Data Lake and Digital Enterprise

Oumaima El Haddadi[1], Mahmoud El Hamlaoui[1], Dkaki Taoufiq[2] and Mahmoud Nassar[1]
[1]*IMS Team, ADMIR Laboratory, Rabat IT Center, Mohammed V University in Rabat, Rabat, Morocco*
[2]*IRIS Team, IRIT Laboratory, University Toulouse Jean - Jaurès, Toulouse, France*

Keywords: Data Lake, Big Data, Metadata, Data Management.

Abstract: Due to the digital transformation and huge amount of publicly available data, decision support systems are becoming highly useful in helping to defining, managing and improving business strategies and objectives. Indeed, data is a key asset and a key competitive differentiator for all organizations. This newly available data has changed traditional data processing and created new challenges related to the velocity, volume and variety of data. To address these challenges related to the storage of heterogeneous data and to provide the ability of rapid data processing, we explore the data lake paradigm. In this paper, we present the state-of-the-art of Data Lake systems and highlight their major advantages and drawbacks. We also will propose a solution to improve Data Lake System.

1 INTRODUCTION

We live in a society where information and data became the new oil. Data is growing rapidly because of the over-connected world we live in.

Data becomes more important over time, since it is a key issue in understanding and analyzing scientific, economic, political and social problems.The volume of digital data doubles every year and is expected to reach 44 billion gigabytes (GB) during 2020 according to *talend* website [1].

With the diversity of unstructured and semi-structured data, we face a dual challenge in finding efficient solutions to store this huge amounts data and in having the necessary capacities for their fast processing.

In this article, we will focus on the problems of migrating a company's traditional information system to a DL. These problems are related to the implementation, management and the exploitation. Since it is not easy to transform all data in a DL at once, because it is costly. Companies need a service-oriented architecture. That is why we propose an approach that creates an interface for each source of information to menage access to external data (along with the company internal databases) without actually permanently storing it. We will concentrate on extracting, if it is possible, the metadata (schema) of each sources to perform a "virtual" data migration (metadata migration). These schemata will be merged to form one global schema. The advantage of this approach is that we don't need to change the whole systems information to a single server in one time (which will be highly expansive).

This paper is organized in three sections. The first section describes the literature related data storage. In the second section, we propose a two layers architecture (physical and logical) for managing data lake. We show our contribution related to the problem of linking the physical layer and the logical layer using metadata for each type of data (data source). The third section presents the experimentation we conduct to assess our proposal.

2 LITERATURE REVIEW

The huge amount of available digital data -Big Data-, has changed our way of conceiving scalable data operations from collecting to analysis. Indeed, data community has been forced to find new solutions to collect, store and use data. The need for these new solutions mainly stems from the new characteristics of nowadays data -variety of sources and schemata (open data, social data, GPS...), volume, velocity (continual changing), ... Specifically for data storage, traditional IT solutions, such as data warehouses, that use relational data are no longer suitable as they do not scale

[1]https://www.talend.com/resources/what-is-data-lake

El Haddadi, O., El Hamlaoui, M., Taoufiq, D. and Nassar, M.
Data Lake and Digital Enterprise.
DOI: 10.5220/0009415604230429
In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2020)*, pages 423-429
ISBN: 978-989-758-421-3
Copyright © 2020 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

to thees new characteristics.

New architecture and approaches has been proposed. Among them is a new storage concept called Data Lake (DL), which addresses the big challenges of Big Data that correspond to the question raised by the authors of (R. Hai and Quix, 2016): 'how to make an easy use of highly diverse data and provide knowledge?'

Data Lake was first introduced by James Dixon (Chief Technology Officer (CTO) at Pentaho) in a blog post (Dixon, 2010), where he said that if we think that a data marts as a store of bottled water so the DL is the rich source of water in a more natural state (raw data). In addition, in his initial article (Dixon, 2014), he describes the DLs as a system that store data into a single source (Miloslavskaya and Tolstoy, 2016) (Quix and Hai, 2018).

