Data Lake Ingestion Management

Yan ZHAO\textsuperscript{1,2,4}, Imen MEGDICHE\textsuperscript{1,3}, and Franck RAVAT\textsuperscript{1,2}

\textsuperscript{1} Institut de Recherche en Informatique de Toulouse, IRIT-CNRS (UMR 5505),
\textsuperscript{2} Université Toulouse 1 Capitole, Toulouse, France,
\textsuperscript{3} Institut National Universitaire J-F Champollion, ISIS Castres, France,
\textsuperscript{4} Centre Hospitalier Universitaire (CHU) de Toulouse, France
\texttt{firstname.lastname@irit.fr}

\textbf{Abstract.} Data Lake (DL) is a Big Data analysis solution which ingests raw data in their native format and allows users to process these data upon usage. Data ingestion is not a simple copy and paste of data, it is a complicated and important phase to ensure that ingested data are findable, accessible, interoperable and reusable at all times. Our solution is threefold. Firstly, we propose a metadata model that includes information about external data sources, data ingestion processes, ingested data, dataset veracity and dataset security. Secondly, we present the algorithms that ensure the ingestion phase (data storage and metadata instanciation). Thirdly, we introduce a developed metadata management system whereby users can easily consult different elements stored in DL.

\textbf{Keywords:} Data lake \cdot Data ingestion \cdot Metadata management.

\section{Introduction}

Data Lake (DL), previously emerged only as a data repository \cite{3}, is today one of the most popular big data analytics solutions. A DL ensures data ingestion, storage of raw and processed data, preparation of data by different users and consumption of data for different types of analyzes \cite{9}.

The major issue of DLs is that they can easily turn into Data Swamps (DS). A DS, contrary to a DL, is invisible and inaccessible to users. It is impossible for data analysts to find, access, retrieve or even reuse needed data effectively \cite{2}. To meet this lack, metadata management is emphasized for efficient big data analytics \cite{4,5,14,23}, notably for data lineage. Data lineage allows analysts to have confidence and trustworthiness in data by providing information about data life-cycle in a DL from raw sources to different usages.

In addition, DLs are characterized by 'schema on read': raw data are ingested without transformation and are processed only when needed. Contrary to traditional ETL (Extract, Transform, Load) processes used for data warehousing and essentially dedicated to the transformation of structured data, ingestion processes of DL face the following specific challenges: connecting to different sources (Databases, Web servers, Emails, IoT, FTP, etc), integrating different data types (structured, semi-structured, unstructured), integrating 'small' data
transformation (File format change, File compression, Merging multiple small files), supporting multiple ingestion modes (batch, real-Time, one-time load), capturing changes and replicating them in the DL.

To meet these different challenges, it is essential to lean the ingestion processes of a DL associated with an efficient metadata management. As a matter of fact, data ingestion, as the first phase of data life-cycle in a DL, is the foundation of the whole big data analytics process. Current researches only partially meet this problematic. Some works focus only on one data type (structured [6], semi-structured [2], unstructured [12]), an ingestion mode (batch [2,12] and real-time [4]) or a specific domain [14]. Some works address metadata partially [6,2]. Some works introduced different types of metadata without proposing a formalized meta model [8]. To the best of our knowledge, there is no work that proposes a metadata model to fully characterize the ingestion process: data sources, ingested data and ingestion processes.

Therefore, in this paper, we propose a data lake metadata management system focusing on data ingestion. Our contributions are (i) a metadata model that includes different categories of metadata generated during the ingestion phase, (ii) metadata feeding mechanisms, and (iii) a tool that allows users to consult metadata to easily interoperate ingested real datasets and reuse them.

The paper is organized as follows, in section 2, we propose a complete metadata model related to the ingestion phase. In section 3, we introduce different algorithms to populate the metadata system. In section 4, we present how to find useful information through a web application And in section 5, we discuss different solutions of metadata management and data ingestion of DLs.

