or no PULSE 0, as response pulses **204**, based on the one or more actions. As an example, the pulse service may calculate a pulse score for one or more actions. The score may be calculated using the method given below,

F(x) = I(x), where x represents a dataset and F(X) represents the pulse score, I(x) represents number of actions such as Read, Write, Execute, Legal Hold. In an alternate embodiment, the pulse score may be calculated considering each action separately, for example the pulse score may be F(x) = r(x) + e(x), where r(x) and e(x) represents the number of read and execute actions.

**[0032]** In some embodiments, if dataset is qualified for a legal hold, then F(x) is set to 1. A legal hold is a temporary immutability policy that may be applied for legal investigation purposes or general protection policies. A legal hold stores data in a Write-Once, Read-Many (WORM) formats until the data explicitly cleared. In some embodiments, if read or execute action is happened on the dataset, counts of read or execute may be added up to generate a total pulse score. Upon generating the total pulse score, the total pulse score may be compared with a predetermined threshold. If the total pulse score is greater than the predetermined threshold, then the pulse service may return a PULSE 1, else PULSE 0. The predetermined threshold may vary from dataset to dataset based on certain conditions such as the number of users and usage. As an example, if there are 50 users in an organisation, the predetermined threshold may be set to 50 for certain datasets. Further, in some embodiments, the pulse service may return PULSE 1 if the dataset is qualified for a legal hold. The pulse service may return PULSE 1 if a read or execute action happens for a dataset. Also, the pulse service may return PULSE 0 if only write action occurred no other action occurred. The pulse service may return the pulse scores as shown below:

**Pulse Service:**

Request:

/pulseApi/{dataset}/{dt}

Response:

Inactive Pulse

{
Pulse:0
lastAccessTime: ‘01/01/2020 12:10:12’
reads: 0,
legal: 0,
write: 1,
pulse_score: 0
}

Active Pulse

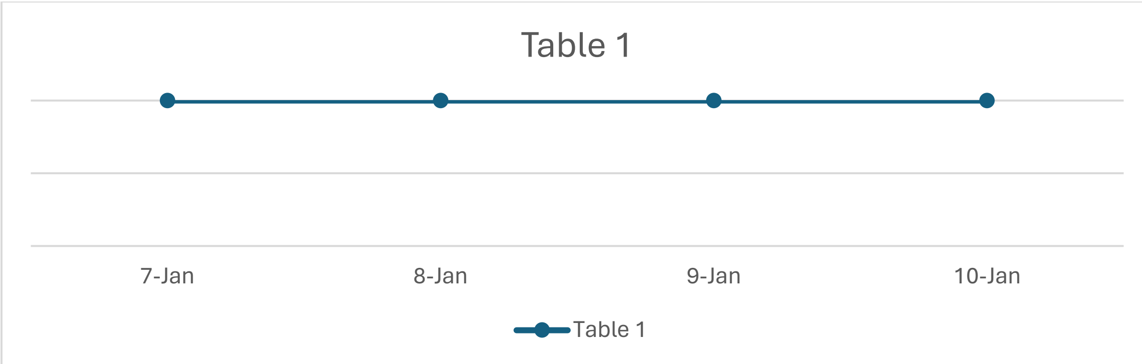
{
Pulse: 1
lastAccessTime: ‘05/15/2023 00:00:00’
reads: 500,
legal: 1,
write: 1,
pulse_score: 500

}

**[0033]** In an embodiment, upon receiving the response pulses **204**, the metadata hub **202** may be configured to perform one or more actions on the datasets stored in the data lake **102**, based on the response pulses **204**. The one or more actions may include deprecation of unused datasets by the deprecator system in the metadata hub **202**, using a decision tree model algorithm. As an example, if a given dataset has not been accessed for a set period of time, then the metadata hub **202** may record the pulse as 0 and may mark the dataset for deprecation.

**[0034]** **FIG. 4** shows an exemplary API **400** of a system for identifying stale data sets in the data lake **102** for deprecation. The system may be provided with a communication Application programming interface (API) to access the data lake **102**, send pulse queries and to receive the response pulse. In some embodiment, the communication API may comprise a high-speed local area network data interface and a storage array for storing data, where the high-speed local area network data interface may be connected with an external network equipment, the network equipment may be an external programmable data collector and a server. The API may include functionalities to adopted for the data access of the data lake **102**, the programmable data collector may actively store the collected data into the storage array, and the server may actively acquire the data from the data lake **102** or writes the data into the storage array. In some embodiments, the data lake **102** may realize data reading and writing and information exchange with the programmable data collector and the server through a data access mode of an API based on an http network protocol.