Barry Delvin said that a DL is: "In the simplest summary, it is the idea that all enterprise data can and should be stored in HADOOP and accessed and used equally by all business applications."

Before moving on to DL it's necessary to define what is a digital enterprise. There is undoubtedly a change in companies' ecosystem (Information systems), due to the rapid evolution of technologies - essentially the rapid explosion of data. Therefore the mastering this ecosystem data with fast and efficient tools becomes a necessity for organisations. Hence the digitalization of internal business processes for example: customer portfolio management in the banking sector, production management in the industrial sector..., and external business processes for example: customer relations, supplier relations...

So what is digital enterprise?, according to Margaret Rouse[2] , It is an organization that uses technology as a competitive advantage on these operations. In addition, the rapid technological change is forcing business leaders and heads of institutions to regularly review their strategy to adapt it to the needs and the environment in which they operate.

2.1 Data Lake Context

DL can be defined as a massively scalable storage repository that contains a very large amount of raw data in its native (original) format from various sources for an indefinite period of time (till it is no longer neededd) (Wang, 2017). Among the benefits of this storage method, is the ease of coexistence between different schema and structural forms of data, usually blobs of objects or files (R. Hai and Quix, 2016),(Miloslavskaya and Tolstoy, 2016), (Llave, 2018), (Kachaoui and Belangour, 2019). Most researcher take the DL as just a methodology but it is an actual new data architecture that includes both hardware, software and conceptual design (Madera and Laurent, 2016).

Different users can access to the DL at any time because the data are stored in its raw form (Zagan and Danubianu, 2019), so they can analyze, structure and process it. Moreover, a DL can store all types of data such structured, semi-structured and unstructured data. Also, we can say that the advantage of a DL is its ability to trace each piece of data back to any time in the past (because the data are stored as it is directly from its original source of storage), and any organization can perform any analysis on any old stored data.

DL use Schema on Read technique as it is the case for ELT process. This technique offers more flexibility in using a huge amount and different types of data (Zagan and Danubianu, 2019).

According to a survey conducted by TDWI [3], 82% of users deal with rapidly changing data in terms of structures, types, sources and volumes. In addition, 85% of the people surveyed consider the DL as an opportunity, because of analytical applications that require all types of data (old and new data) to be widely consolidated. On another hand, the DL is exploited for several benefits and use cases (Advanced Analysis 49% and Data Discovery 49%). Also the TDWI survey states that "only a quarter of surveyed organizations have at least one DL in production, but another quarter plan to enter production within a year".

2.2 Data Lake Solutions

Most researchers in DL offer solutions (system, service, product ...) that handle some specific problems. For example, (A. Beheshti and Zhao, 2017) propose two DL services CoreDB and CoreKG. CoreDB was created in 2017 as is an open source service that provide relational and NoSQL databases-as-a-service to develop web data application, and it use the power of Elasticsearch as a seaurch engine. It helps analysts to create a DL. This solution offers tools to easily organize, index and query data and metadata and provides an integrated design for security and tracking. CoreKG was created in 2018 to complete the CoreDB services. CoreKG offers a contextualized DL service that provides researchers and developers with the possibility of managing multiple database technologies from relational to NoSQL. It offers a built-in design for data curation, security and provenance

[2]https://searchcio.techtarget.com/definition/Digital-enterprise

[3]https://tdwi.org/articles/2017/03/29/tdwi-research-report-examines-emerging-best-practices-for-data-lakes.aspx

(A. Beheshti and Tabebordbar, 2018).

In (M. Wibowo and Shamsuddin, 2017), the authors propose a machine learning technique to optimize data management processes in a DL by combining data silos. This solution intended to improve data quality is divided into two phases. The first phase bridges the gap between the data sources, i.e. the data silos, and the DL that will manage the data source. In this phase, data discovery will describe the data, governance will capture the data using evolving metadata, and data mining will new data models to combine it with other ML processes. The second phase, is for verifying the result. It uses several tools related to Reporting, BI, Visualization...