2 Metadata Model for Data Ingestion

Data ingestion is the first phase of the data life-cycle in a DL. This phase is the basis of DL management, it cannot be just a copy and paste of raw data for the reason that finding, accessing, interacting and reusing ingested data are all perceived as user needs. In addition, trust and confidence are also expected from users for data exploitation.

Therefore, metadata must be integrated to facilitate future use of DL and they must be collected starting from the ingestion phase. Hence, we present in this section an extension of our metadata model [10] with precise and complete metadata for the ingesting phase. The extended metadata model is shown in Fig. 1. Our model has the following advantages:

- It provides external data sources information (class DatasetSource) including data source name, type, location, administrator and owner, in addition, source of stream information can be added for IoT (Internet of Things) data. So that users can confirm the origin of the stream data and regroup data from the same source easily.
- It provides data ingestion process information (class Ingest and its association relationships) including source code, execution details and links to
Fig. 1. Conceptual Metadata Schema

the user, the upstream and downstream datasets. So that users can find the involved datasets and understand how data are ingested, reuse the ingestion process or modify the process when errors occurred during the ingestion phase. Note that for batch and real-time data, the start time (ingestionStartTime) and end time (ingestionEndTime) is the real start and end time of an ingestion process, however, the duration (definedDuration) is only required in case of real-time ingestion.

- It provides **ingested data information** including basic characteristics (class DatalakeDataset): schematic metadata (classes DLStructuredDataset, DLSemiStructuredDataset, DLUnstructuredDataset and their components) that help users to easily understand the structure of data; semantic metadata
(classes `Tag`, attribute `DatalakeDataset.description`) that help users understand the meaning of data.

– It provides **dataset veracity information** (classes `VeracityIndex`) that help users have a first glimpse of data quality. The veracity index is a composite index which combines different measures [11]: (i) objectivity/subjectivity, (ii) truthfulness/deception, (iii) credibility/implausibility.

– It provides **data security information** (classes `SensitivityMark`, `SensitivityLevel`) to protect sensitive or private data. Users can mark sensitive level for datasets, tables or attributes to limit data access.

– It provides **relationships between datasets** (classes `RelationshipDS`, `AnalysisDSRelationship`) so that when users search a dataset, they can find all the relevant data to enrich their analyzes. There are some predefined relationships that can be automatically detected by the system such as the similarity, correlation, containment and logical cluster. In addition, users can define relationships such as a common subject.

– It provides **global metadata** (class `GlobalDict`) which contains DL logs, shared information or public dictionaries among different datasets.

The presented meta-model is dedicated to the ingestion phase. It is a complete model which includes information about data sources, all characteristics of the ingested datasets and the ingesting processes. In addition, our model responds to both real time and batch ingestion. The model builds a good base for the data lake management and ensures the good efficacy and high efficiency for different users who work with the DL.

### 3 Data Ingestion Processes

The ingestion process in DL consists of two sub-processes: (i) store data in the data lake and (ii) complete and store corresponding metadata. We use algorithms to explain how to ensure the two sub-processes.

The sub-process **store data in the data lake** has as an objective to ingest source data into DL by different operation modes and store basic metadata. To do so, different steps need to be ensured: (i) Establish the source dataset connection and instantiate data source metadata (class `DatasetSource`), the connection information is entered manually. For this step, the function `connectDataSource()` in Algo. 1 is used. (ii) According to the ingestion mode (real-time/batch), store new dataset and its metadata (class `DatalakeDataset`) as well as ingestion process metadata (class `Ingest`). For this step, the function `ingestDataset()` in Algo. 1 is called.

