A generic metadata management model for heterogeneous sources in a data warehouse

Lamy Oukhouya^{1*}, Anass El haddadi², Brahim Er-raha¹, Hiba Asri¹

¹LIMA laboratory, Ibn Zohr University, Agadir, Avenue Tamsoult, Morocco

²AS laboratory, Abdelmalek Essaadi University, Al hoceima, Morocco

Abstract. For more than 30 decades, data warehouses have been considered the only business intelligence storage system for enterprises. However, with the advent of big data, they have been modernized to support the variety and dynamics of data by adopting the data lake as a centralized data source for heterogeneous sources. Indeed, the data lake is characterized by its flexibility and performance when storing and analyzing data. However, the absence of schema on the data during ingestion increases the risk of the transformation of the data lake into a data swamp, so the use of metadata management is essential to exploit the data lake. In this paper, we will present a conceptual metadata management model for the data lake. Our solution will be based on a functional architecture of the data lake as well as on a set of features allowing the genericity of the metadata model. Furthermore, we will present a set of transformation rules, allowing us to translate our conceptual model into an owl ontology.

1 Introduction

Use The main role of the decision-making system is to help decision-makers to effectively broaden their strategic decision-making within companies. To achieve this goal over the past 30 years, these systems have continued to evolve their architectures to effectively adapt to the evolution of data. Indeed, the volume, variety and speed of data production pose a real challenge to traditional storage systems, especially for data warehouses. A data warehouse is a structured, non-volatile historical storage system that makes data organized and oriented by subject for analytical purposes [1].

However, the data warehouse lacks the capacity to handle semi-structured and unstructured data, which is a typical feature of big data [2]. As a result, the concept of the data lake has been introduced into the business intelligence architecture as a big data storage solution, capable of serving as a heterogeneous data source for the data warehouse.

The data lake applies the “schema on read” strategy, from which the data is stored in a centralized repository in raw format and does not undergo any structuring, allowing for flexibility in the processing of the data, as well as acquiring a high speed of data ingestion [3]. However, the lack of schemas for the data can easily make the data lake incomprehensible and inaccessible [4]. For this reason, metadata management must be applied by the user to efficiently extract and analyze data.

Metadata management can be implemented in the data lake either by a model or an implementation [5]. Both need a good design strategy to discover, govern and exploit the data stored in the data lake. Indeed, such a system relies mainly on 1) a good architecture of the data lake to allow to efficiently capture the metadata, 2) a classification of metadata for well-organized metadata 3) a set of functionalities that the system must ensure to manage traceability, confidentiality, quality and aggregation of data.

These metadata features help structure and contextualize the data stored in the data lake. In addition, they allow the metadata model to be generic in the face of different use cases. Indeed, from the various metadata management model works [5] [6], there are so far 8 key features used to design a good metadata management system, namely, Semantic enrichment, data polymorphism, data versioning, usage tracking, categorization, similarity links, metadata properties, multiple granularity levels.

Concerning our work, these features are not sufficient in the situation where the data lake is used as a single source for the data warehouse. Indeed, the dynamics, the volume and the heterogeneous Big Data lead to the evolution of the schema of the sources, which affects the schema of the data warehouse. In addition, a change and adaptation must be applied to the data structure in accordance with changes in the analytical requirements of decision makers. Therefore, the “schema evolution” functionality is important in a metadata management system.

* Corresponding author: o.oukhouya@uiz.ac.ma

The objective of this article is to explore the data lake that can be efficiently exploited by the data warehouse by developing a complete conceptual model of metadata, integrating the 9 features and covering all areas of the architecture of the data lake. This article is divided into five sections. Section 1 gives a brief introduction. Section 2 presents the analysis and comparison of the works examined, this section is organized into 3 subsections: categories of data lake architectures, typologies of metadata management and models of metadata management. Section 3 presents our contribution to the architecture of the data lake, and the conceptual model of metadata management. Section 4 propose a Translation rule of the conceptual model in OWL ontology. Finally, we summarize our work with a brief conclusion and future perspectives.