In the same context, (A. Farrugia and Thompson, 2016) proposes a DL management (DLM) by extracting the metadata from the database using Social Network Analysis. (Z. Shang and Feng, 2016) proposes the iFuse (data fusion platform) that bases on Bayesian Graphical Model to manage and query a DL. (I. D. Nogueira and Ea, 2018) uses a group of modeling to handle schema evolution in a DL and proposes a data vault. (Sawadogo and Darmont, 2019) presents a methodological approach to manage and build a metadata system for textual document in a DL. Also, (L. Chen and Zhuang, 2015) propose a data model for unstructured data and the RAISE method to process it using a SQL-like query language. Hai and colleagues present an intelligent system under the name Constance. This system (R. Hai and Quix, 2016) have been proposed as a solution to non-integrated data management system with heterogeneous schema, and to avoid the problem of "data swamp". Constance was built to discover, extract and summarize the structural metadata of data sources and to annotate data and metadata with semantic information to avoid ambiguities. Another system proposed in (M. Farid and Chu, 2016) introduces the CLAMS system that discovers the raw data and metadata integrity constraint using the RDF model. To validate the result, this system requires human intervention.

From these previous work, we state that most researchers use specific data type as while addressing the data heterogeneity problem. In another words, contributors start defining the two main axes of a DL, which are the data extraction and data management phase. they start choosing, beforehand, the targeted data type. Since we acknowledged the variety aspect of nowdays data -there are various of structured, semi-structured and unstructured data-, the proposed DL architectures and models are limited to the only type of data they explicitly target. also, what is missing in the existing work is the projection of the approaches in a real case (some of them have a large project which, based on a real case as (R. Hai and Quix, 2016)).

To summarize, most research focuses on the management system and exploration of DL using popular knowledge and tools such as machine learning, data quality, social networks focus on textual data. This only concerns a part of the variety of data types.

3 DATA LAKE MANAGEMENT

As stated previously, a Data Lake is a sustainable solution for companies which want to take advantage of publicly available data. However, DL solutions hard to implement, manage and operate especially if targeted data sources are heterogeneous. It is therefore necessary to have an architecture that can adapt to any type of data structure or format and ensure the storage, ingestion and preparation policy.

According to the literature, the company needs a service-oriented architecture but it isn't easy to transform the entire information system into a single DL. To deal with this problem we propose the creation of interfaces for each source of data. In addition, we believe that we need to create a virtual DL with two layers (physical layer and logical layer) in order to conserve resources adequately.

The question that arises now is how to link the two layers? To do so, we propose a DL architecture, covering the business perimeter, then we focus on managing a DL by grouping metadata, managing the schema, managing database access and indicating how to extract metadata from any possible source.

3.1 Architecture

The figure 1 bellow shows the architecture of our DL. It is divided into two layers:

- **Physical Layer:** This layer, makes the physical and real link between the DL and the external (API, web page, etc.) and internal (databases, flat file, etc.) sources of an organization. In other words, it is responsible for establishing the connection of each source through a dedicated interface that take into account the nature of each source.
- **Logical Layer:** this layer illustrates the core of our architecture. It Contains several functionalities. For example, as soon as the connection is established with a source, we retrieve the metadata of this source and store it in our database. After that, the integration of all metadatas is performed by storing them in the same database. From this

moment, it is possible to update the data dictionary that represents the mapping of the sources in the DL.

Moreover, the purpose of a DL is to prepare the data for the user. In order to achieve this purpose, we use the following technique: Data preparation, data pre-processing and data wrangling. In addition, we have to implement the necessary treatments such as machine learning which helps us to prepare the data and then create interfaces to disseminate the data requested by the user.