The sub-process **complete and store corresponding metadata** has objective of facilitating the future data analysis by helping users to find, access, interoperate the needed datasets by giving users a first vision of dataset semantic and schematic information of chosen datasets. To achieve this objective, we need to complete and store not only the information of each single dataset but also the relationships among different datasets (in no particular order and can be parallel):
Data Lake Ingestion Management

Algorithm 1: Functions used to ingest datasets into a data lake

/* establish source dataset connection */

1 Function connectDataSource(dataSourceConnection, dataSourceType, dataSourceLocation, dataSourceName, dataSourceOwner, sourceOfSteam, owner):

2 if tryConnectDataSource(dataSourceConnection, dataSourceLocation) then

3 datasetSource ← instantiateClass('DatasetSource', createProperties('name', dataSourceName, 'type', dataSourceType, 'location', dataSourceLocation, 'owner', owner))

4 if sourceOfSteam then

5 createRelation('DatasetSource-SourceOfSteam', datasetSource, sourceOfSteam)

6 return datasetSource

7 else

8 return None

/* ingest dataset */

9 Function ingestDataset(datasetSource, ingestionComment):

10 ingestionProcess, ingestedDataset ← ingestDatasetInDL(datasetSource)

11 ingest ← instantiateClass('Ingest', getPropertiesIngest(ingestionProcess), 'comment', ingestionComment)

12 createRelation('DatasetSource-Ingest', datasetSource, ingest)

13 datalakeDataset ← instantiateClass('DatalakeDataset', getPropertiesDatalakeDataset(ingestedDataset))

– Instantiating definitional metadata concerns semantic and schematic metadata. Semantic metadata include a list of tags (class Tag) and a description (DatalakeDataset.description) of datasets and can be manually input by users. Schematic metadata of structured and semi-structured databases consist of the database model with tables/entities and attributes information. For unstructured datasets, the data format is detected. This task can be implemented according to the functions instantiatingSemanticMetadata() and instantiatingSchematicMetadata() which are introduced in Algo. 2.

– Instantiating security metadata consists of the security levels of different data granularity (dataset, table, column, etc). The security metadata is not necessary, they can be manually input or not, the default value is zero which means that all users can access the dataset. However, for some special domains, for example, hospitals, to protect the personal data of patients, the sensitivity metadata are mandatory. Concerning the population of metadata database, security metadata is input by users to ensure the protection of sensitive data.

– Instantiating inter-metadata concerns various relationships among datasets which are important for data finding, reuse and interoperability in a data lake. The inter-metadata can be automatically generated by the system (for instance, similarity and coherence between one dimension tables) or input manually by users. Note that the generation of inter-metadata is not limited in this sub-process, for example, the provenance information can be instantiated during the data storage phase as we mentioned previously. To populate the inter-metadata, the functions calculateRelationships() and inputRelationships() can be called (see Algo. 3).

– Instantiating veracity metadata is about calculating the veracity composite index which includes objectivity, truthfulness and credibility [11] of datasets.
Algorithm 2: Functions used to complete definitional metadata

```
/* instantiating definitional metadata - semantic */
1 Function instantiatingSemanticMetadata(datalakeDataset, datalakeDatasetDesc, datalakeDatasetTags):
2 addAttributes('DatalakeDataset', datalakeDataset, createProperties('description', datalakeDatasetDesc))
3 /* store tags for datalakeDataset */
4 allTags[] ← getAllTags()
5 foreach t ∈ datalakeDatasetTags do
6   if t in allTags[] then
7     tag ← getTag(t)
8   else
9     tag ← instantiateClass('Tag', createProperties('name', t))
10    createRelation('DatalakeDataset-Tag', DatalakeDataset, tag)
11 /* instantiating definitional metadata - schematic */
12 Function instantiatingSchematicMetadata(datalakeDataset):
13 if (datalakeDataset.type = "structured" or "semi-structured") then
14  entities[] ← getEntityClasses(datalakeDataset)
15  foreach e ∈ entities[] do
16    atts[] ← getAttributes(e)
17    entityClassProperties ← createProperties('name', getEntityName(e), getEntityStatistics(atts[]))
18    // function getEntityStatistics() returns an array including the name and value of each statistical metadata
19    entityClass ← instantiateClass('EntityClass', entityClassProperties)
20    createRelation('DatalakeDataset-EntityClass', datalakeDataset, entityClass)
21    foreach att ∈ atts[] do
22      if getAttType(att) = 'numeric' then
23        numericAttributeProperties ← createProperties('name', getAttName(att), getNumericAttStat(att))
24        attribute ← instantiateClass('NumericAttribute', numericAttributeProperties)
25      else
26        nominalAttributeProperties ← createProperties('name', getAttName(att), getNominalAttStat(att))
27        attribute ← instantiateClass('NominalAttribute', nominalAttributeProperties)
28        createRelation('EntityClass-Attribute', entityClass, attribute)
29    /* For each predefined RelationshipAtt we calculate the value of relationship between attributes */
30    analysisAttributes[] ← getAnalysisAttribute(atts[], relationshipAtts[])
31    foreach an ∈ analysisAttributes[] do
32      relationArr ← instantiateClass('AnalysisAttribute', createProperties('value', an.value))
33      createRelation('AnalysisAttribute-Attribute', relationArr, an.attribute1)
34      createRelation('AnalysisAttribute-Attribute', relationArr, an.attribute2)
35      createRelation('AnalysisAttribute-RelationshipAtt', relationArr, an.relationshipAtt)
36  else
37    addAttributes('DatalakeDataset', datalakeDataset, getDatasetFormat(datalakeDataset))
```
Algorithm 3: Functions used to complete inter metadata