2 Related work:

The use of the data lake as a heterogeneous data source Big Data takes place in the context of the modernization of data warehouses. This modernization is based on a hybrid architecture between these two technologies, allowing the data warehouse to gain scalability, increase speed and flow, reduce costs as well as the unification of BI and IA analyses. For the data warehouse to gain these benefits, the Data Lake must use metadata to solve the problems of contextualizing and structuring data. To do this, the metadata management model design requires 1) a data lake architecture to extract the metadata effectively and 2) a metadata classification to meet user requirements. In this section, we review, on the one hand, the data lake architectures, and on the other hand, the classification and the metadata management model. In addition, a comparison will be made on this work to extract the positive and negative points of each approach. This section is organized as follows: 2.1 data lake architectures, 2.2 metadata classifications and 2.3 metadata management models.

2.1 The Architecture of data lake

We have detected two visions of functional architecture. Multi-zone distinguished by 4 zones, raw data zone, process zone, access zone and governance zone [11]. The second vision is merged with the notion of data pools with 5 levels, raw data, analog data, application data, textual data and data archiving [12]. After we have roughly presented the different architectures of the data lake in the literature, we will be most interested in multi-zone architectures (technical and functional) [8,11-12], because they are better suited to the definition of the data lake [13]. This architecture [12] classifies the raw data of the transient data pool according to three categories analogous data pools for semi-structured data, applicative data pool for structured data, and a textual pool for unstructured data. These pools are equipped with a processing, presentation and analysis mechanism. Once the data is no longer used, they are saved in the 5th

archived data pool. This architecture has the advantage of facilitating and accelerating analyses, but it suffers from the loss of raw data because it is deleted once transferred from the raw pool to the other pools. Moreover, this absence of raw data is contradictory with the concept of the data lake, which ingests and stores the raw data for a future request for analysis by the user.

As for the architecture of [8], the data is ingested in its raw format in the transient loading zone under basic quality control. Subsequently, this data undergoes a process of successive processing in 3 areas; the raw data area integrates and structures the data. The trusted zone cleans and normalizes the data. The Discovery Sandbox Zone contains data dedicated to analysis. The consumption zone allows experts to explore the data and the sixth zone is specific to govern the data lake. This architecture is perfect for the context of the data lake. However, the transit of data in the six zones creates problems related to the traceability of the data.

Concerning the latest architecture [11], the data is stored in their native formats in the raw data area either in batch or / and in real time. Afterwards, at the level of the process area, the raw data is processed according to the analytical needs of the users. This area provides processing, such as aggregation, selection, join and projection. Once the data is processed, they are all transferred to the access zone, which is responsible for scanning and assigning access to the data. The last area is governance, which applies to all other areas. Its mission is to manage metadata, ensure the security, quality and life cycle of data. This architecture is advantageous by contributing to [8], because the raw data are stored in a specific area; and by contributing to [12] by controlling the lineage, because the data passes between 3 layers only. However, access to data with this architecture is only possible for data stored in the process area.

Based on this comparison, we note that each of these architectures has limits and that a new architecture is needed to solve the following problems: the data lineage caused by the number of zones, deletion of the raw zone data, and data access performed only from a single area.

2.2 Topology of metadata

We have distinguished three typologies used to classify metadata in the data lake:

1) The first typology was drawn from the context of data warehouses, described by the authors [14] by three categories:

- Technical metadata: groups together the type of data, the format and the structure.
- Operational metadata: manages news on data processing and lineage.
- Business metadata: includes governance processes, trade names and descriptions.

2) The second typology was used for the 1st time in the context of the data lake by [20], and later completed in the work [11]. Metadata is classified into two categories:

- Intra-metadata: identify and classify each dataset by 7 subcategories: the characteristics of the data, definition, quality, security, navigation, access and lineage.
- Inter-metadata: describe and classify the relationships between data sets by four types: data set containment, provenance, logical cluster and content similarity and Dataset containment.

3) This third typology adds a third category to the previous classification as well as a change on the use of the dataset, which has been replaced by the notion of object [5]. Their description is as follows:

- Intra-object metadata: refers to metadata relating to a specific object. Four types are distinguished: properties, summaries and previews, version and presentation, semantics of metadata.
- Inter-object metadata: Describes the relationships between objects by three types of metadata: object groupings, similarity links, and kinship relationships.
- Global metadata: contextualizes all the data stored in the lake in order to optimize their analysis. This category distinguishes 3 types: resources

After examining these three classifications, we notice that each category has limitations. For example, the first classification does not cover the relationships between datasets, which is an important aspect of analysis in the data lake. In the third classification, when grouping objects, one must pay attention to the functions chosen as this impacts the categorization of sources and types of data. However, we find the extensibility of the second classification more interesting than the other two, and for us, the use of datasets is more expressive for the data lake than the objects.