**[0035]** In some embodiments, the API may provide a pulse check option to the user. The user may select the pulse check option, to send meta pulse queries to the data lake **102**, where the meta pulse queries are used to pull usage metrics/statistics of datasets, which includes identifying one or more actions performed on the datasets stored in the data lake **102**. In response to the meta pulse queries the data lake **102** may provide response pulses **204** indicating the usage statistics of the datasets stored in the data lake **102**. As shown in **FIG.4**, the user may be able to see the usage statistics on the API and may be provided with options to select one or more actions, where the one or more actions may include, deprecate unused datasets. In some embodiments, the user may be provided an option to the usage statistics in detail, as shown in Table 1, below.




**[0036]** In an embodiment, the metadata hub **202** may include a deprecator system, that uses an AI (artificial intelligence) Model for deprecation of unused datasets. The metadata hub **202** may be configured to constantly send meta pulse queries to the data lake **102**, where the meta pulse queries are used to pull usage metrics/statistics of datasets. In response to the meta pulse queries the data lake **102** is configured to provide response pulses **204** indicating the one or more actions performed on the datasets stored in the data lake **102**. Upon receiving the response pulses **204**, the metadata hub **202** may be configured to perform one or more response actions on the datasets stored in the data lake **102**, based on the response pulses **204**. The one or more response actions may include, deprecate unused datasets by the deprecator system in the metadata hub **202**, using a decision tree model algorithm.

**[0037]** In some embodiments, the deprecator system may also use machine learning models to decide whether execute deprecation of unused datasets automatically. The response pulses **204** indicating the one or more actions performed on the datasets stored in the data lake may be used to feed into the AI models to deprecate unused datasets automatically. The models (AI models or machine learning models) may be trained converting and transforming a variety of inconsistent and incoherent supervised and unsupervised training data for decision tree model received by the system, as electronic data files. The data values in each data field in the inconsistent and incoherent supervised and unsupervised training data may compared and corrected according to the user preferences and a predefined data dictionary of valid data values. As an example, the data for training the models may look like as shown in Table 2,

| Table | Pulse | Pulse Score_avg | Legal | Table_type | Function | Days | Deprecate |
|---|---|---|---|---|---|---|---|
| T1 | 0 | 0 | 0 | Code | Finance | 30 | 1 |
| T2 | 1 | 1 | 1 | Fact | Auth | 30 | 0 |
| T3 | 1 | 100.5 | 0 | Fact | CS | 60 | 1 |

**Table 2**

**[0038]** As, shown in Table 2, the Table 2 may include, different the database tables T1, T2, and T3 and also includes one or more data associated with different datasets. Initially the column deprecated is updated using user inputs from metadata hub **202**. The models may use certain percentage of this available data to prepare the model and may use certain percentage of the available data to test the predictions. In some embodiment, upon training the models, unused datasets may be deprecated by the deprecator system in the metadata hub **202**, using a decision tree model algorithm.

**[0039]** **FIG. 5,** shows an exemplary flow diagram of execution of deprecation **500** of table T1 (of **Table 2**), using a decision tree model algorithm. The decision tree model algorithm may initially check (1) whether the dataset is in legal hold. If the dataset is in legal hold, the deprecator system of the metadata hub **202** may not execute deprecation of table T1. If not, (2) the decision tree model algorithm may (3), (4) check the type of dataset. Upon deciding the type of the dataset, the deprecator system may check the pulse score associated with the data set, to determine whether pulse score associated with the data set is greater or less than a predetermined threshold. The predetermined threshold may vary from dataset to dataset. As shown in the FIG.5, the (5) predetermined threshold for type code is set as 1000 and the (6) predetermined threshold for type fact is set as 500. If the pulse score associated with the dataset is greater than the predetermined threshold, the deprecator system of the metadata hub **202** may not execute deprecation of table T1. If the pulse score associated with the dataset is less than the predetermined threshold, the decision tree model algorithm may (7) check the function associated with the data set and based on the function the deprecator system of the metadata hub **202** may, whether to execute deprecation of table T1. As shown in the example, the function is finance and deprecation of table T1 is (8) not executed. Based on the output of the decision tree model algorithm, the tables which are not eligible for deprecation, may be assigned a deprecation score 1 by the deprecator system. Thereafter, the deprecator system may choose all tables with deprecation score 1 and may put them up for deprecation with a backup time for set number of days, e.g., “30 days”.