/* for chosen dataset, calculate automatically relationships */
1 Function calculateRelationships(datalakeDataset):
   /* For each predefined RelationshipDS we calculate the value of relationship between datasets */
   2 analysisDSRelationships[] ← getAnalysisDSRelation(datalakeDataset, datalakeDatasets[], relationshipDSs[])
   3 foreach anDs ⊂ analysisDSRelationships[]
   4     relationDs ← instantiateClass('AnalysisDSRelationship', createProperties('value', anDs.value))
   5     createRelation('AnalysisDSRelationship-DatalakeDataset', relationDs, anDs.datalakeDataset1)
   6     createRelation('AnalysisDSRelationship-DatalakeDataset', relationDs, anDs.datalakeDataset2)
   7     createRelation('AnalysisDSRelationship-RelationshipDS', relationDs, anDs.relationshipDS)

/* for chosen datasets, users input a relationship manually */
8 Function inputRelationships(datalakeDataset1, datalakeDataset2, relationshipDS,
   dsRelationshipName, dsRelationshipDesc, dsRelationshipValue):
   9     analysisDSRelationship ← instantiateClass('AnalysisDSRelationship', )
   10    createRelation('AnalysisDSRelationship-DatalakeDataset', relationDs, anDs.datalakeDataset1)
   11    createRelation('AnalysisDSRelationship-DatalakeDataset', relationDs, anDs.datalakeDataset2)
   12    createRelation('AnalysisDSRelationship-RelationshipDS', relationDs, anDs.relationshipDS)

4 A Metadata Management System

To facilitate data discovery in a data lake, we have implemented a metadata management system. This system contains a user-oriented web application and a database to store metadata. The application, developed mainly in JavaScript and HTML/CSS, has an ergonomic graphical interface that allows users to easily interact with metadata. The database is a graph database (Neo4j) which has the advantages of good scalability, flexibility and maturity.

To illustrate the usage of the metadata application and set up an available example, we ingested different types of real open-source datasets (see table 1). The metadata graph database contains 2315 nodes and 2775 relationships. Where, regarding data ingestion, there are 7 nodes of DatasetSource, 7 nodes of Ingest, 55 nodes of EntityClass, 1197 nodes of Attribute and 74 nodes of Tag etc. Regarding the relationship detection, we created 3 datasets based on the CHSI cancer dataset and calculated the relationships between these datasets (1 node of RelationshipDS and 3 nodes of AnalysisDSRelationship) and the relationships between attributes in these datasets (5 nodes of RelationshipAtt and 62 nodes of AnalysisAttribute).