2.3 Managing metadata in the data lake

The first work [15] proposes a graph-based metadata model to unify and integrate heterogeneous data sources into a data lake. The particularity of this model is that it gives a partial structure to unstructured data before integrating it into the data lake. The metadata functionalities proposed by this model are mainly and generation of data semantics, definition of the similarity relationship between different data as well as the possibility to keep several representations of the same data. In [16], the authors propose a conceptual metadata schema described by a UML class diagram and validated by an implementation for the metadata management system on two DBMSs including the relational and the NoSQL graph. This diagram was presented based on an intra-metadata (for each dataset) and inter-metadata (presence of relationships between datasets) classification that includes all processes and exploration in a data lake. This classification contains all the major functionalities of a metadata management system; therefore, the authors excluded the description of the data context. Another work [5] describes a metadata management model covering the whole functional architecture of a data lake. Based on a graph

model, this system uses the notion of object instead of dataset and a classification into three types, intra-object, inter-object and global. In addition, it allows contextualizing the data in order to facilitate its analysis. More recently, this approach [17] consists of a base model capable of integrating any fundamental entity and relationship to provide any metadata management use case. In addition, this model uses 3 extensions, each of which handles metadata according to: 1) the different fields of the data lake, 2) the finest level of granularity, and 3) the category of the metadata. The authors agreed that their model is more generic by contributing to the work [5]. This is due to the absence of granularity in their model. For this reason, the authors presented an evolutionary version of their model named GOLDMEDAL[6]. This model is based on the principles of generalization of the 3 concepts seen in [7], namely; version and representation, transformations and updates, and finally link similarity generalized by the global concept of link.

Table 1 below summarizes this study by contributing the phases covering metadata, the type of classification used, the 8 functionalities of metadata, namely, Semantic enrichment (SE), data polymorphism (DP), data versioning (DV), usage tracking (UT), categorization (CG), similarity links (SL), metadata properties (MP), multiple granularity levels (GM). We have also summarized this study by the implementation model of the metadata management system.

Table 1. Comparison of metadata management systems for data lakes

Characteristics		[15]	[16]	[5]	[17]	[6]
Phases covering metadata Management	Ingestion	x	x	x	x	x
	Preparation			x	x	x
	Process		x	x	x	x
	Exploration		x	x	x	x
Classification types	Inter/Intra		x		x	x
	Inter/Intra/Global			x	x	x
	Bus./Techcl./Opera	x			x	x
Metadata features	SE	x	x	x	x	x
	DP	x	x	x	x	x
	DV		x	x		x
	UT		x	x	x	x
	CG		x	x	x	x
	SL	x	x	x	x	x
	MP		x	x	x	x
	GM	x			x	x
Implementation models	Relational		x			
	XML					
	RDF	x				
	JSON		x		x	x

From this table we can see that the model [6] is the most generic compared to the other works with all the functionalities, unlike the model [15] which is the least expressive for the management of metadata with four

functionalities. The other models have seven features; [17] does not support data version, [5;16] does not cover granularity. However, the evolution of data support in [16,5-6] covers only the aspect of the transformation of the data from one version to another. Moreover, the evolution of the structure of the data during the ingestion is not supported by these works. These two concepts have always been confused in the literature because they are closely related, but each has its own approach. Indeed, the evolution of a diagram is only applied if there is a transformation between two versions of data allowing for only one current version of the data schema. While the data version transfers the data from the existing schema to a new schema in order to apply the necessary modifications in it. Therefore, after clarifying the two concepts, we see that the models examined above are not generic enough to handle a use case on the evolution of data schema.

This theme has been widely investigated in the context of data warehouses mainly because the data schema must evolve over time to meet the new requests of decision makers, as well as adapt to the heterogeneity of data sources, unlike the data lake in which research is scarce. Indeed, this work presents the “constant” tool for the management of metadata in the data lake [18]. The architecture of this system is based on 3 layers: ingestion, maintenance and query. The first layer ingests the data raw from heterogeneous sources for native data storage in the data lake. This layer also includes extractors capable of detecting and adapting to new data sources. The second layer enables metadata discovery and matching, and the final layer provides quality control and user interference through a Request Rewrite Engine. In another work [19], which is in the context of using the data lake as a heterogeneous source of the data warehouse, the authors present a metadata management model to ensure semantic enrichment, categorization of data, the properties of the metadata, the granularity of the data as well as the evolution of the schema. The work focuses on the use of 5 algorithms to adapt the structure to the changes detected at the level of: data sources, metadata properties, dataset, data attributes, and finally at the relationship level. It should be noted that we limited ourselves to studies processing the evolution of the schema (SEv) in the context of the management of metadata in the data lake.