**[0040]** **FIG. 6,** illustrates an exemplary flow diagram of a method for identifying stale data sets in the data lake **102** for deprecation. The method may include a metadata hub **202**, that may acts as a central repository. The metadata hub **202** may also include a deprecator system, which uses an AI (artificial intelligence) model to deprecate unused datasets. The metadata hub **202** may be communicatively connected to a large-scale data repository, also known as a data lake **102**.

**[0041]** **At block 602**, the data usage store may include a physical dataset. The physical data set may have its logical metadata stored in the data lake **102**.

**[0042]** **At block 604**, meta pulse queries may be sent to the data lake **102**, where the meta pulse queries may be used to pull usage statistics of datasets, which includes identifying one or more actions performed on the datasets stored in the data lake **102**.

**[0043]** **At block 606**, the pulse API, may provide a pulse check option to the user. The user may select the pulse check option, to send meta pulse queries to the data lake **102**, where the meta pulse queries may be used to pull usage metrics/statistics of datasets.

**[0044]** **At block 608**, the metadata hub **202** may be configured to send meta pulse queries to the data store or the data lake **102**.

**[0045]** **At block 610**, the data store or the data lake **102** may receive the meta pulse queries.

**[0046]** **At block 612,** the deprecator may execute deprecation of unused datasets in the data store, using a decision tree model algorithm and the deprecator may inform the status of deprecation to the metadata hub **202**.

Advantages of the present disclosure

Embodiments of the present disclosure provides a method and a system for metadata maintenance in a data lake by identifying stale data sets in the data lake for deprecation.

- The present disclosure facilitates, automatic deprecation of inactive datasets.
- Using meta pulse queries organisations may derive insights on true and active datasets.
- The present disclosure enables, automatic deprecation of inactive datasets, thus reducing storage and helping organisations to manage and mine only alive datasets.

## ABSTRACT

## METHOD AND SYSTEM FOR METADATA MAINTENANCE IN A DATA LAKE

The present disclosure provides a method and a system for identifying stale data sets in the data lake for deprecation. The system comprises a metadata Hub, having a processor and a memory, and may act as a central repository. The metadata hub may also include a deprecator system or a deprecator, that uses an AI (artificial intelligence) model to deprecate unused datasets. The metadata hub may be communicatively connected to a large-scale data repository, also known as a data lake. The metadata hub may be configured to control the flow of ingest data from any relational database into the data lake. The data lake may include any data storage technology for offering storage for all data types. The metadata hub may be configured to send meta pulse queries to the data lake, where the meta pulse queries are used to pull usage stats of datasets, which includes identifying one or more actions performed on the datasets stored in the data lake. In response to the meta pulse queries the data lake is configured to provide a response pulse or metadata pulses indicating, the one or more actions performed on the datasets stored in the data lake. Upon receiving the response pulses, the metadata hub may be configured to perform one or more response actions on the datasets stored in the data lake, based on the response pulses. The one or more response actions may include deprecation of unused datasets by the deprecator system in the metadata hub, using a decision tree model algorithm.