Moreover, the system allows users to easily find, consult datasets that are ingested in the DL via a web application. Data analysts who work with the DL can use the application to find useful information. To validate the interest of this metadata, we asked end users to use our tool. For instance, a data analyst needs to start a project concerning cancer analysis. He wants to find in the DL if there are existing datasets about cancer, he can easily search the key word 'cancer' in the application. Then he can see a list of datasets whose name, description

5 https://github.com/yanzhao-irit/data-lake-metadata-management-system
6 https://github.com/yanzhao-irit/data-lake-metadata-management-system/tree/main/example-metadata
or tags contain the key word (see the left result in Fig. 2). Moreover, he can choose different datasets then click on available tabs to get more details of chosen datasets to see if they are useful for his project or not. For instance, he can find the properties and lineage information of dataset to see what is the dataset about and how and when it is ingested in the DL (see Fig. 2). In addition, he can find more details of datasets attributes information and the relationships between different datasets (see Fig. 3).
5 Discussion

The data ingestion in DL is not a simple copy of data. The process is much more complicated as illustrated throughout the paper. Bringing a complete metadata system is the best solution to prepare a DL for data analytics, data lineage and making it compliant with FAIR principles [1]. To the best of our knowledge, there is no complete solution which includes all metadata of ingested datasets, data ingestion processes and developed metadata management tool.

Regarding the metadata management and in order to discuss our proposal compared to the state of the art approaches, we summarize in Fig. 4 the different types of metadata that should be addressed. This comparison includes the most relevant approaches which studied the metadata for DLs: MEDAL [12] that proposes a not formalized metadata model focusing on semi-structured and unstructured data; CM4DL [2] that focuses on the inter-metadata of semi-structured data; DL-Wrangling [13] that introduces a not formalized metadata model which contains global metadata; Constance [5] that focuses on data ingestion without proposing a formalized metadata concerning this phase, GEMMS [8] that concerns a post-hoc metadata management.
In Fig. 4 concerning the management aspects, we can observe that our approach is the only one that covers different types of ingested data (structured, semi-structured and unstructured) supported with a metadata model. Regardless its importance, the state of the art approaches do not precise the ingesting mode. Our approach satisfy both batch and real-time data ingestion. Both types of ingestion mode are crucial in data lake environment.

In addition, it is always important to be able to trace the data source used for the ingestion. External data source metadata allow users to check the provenance of ingested datasets so that they can find the data source in case of updating datasets. Five attributes were dedicated in our model for this subject but we have noticed that the other approaches neglected this point.

Moreover, concerning datasets metadata, we distinguish three types of metadata: intra-, inter- and global-metadata. Regarding intra-metadata, we have the originality of including sensitivity and veracity metadata to ensure the access control and data governance. Regarding inter-metadata, our model is more complete than GOODS and MEDAL with particularly the possibility for users to define their own dataset relationships. Regarding global metadata, we use a global dictionary for shared information among datasets and it can be simply used by research engines.

What’s more, concerning the ingestion process metadata, to the best of our knowledge, there is only GOODS [6,7] which address this point without a formalized processing model. For the ingestion process, program, execution and data flow metadata should be saved. The approach of Goods [6] saves reading and writing jobs. In our approach, we are more generic as we save source code URL, binary machine code URL and the whole environment. GOODS does not track the execution metadata while we back up ingestion start time, duration, output log, error log and further comments. Our approach is then compliant with lineage on processes.

Regarding the ingestion process, to the best of our knowledge, there are only a few solutions. [2] introduced metadata management for data ingestion by three phases (ingest, digest and exploit) for batch data. They focused on the extraction of inter-metadata from semi-structured data. [3] presented a data ingestion framework which consists of a data collector and a data integrator. They explained different approaches to bring data into a Hadoop data lake without integrating a formalized ingestion processing and a metadata model for this phase. [13] introduced data ingestion by three steps without a formalized processing.