The table 2 below compares all of the work seen in this subsection with respect to the functionality supported by the model or the metadata system, by using the features as a basis for comparison

Table 2. Overview of the functionalities offered by a metadata management system

Works Features	[15]	[16]	[5]	[17]	[6]	[18]	[19]	Our model
SE	x	x	x	x	x	x	x	x
DP	x	x	x	x	x		x	x
DV		x	x		x			x
UT		x	x	x	x			x

CG		x	x	x	x			x
SL	x	x	x	x	x			x
MP		x	x	x	x		x	x
GM	x			x	x		x	x
SEv						x	x	

The table 4 confirms that so far, no model supports all of the 9 features, and that despite the importance of integrating schema evolution into metadata management, it is neglected by most of the work. In the next section, we will propose our conceptual model of metadata management, making it possible to express genericity in the data lake with the 9 functionalities already seen previously.

3 The Proposed Metadata Management Model : Architecture & Conception

As we saw earlier, the design of a generic metadata management model is based on a reliable data lake architecture, and metadata classification to meet user requirements. We also studied that a metadata management model is said to be generic, when it covers a set of features capable of handling any use case in the data lake.

In this section, we will present our contribution by contribution to 1) the design of a data lake architecture covering all the limits seen in subsection 2.1, 2) a generic metadata management model covering the set of functionalities seen in subsection.

3.1 Functional architecture of the data lake

The study that we established in the present section revealed that the current architectures suffer from the following limitations: The absence of the raw area to store and save the raw data, the loss of traceability on the data caused by an excessive number of Data Lake areas; data access is only performed from a single data lake area. To overcome these limits, we suggest in Figure 1, our functional architecture of the data lake, which contains 5 zones, namely:

- Raw data management area: it allows for integrating and ingesting data in the data lake in native format.
- Data processing area: in batch or streaming, this area is used to process and transform data using processes such as join, aggregation, selection, projection.
- Data Trust Zone: This zone is used to store highly clean data. It can be used as a source to feed a data warehouse.
- Data access area: this area allows the user to access all areas of the data lake, either to feed it from the trust area, or to apply analytical processes.
- Governance zone: it is applied to all other zones. It is responsible for ensuring data security and confidentiality, data quality, metadata management and master data management.

Figure1. Functional architecture of the data lake

3.2 The conceptual model of metadata management

In this section, we have used the classification presented in section 2 [11] to design our generic metadata management conceptual model. This model, as displayed in figure 1, treats a data set by the dataset concept which describes the metadata that can be stored either in a data source or in the data lake. The metadata in the data lake is classified into three categories, structured, semi-structured and unstructured, each of which uses an enumerated class to collect metadata according to the specific format for that category. The granularity of these types of formats is applied by the "typeItems" enumeration, which allows data items to be represented at several granular levels; for example, in the structured category, a relational schema can be expressed by tables, a domain, lines, and so on. Likewise, the JSON format of the semi-structured category can be described according to the granular level Key and value. In the case where the diagram has undergone an evolution over time, 4 elements are concerned by the change, namely, the dataset, the data sources, the data lake, the data items, and finally the relationships; inter and intra dataset. All of these changes are kept in the "change" class which makes it possible to store the metadata related to this change, such as the date of the change, the data source, the relationships, the dataset affected, the label and the new values/type of the attribute.... Each change on the stored metadata involves adding, deleting or updating, which are maintained by the "Process" class. These two classes will allow us in the implementation part of the model to define and automate the change of the schema when the requirements and data sources evolve. For security and quality metadata, they are also managed by the metadata model. In the case of data access control, for example, processing or consultation of data via an application is performed only if the user undergoes access control

applied by the class "Access Role" on its role as well as the types of privileges assigned by the "PLTtype" list. The semantic description is performed by the Keyword class, allowing the user to detect the most relevant datasets. In addition, our model manages the metadata set related to the processing and ingestion of data in the data lake.