**FIG.2**

100

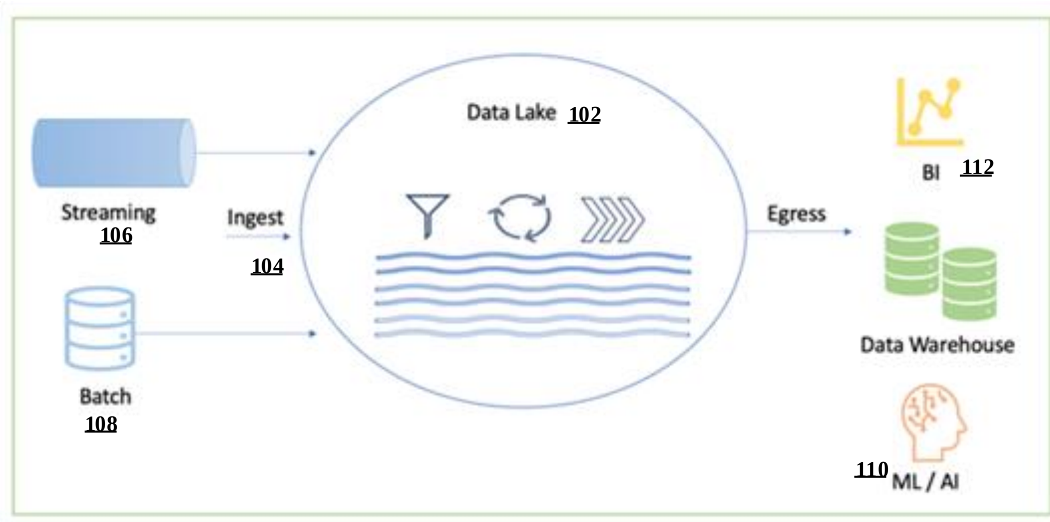



**FIG.1**

200

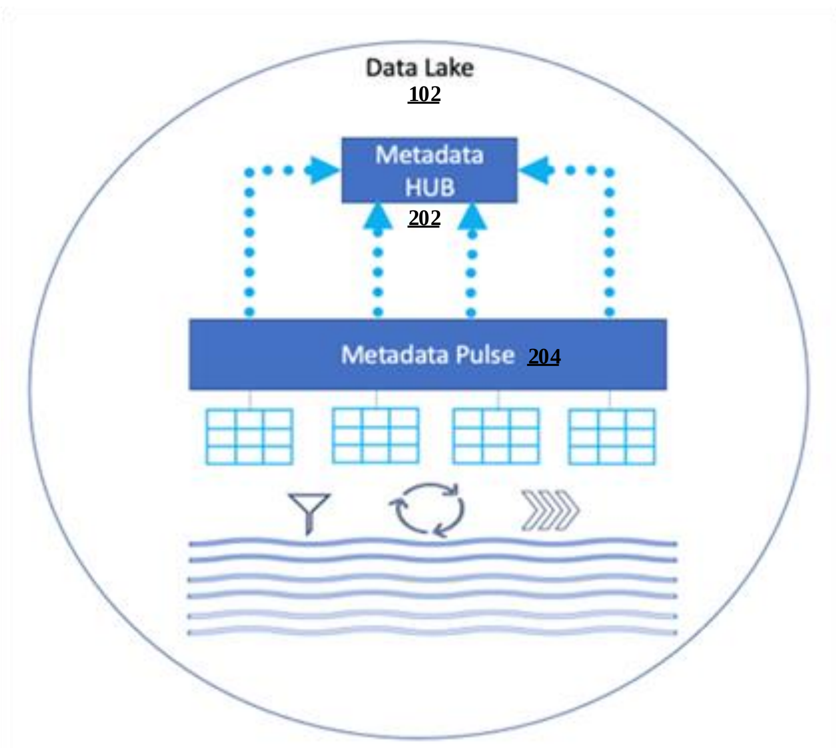



**FIG.2**

300

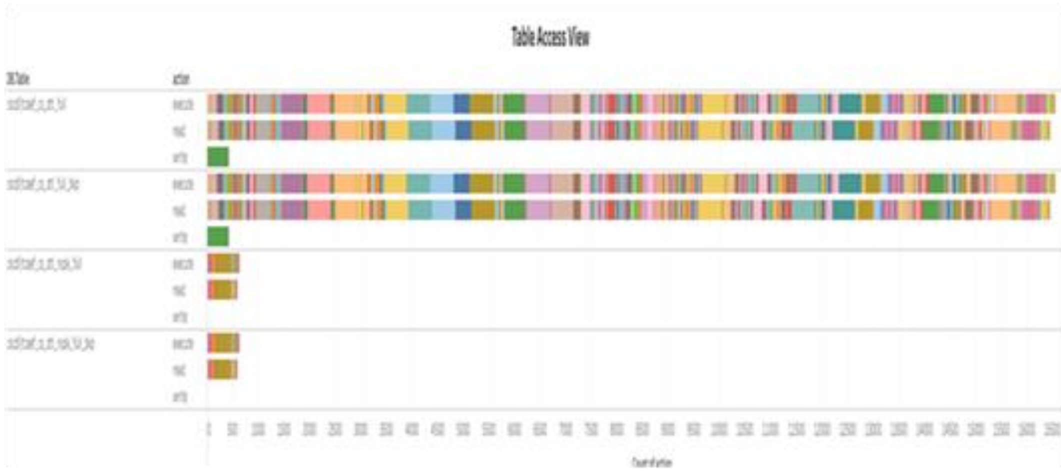



**FIG.3**

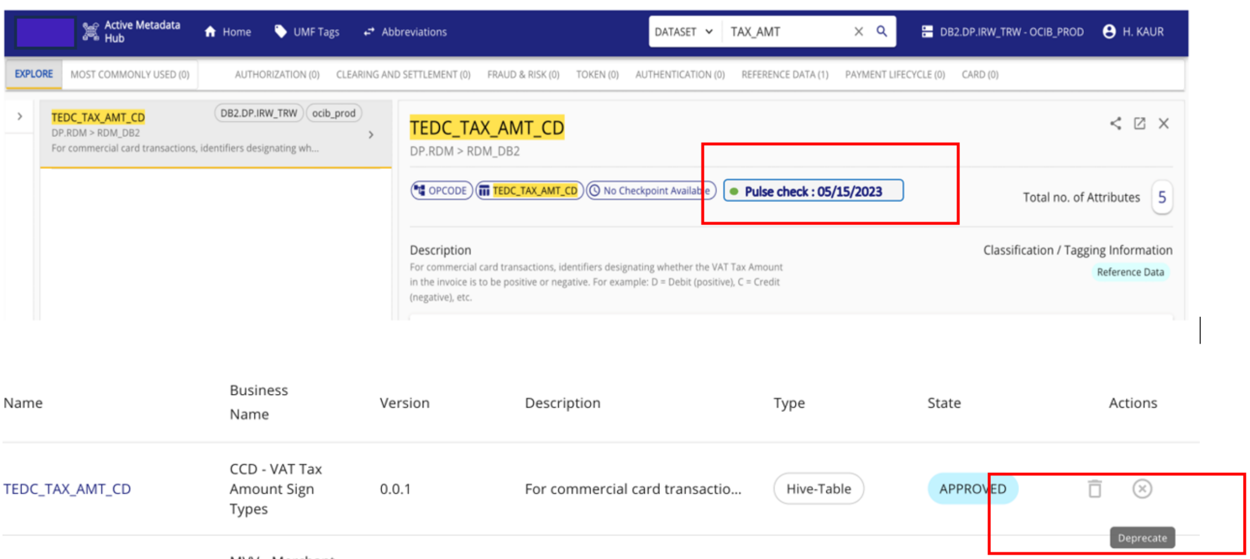