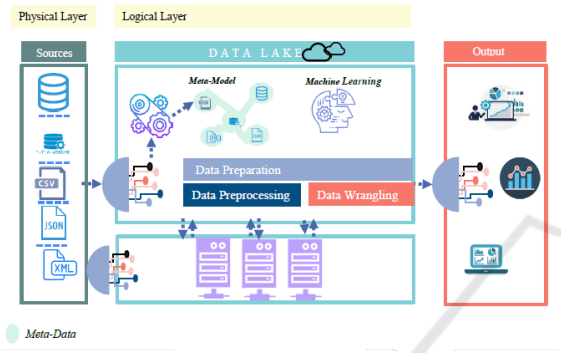

Figure 1: Proposed architecture of the Virtual Data Lake.

3.2 Conception Model

We have built a conceptual model of our DL as a set of stores. Each store created by a user contains two types of sources: input sources and output sources. The source is an interface between the physical source of data and the logical layer of DL.

The first constraint to solve is how to integrate all possible sources such as relational databases, flat files or other data sources in the DL. To overcome these constraints, we use different design patterns (E. Gamma and Vlissides, 1994). The first design pattern we use is the Bridge pattern.

So, our Source class will become an abstract class containing the methods: connect (), disconnect (), state (), getShemaFromSource (), TransformToMataData () and other methods that we will added subsequently.

Therefore, we take the first case of a DBMS Source (MySQL, PostgreSQL, etc.) . We use SgbdSource class to connect to a relational database. This requires three important elements to communicate with the database:

1. The database link with the port,
2. The login,
3. Password.

For any other type of source, we add the necessary elements to establish the links with our DL as shown the figure 2.

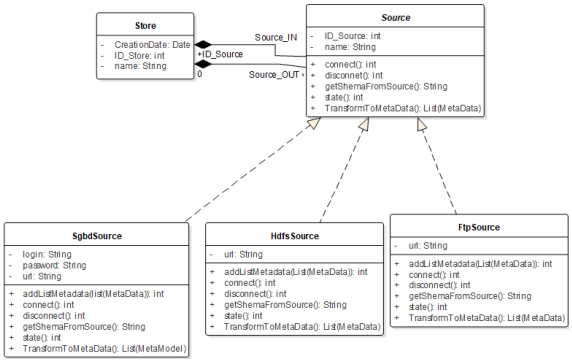

Figure 2: Conception the Source class with the patron bridge.

3.3 Meta-Model

The goal of a DL is to enable the user to see and understand all the data in his information system. This information system consists of a large variety of data. So, we must first analyze and manage the metadata of each source and use it through the metadata set. Metadata management is a vital step in understanding the business of a company. With the previous design, we identified the source and the next step is to extract the metadata from them. When talking about metadata, we are implicitly talking about the types of data we can solicit, so we are talking about a classic Big Data problem: the variety of data. To unify and make all metadata uniform, we have created a general meta-model that groups all metadata together (figure 3). Therefore, each entity of metadata is a set of variables. We note:

$$M = \{v_1, ..., v_n\}, \; such \; v_i \in V \; where \; V \; a \; set \; of \; variables \tag{1}$$

3.4 Transformation of Relational Metadata

For a relational schema, we transform all relationships into a metadata entity. Afterwards, we take all the names of the properties out of foreign keys and we transform them on a variable typed as String. The third step aims to transform the foreign keys into variables. For each foreign key, we should know its source relationships because we use the metadata name of this source relationships to assign the type of variable. The algorithm 1 summarize all this step.