4. Implementation of the Proposed Model

4.1. Translation rule of the conceptual model in OWL ontology:

In the previous section, we used the UML formalism to present our conceptual metadata management model for data lakes. This modeling language, adopted for several years by developers, provides a precise and accurate description of the logical level of the information system [21]. However, since it is a semi-formal language, UML suffers from ambiguity in its semantics [22]. On the other hand, and with the emergence of Big Data, ontologies are increasingly used, allowing to assign a formal semantics to a conceptual representation, so share the UML can be used as a basis for the design of ontologies [23]. In addition, due the similarity between the elements of the two formalisms, a transformation can be performed from a UML class diagram to an OWL ontology. In this sense, this section presents a set of rules to transform our conceptual metadata model into an OWL2 ontology. It should be noted that the ontology represented by the OWL code in Figure 3, will be detailed and tested in another work, using the ontology editor Protégé 4.3.0.

the table 4 below, present the rules for transforming our UML class diagram into OWL ontology, according to the UML elements extracted from figure 2, namely, classes, attributes, classes, attributes, enumerations, aggregation, composition and class of association.

Figure2. Class diagram of metadata conceptual model

Table3.Transformation Rule UML elements into an OWL Ontology

	UML Element	Description Rules	Transformation Rules
The Class & Attributes	<pre> classDiagram class Dataset { DS_ID: String DS_Name: String DS_Descp: String } </pre>	For each UML class transformed into an OWL ontology, the declaration axioms are defined in terms of the class, attributes, domain, and the property range of the data for the attributes [24-25].	Declaration(Class(:Dataset)) Declaration(DataProperty (:DS_ID)) Declaration(DataProperty (:DS_Name)) Declaration(DataProperty (:DS_Descp)) DataPropertyDomain(:DS_ID :Dataset) DataPropertyDomain(:DS_Name :Dataset) DataPropertyDomain(:DS_Descp :Dataset) DataPropertyRange(:DS_ID xsd:string) DataPropertyRange(:DS_Name xsd:string) DataPropertyRange(:DS_Descp xsd:string)
Binary association	<pre> classDiagram class Acces_info { AC_ID: String AC_privilege: TypeRL AC_time: datetime } class Acces_Role { AR_ID: String AR_role_ID: String AR_acces_ID: String } Acces_info "0..*" -- "1..*" Acces_Role </pre>	A binary association is transformed into OWL by declaring the property object axioms, the property object domain and range for the association ends, and an inverse object property that shows the relationship between the two classes [26]. According to [27], if the end-of-association names are not defined, the class name is used with lowercase letters.	Declaration(Class(:Acces_Role)) Declaration(Class(:Acces_info)) Declaration(ObjectProperty(:acces_role)) Declaration(ObjectProperty(:acces_info)) ObjectPropertyDomain(:acces_info :Acces_Role) ObjectPropertyDomain(:acces_role :Acces_info) ObjectPropertyRange(:acces_info :Acces_info) ObjectPropertyRange(:acces_role :Acces_Role) InverseObjectProperties (:Acces_Role :Acces_info)
The generalization « Disjoint,Full »	<pre> classDiagram class DataLake_Dataset { DL_ID: String DL_Items: String DL_Dataset: String } class Structured_data { SD_DB_ID: String .. } class Semi-Structured_data { SSD_Instances: String .. } class Unstructured_data { UD_Instances: string .. } DataLake_Dataset < -- Structured_data DataLake_Dataset < -- Semi-Structured_data DataLake_Dataset < -- Unstructured_data </pre>	The type generalization (disjoint, full) is transformed into OWL by the use of SubClassOf for subclasses, and the type of generalization between subclasses by DisjointWith[28].	Declaration(Class(:DataLake_Dataset)) Declaration(Class(:Structured_data)) Declaration(Class(:Semi-Structured_data)) Declaration(Class(:Unstructured_data)) SubClassOf(:Structured_data :DataLake_Dataset) SubClassOf(:Semi-Structured_data :DataLake_Dataset) SubClassOf(:Unstructured_data :DataLake_Dataset) DisjointWith(:Structured_data :Semi-Structured_data) DisjointWith(:Semi-Structured_data :Unstructured_data)

Table4.Transformation Rule UML elements into an OWL Ontology “suite”