**FIG.4**

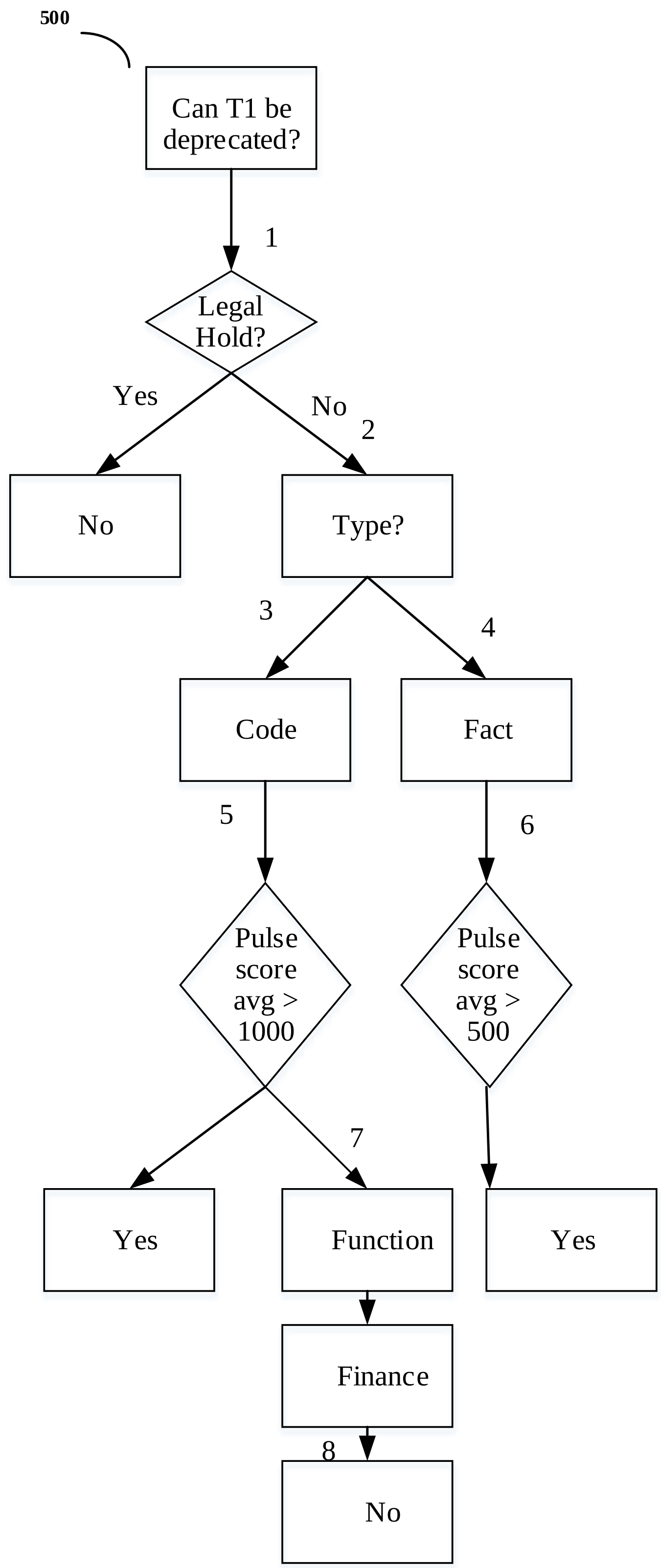

500
Can T1 be deprecated?
1
Legal Hold?
Yes
No
2
No
Type?
3
4
Code
Fact
5
6
Pulse score avg > 1000
Pulse score avg > 500
7
Yes
Function
Yes
Finance
8
No