These data ingestion solutions did not introduce a detailed ingestion processing model. Moreover, they did not include both the aspects of data ingestion processing and the metadata of data ingestion. Therefore, we proposed a more complete metadata model and a data ingestion processing that includes the interaction of the DL system and users. Currently, we are formalizing the other data lake processes (data processing and data access) using algorithms. After having demonstrated the feasibility of our approach, we intend to focus on performance optimizations of integrating a great volume of various types of datasets.
Today, data lakes are increasingly promoted as a big data analysis solution. In this paper, we focus on improving the efficacy and the efficiency of data ingestion. To the best of our knowledge, our work is the only one which proposes both a complete metadata model and the underlying processes for the ingestion phase.

Regarding metadata for data ingestion, we propose a complete model including information on data source, datasets and ingesting process. In addition, for datasets, our model contains inter and intra metadata to facilitate future data analysis. Regarding ingestion processes, we formalize the data ingestion processes including metadata generation using algorithms. Moreover, our inges-
tion processes are general enough for different structural types of data and for different ingestion modes. We validate our solution by implementing a metadata management system with real datasets and illustrate the system with a use case to show its ease of use and ergonomic interface.

Currently, we are completing our proposal with additional data lake processes (data preparation and data analysis). To be compliant with data lineage, we formalize these processes and identify the associated metadata. After having demonstrated the feasibility of our approach, we intend to focus on performance optimizations of our metadata management system.

References

1. FAIR Principles, https://www.go-fair.org/fair-principles/
2. Alserafi, A., Abelló, A., Romero, O., Calders, T.: Towards information profiling: data lake content metadata management. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). pp. 178–185. IEEE (2016)
3. Dixon, J.: Pentaho, Hadoop, and Data Lakes (Oct 2010), https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/
4. Gupta, S., Giri, V.: Data lake ingestion strategies. In: Practical Enterprise Data Lake Insights, pp. 33–85. Springer (2018)
5. Hai, R., Geisler, S., Quix, C.: Constance: An intelligent data lake system. In: Proceedings of the 2016 International Conference on Management of Data. pp. 2097–2100. ACM (2016)
6. Halevy, A., Korn, F., Noy, N.F., Olston, C., Polyzotis, N., Roy, S., Whang, S.E.: Goods: Organizing google’s datasets. In: Proceedings of the 2016 International Conference on Management of Data. pp. 795–806. ACM (2016)
7. Halevy, A.Y., Korn, F., Noy, N.F., Olston, C., Polyzotis, N., Roy, S., Whang, S.E.: Managing google’s data lake: an overview of the goods system. IEEE Data Eng. Bull. 39(3), 5–14 (2016)
8. Quix, C., Hai, R., Vatov, I.: Metadata extraction and management in data lakes with gemms. Complex Systems Informatics and Modeling Quarterly (9), 67–83 (Dec 2016)
9. Ravat, F., Zhao, Y.: Data lakes: Trends and perspectives. In: International Conference on Database and Expert Systems Applications. pp. 304–313. Springer (2019)
10. Ravat, F., Zhao, Y.: Metadata management for data lakes. In: European Conf. on Advances in Databases and Information Systems. pp. 37–44. Springer (2019)
11. Rubin, V., Lukoianova, T.: Veracity roadmap: Is big data objective, truthful and credible? Advances in Classification Research Online 24(1), 4 (2013)
12. Sawadogo, P., Kibata, T., Darmont, J.: Metadata management for textual documents in data lakes. In: 21st International Conference on Enterprise Information Systems (ICEIS 2019) (2019)
13. Terrizzano, I.G., Schwarz, P.M., Roth, M., Colino, J.E.: Data wrangling: The challenging journey from the wild to the lake. In: CIDR (2015)
14. Walker, C., Alrehanny, H.: Personal data lake with data gravity pull. In: 2015 IEEE Fifth International Conference on Big Data and Cloud Computing. pp. 160–167. IEEE (2015)