	Elements UML	Description	Transformation Rules
The enumeration type		<p>An enumeration represents a new data type defined by the user, whose transformation into OWL is performed by the dataOneOf syntax. This transforms the attribute of the UML class from the enumerated data type to the property and range of the data type, the latter representing the enumerated attribute data type [28-29].</p>	<pre> <Declaration> <DataProperty IRI= "#TypeRole"/> </Declaration> <DataPropertyRange> < DataProperty IRI= "#TypeRole"/> <Datatype IRI="#TypeRole_enum"/> </DataPropertyRange> <DatatypeDefinition> <Datatype IRI="# TypeRole_enum"/> <DataOneOf> <Literal datatypeIRI="#PlainLiteral">Selection</Literal> <Literal datatypeIRI="#PlainLiteral">Insert</Literal> <Literal datatypeIRI="#PlainLiteral">Read</Literal> <Literal datatypeIRI="#PlainLiteral">Update</Literal> </DataOneOf> </DatatypeDefinition> </pre>
The N-ary association		<p>The N-ary association cannot be translated into OWL. According to [30], the transformation of this association can be done in a partial way by creating a new class N and its properties. Then, the set is transformed according to the rules mentioned in the binary association, provided that the old classes are SubClassOf the N-art class. Similarly, the association class is transformed by applying the rules of binary association [26-27].</p>	<pre> Declaration (Class(:DataLake_Dataset)) Declaration (Class(:DataSource_Dataset)) Declaration (Class(:Ingestion)) Declaration (Class(:User)) Declaration (Class(:Acquire)) Declaration(ObjectProperty(:DL_DS)) Declaration(ObjectProperty(:DS_DS)) Declaration(ObjectProperty(:Ingestion)) Declaration(ObjectProperty(:user)) ObjectPropertyDomain (:DataLake_Dataset : DL_DS) ObjectPropertyDomain (:DataSource_Dataset : DS_DS) ObjectPropertyDomain (:Ingestion :Ingestion) ObjectPropertyDomain (:User :user) ObjectPropertyDomain (:Acquire ObjectUnionOf(:DataSource_Dataset :User :Ingestion :DataLake_Dataset)) ObjectPropertyDomain(:User ObjectUnionOf(:DataSource_Dataset :Ingestion)) ObjectPropertyDomain(:DataSource_Dataset ObjectUnionOf(: User :Ingestion) ObjectPropertyDomain(:DataLake_Dataset ObjectUnionOf(: User :Ingestion) ObjectPropertyRange (: DL_DS :Acquire) ObjectPropertyRange (: DS_DS :Acquire) ObjectPropertyRange (:Ingestion :Acquire) ObjectPropertyRange (:user :Acquire) SubClassOf(:Acquire ObjectSomeValuesFrom(: DL_DS : DataLake_Dataset)) SubClassOf(:Acquire ObjectSomeValuesFrom(: DS_DS :DataSource_Dataset)) SubClassOf(:Acquire ObjectSomeValuesFrom(:Ingestion :Ingestion)) SubClassOf(:Acquire ObjectSomeValuesFrom(:user :User)) </pre>

It should be noted that UML aggregation and composition can only be transformed into OWL as binary associations. This approach loses the specific semantics related to composition or aggregation, which is untranslatable in OWL.