**FIG.5**

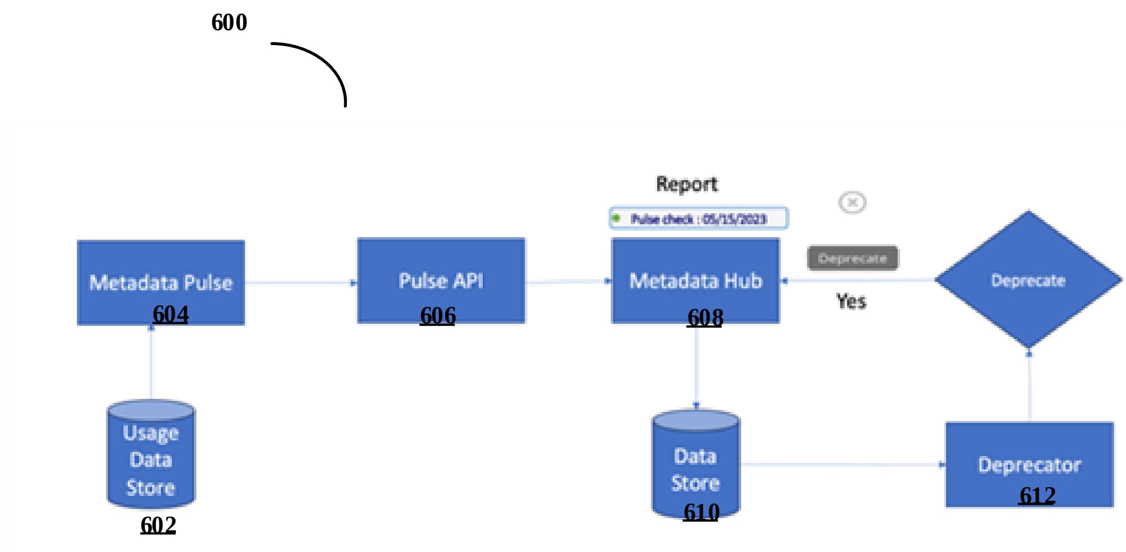

600
Report
Pulse check : 05/15/2023
Metadata Pulse
604
Pulse API
606
Metadata Hub
608
Deprecate
Yes
Deprecate
Usage
Data
Store
602
Data
Store
610
Deprecator
612

**FIG.6**