4.2. Implementation

From these transformation rules, we were able to code our UML model into OWL2 code. An extract of this code is presented in figure 3.

```
<?xml version="1.0" encoding="ISO-8859"?>
<rdf:RDF
  xmlns      = "http://domain.tld/path/MetadataDL#"
  xmlns:MetadataDL = "http://domain.tld/path/MetadataDL#"
  xmlns:base    = "http://domain.tld/path/MetadataDL#"
  xmlns:owl     = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf     = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs    = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd     = "http://www.w3.org/2001/XMLSchema#"
  <OWL : Ontology rdf : about=" " >
    <rdfs : comment> Ontology to structure and contextualize a DL based on
    metadata </rdfs : comment>
    <rdfs : label>Ontology for metadata management in DL</rdfs : label>
    </OWL : Ontology>
  <!-- declaration des classe -->
  <owl : Class rdf : ID="DataSet">
    <rdfs : subClassOf>
      <owl : Restriction>
        <owl : onProperty rdf : resource="#consistsof" />
        <owl : minCardinality
          rdf : datatype="xsd:nonNegativeInteger">1</owl: minCardinality>
        </owl : Restriction>
      </rdfs : subClassOf>
      <rdfs : subClassOf>
        <owl : Restriction>
          <owl : onProperty rdf : resource="#affectedby" />
          <owl : minCardinality
            rdf : datatype="xsd:nonNegativeInteger">0</owl: minCardinality>
          </owl : Restriction>
        </rdfs : subClassOf>
        <rdfs : subClassOf>
          <owl : Restriction>
            <owl : onProperty rdf : resource="#canrepresent" />
            <owl : Cardinality
              rdf : datatype="xsd:nonNegativeInteger">1</owl: minCardinality>
            </owl : Restriction>
          </rdfs : subClassOf>
          <rdfs : subClassOf>
            <owl : Restriction>
              <owl : onProperty rdf : resource="#DS_ID " />
              <owl : Cardinality
                rdf : datatype="xsd:nonNegativeInteger">1</owl: minCardinality>
              </owl : Restriction>
            </rdfs : subClassOf>
            <rdfs : subClassOf>
              <owl : Restriction>
                <owl : onProperty rdf : resource="#DS_Name" />
                <owl : Cardinality
                  rdf : datatype="xsd:nonNegativeInteger">1</owl: minCardinality>
                </owl : Restriction>
              </rdfs : subClassOf>
              <rdfs : subClassOf>
                <owl : Restriction>
                  <owl : onProperty rdf : resource="#DS_Description" />
                  <owl : Cardinality
                    rdf : datatype="xsd:nonNegativeInteger">1</owl: minCardinality>
                  </owl : Restriction>
                </rdfs : subClassOf>
              </owl : Class>
            </owl : Class ID="Datatable_Dataset">
              <rdfs : subClassOf rdf : resource="#DataSet" />
            </owl : Class>
          </OWL : Class ID="DataLake_Dataset">
            <rdfs : subClassOf rdf : resource="#DataSet" />
            <rdfs : subClassOf>
              <owl : Restriction>
                <owl : onProperty rdf : resource="#stored " />
                <owl : someValuesFrom rdf : resource="#DataSet">
                  </owl : Restriction>
                </rdfs : subClassOf>
              </rdfs : subClassOf>
            </owl : Restriction>
          </owl : Restriction>
        </owl : Restriction>
      </owl : Restriction>
    </owl : Restriction>
  </owl : Class>
```

```
<owl : onProperty rdf : resource="#Searchby" />
<owl : AllValuesFrom rdf : resource="#DataSet">
  </owl : Restriction>
</rdfs : subClassOf>
```

Figure3. An extract OWL code for our UML model

Conclusion

In this article, and as part of the use of the data lake as a heterogeneous source for data warehouses, a conceptual metadata management model was presented to address the issues associated with the transformation of the data lake into a data swamp. We first proposed a data lake architecture covering the limitations of the architectures proposed in the literature. Then, based on the inter and intra metadata classification, we proposed our conceptual model of metadata management covering all the expected functionalities expected of a metadata system. Moreover, we have added the new functionality "Schema evolution" which is essential for schema change, either in the context of the data lake or in the data warehouses. Finally, we proposed a set of rules to transform the elements of the conceptual model into OWL Ontology. In our future work, and based on the conceptual model and transformation rules carried out in this article, we will build a semantic data lake, using the RDF graph as an implementation model for metadata management. This will allow us to build a semantic layer between the data lake and the data warehouse, making it possible not only to structure and contextualize the data in the data lake, but also to improve performance in OLAP analyzes and queries.

Reference

1. W. H. Inmon, Building the Data Warehouse: Getting Started, John Wiley (2000).
2. A. Eberendu, Unstructured Data: An overview of the data of Big Data, IJCTT, **38** (2016).
3. K. Thomas, P.S Nair, Data Lake: A Centralized Repository, IRJET **7,5** (2020).
4. C. Paschalidi Data Governance: A Conceptual Framework in Order to Prevent Your Data Lake from Becoming a Data Swamp, 2015, <https://ltu.diva-portal.org/smash/record.jsf?pid=diva2%3A1019917&dsid=2135>
5. P.N. Sawadogo, É. Scholly, C. Favre, É. Ferey, S. Loudcher, J. Darmont, Metadata Systems for Data Lakes: Models and Features, in *European Conference on Advances in Databases and Information Systems*, CCIS, Vol. 1064, 8-11 September, Bled, Slovenia (2019)
6. E. Scholly, P. Sawadogo, P. Liu, J.A. Espinosa-Oviedo, C. Favre, S. Loudcher, J. Darmont, C. Nouss, Coining goldmedal: A new contribution to data lake generic metadata modelling, in *International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data*, DOLAP, Vol. 2840, 23 March, Nicosia, Cyprus (2021).
7. H. Fang, Managing data lakes in Big Data era: What's a data lake and why has it become popular in data management

- ecosystem, in the *International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, CYBER, 8-12 June, Shenyang, China (2015).
8. A. LAPLANTE, B. SHARMA, *Architecting Data Lakes*, O'Reilly Media (2014).
9. P. MENON, Demystifying data lake architecture, in Medium <https://rpradeepmenon.medium.com/demystifying-data-lake-architecture-30cf4ac8aa07>, 6 July (2017).
10. R. NADIPALLI, *Effective Business Intelligence with QuickSight*, Packt Publishing (2017).
11. F. Ravat, Y. Zhao, *Data Lakes: Trends and Perspectives*, in *International Conference on Database and Expert Systems Applications*, CCIS, Vol.1062, 26-29 August, Linz, Austria (2019).
12. B. Inmon, *Data Lake architecture: Designing the Data Lake and avoiding the garbage dump*, Technics Publications (2016).
13. M.R. LLAVE, Data lakes in business intelligence: Reporting from the trenches, *Procedia Computer Science* **138** (2018).
14. A. Oram, *Managing the data lake*, Zaloni (2015).
15. C. Diamantini, P.L. Giudice, L. Musarella, D. Potena, E. Storti, D. Ursino, A new meta- data model to uniformly handle heterogeneous data lake sources, in *European Conference on Advances in Databases and Information Systems*, CCIS, Vol.909, 2-5 September, Budapest, Hungary (2018).
16. F. Ravat, Y. Zhao, Metadata management for data lakes, in *European Conference on Advances in Databases and Information Systems*, CCIS, Vol.1064, 8-11 September, Bled, Slovenia (2019).
17. R. Eichler, C. Giebler, C. Gröger, H. Schwarz, B. Mitschang, HANDLE - A Generic Metadata Model for Data Lakes, in the *International Conference on Big Data Analytics and Knowledge Discovery*, LNCS, Vol.12393, 14-17 September, Bratislava, Slovakia (2020).
18. R. Hai, S. Geisler, C. Quix, Constance : An Intelligent Data Lake System, in the *international Conference on Management of Data*, SIGMOD, 26jun-1july, New York, United States (2016).
19. D. Solodovnikova, L. Niedrite, Change Discovery in Heterogeneous Data Sources of a Data Warehouse, in the *International Baltic Conference on Databases and Information Systems*, CCIS, Vol.1243, 16-19 June, Tallinn, Estonia (2020).
20. A. Maccioni, R. Torlone, KAYAK : A Framework for Just-in-Time Data Preparation in a Data Lake, in the *International Conference on Advanced Information Systems Engineering*, LNCS, Vol.10816, 15-16 June, Tallinn, Estonia (2018).
21. M.M. Mkhinini, O. Labbani-Narsis, C. Nicolle, Combining UML and ontology: An exploratory survey, *COSREV*, **35**, 1-14 (2020).
22. A. LeGuennec, Méthodes formelles avec uml, in CFIP, Hermes (2000).
23. P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar, J. Smith, UML for ontology development, *KER* **17**, 61-64 (2002).
24. O. El Hajjamy, K. Alaoui, L. Alaoui, M. Bahaj, Mapping UML to OWL2 ontology, *JTAIT* **90**, 126-143 (2016).
25. J. Zedlitz, N. Luttenberger, Transforming between UML conceptual models and owl 2 ontologies, in the *Terra Cognita Workshop on Foundations, Technologies and Applications of the Geospatial Web*, ISWC, Vol.901, 12 November, Boston, USA (2012).
26. A. Belghiat, M. Bourahla, Transformation of UML models towards OWL ontologies, in *International Conference on Sciences of Electronics, Technologies of Information and Telecommunications*, SETIT, 21-24 March, Sousse, Tunisia (2012).
27. Unified Modeling Language, Version 2.5, OMG, 2015, <http://www.omg.org/spec/UML/2.5>
28. J. Zedlitz, N. Luttenberger, Conceptual modelling in UML and OWL-2, in *International Journal on Advances in Software* **7**, 182-196 (2014).
29. J. Zedlitz, N. Luttenberger, Data types in UML and OWL-2, in the *International Conference on Advances in Semantic Processing*, SEMAPRO, Vol.1, 29 September-3 October, Porto, Portugal (2013).
30. M. Sadowska, Z. Huzar, Representation of UML Class Diagrams in OWL 2 on the Background of Domain Ontologies, *EISEJ* **13**, 63-103 (2019).