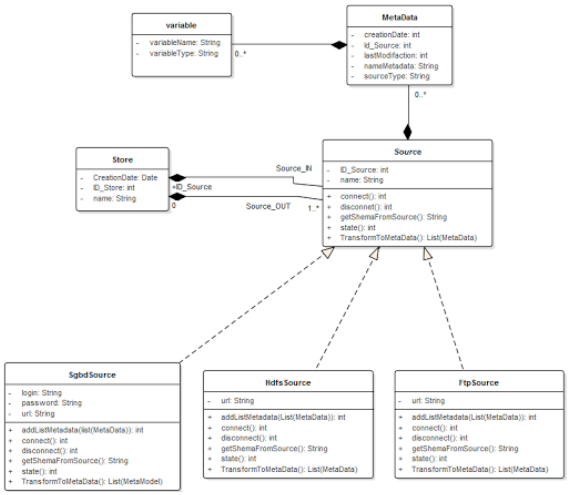

Figure 3: Global Conception.

Algorithm 1: Transformation of relational metadata.

1 $R_x \leftarrow ReLationExtract(q)$; `q: set of relation with attribute`
2 **for** R *in* R_x **do**
3 $m_{new} \leftarrow newMetadata(R)$;
4 $P_x \leftarrow labelExtractPopriety(R)$;
5 **for** P *in* P_x **do**
6 $v_{new} \leftarrow newvaraibles(P,"String")$;
7 $m_{new}.getListVariable.Add(v_{new})$;

As an example, we take the following relational schema: Film(NameFilm(Int), ReleaseYear(date), #FilmMaker(int))
FilmMaker(IdFilmMaker(int), LastName(string), FirstName(String))

After the transformation our example gives the following schema:
MFilm(NameFilm(string), ReleaseYear(string), FilmMaker(MFilmMaker));
MFilmMaker(IdFilmMaker(string), LastName(String), FirstName(String))

3.5 Flat File Transformation

If we took the example of CSV files. For each one, we create a meta-data entity and all columns becomes variables in the new entity as shown the algorithm 2.

For example, a following transaction CSV file: (Transaction_date, Product, Price, Payment_Type, Name, City, State, Country, Account_Created, Last_Login, Latitude, Longitude)

Turns into:

MTransaction(Transaction_date(String), Product(String), Price(String),Payment_Type(String), Name(String), City(String), State(String), Country(String), Account_Created(String), Last_Login(String), Latitude(String), Longitude(String))

Algorithm 2: Flat file transformation.

1 $m_{new} \leftarrow newMetadata("ID")$; `ID: Auto Increment`
2 $P_x \leftarrow labelExtractPoprietyFromFile()$;
3 **for** P *in* P_x **do**
4 $v_{new} \leftarrow newvaraibles(P,"String")$;
5 $m_{new}.getListVariable.Add(v_{new})$;

The management of the store is carried out by the ServiceManageStore class. This class contains different methods like creating stores and creating sources, as well as getting the source list and metadata list.

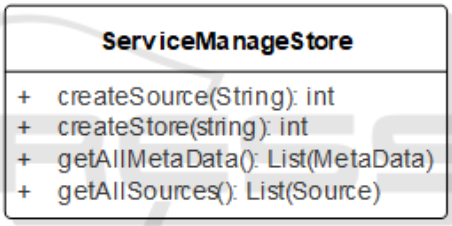

Figure 4: ServiceManageStore.

The createSource() function takes as parameter the user request in a json format file. This latter contains, as shown below, the type of source and the information needed to establish the connection: "Type": "mysql", "Parameter": ["url": "mysql://localhost:3306/todo", "login": "admin", "password": "admin"]

The previous method will establish a connection and then it will call the getShemaFromSource() method. This latter is specific for each type of source. The output result of the previous function is passed as a parameter to the TransformToMetaData() function. It is dedicated for each type of source, like the transformation algorithms seen in section (3.4 and 3.5), and then we will add the output result as a list of meta-data in the source object. Finally, we invoke the SaveSource () method in the SourceRepository to store the source class in the database. The sequence diagram (figure 5) shows the process of creating the source. It represents the most critical function of our model.

To conclude, we propose two examples of transformations. So, for each source type, we create a

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

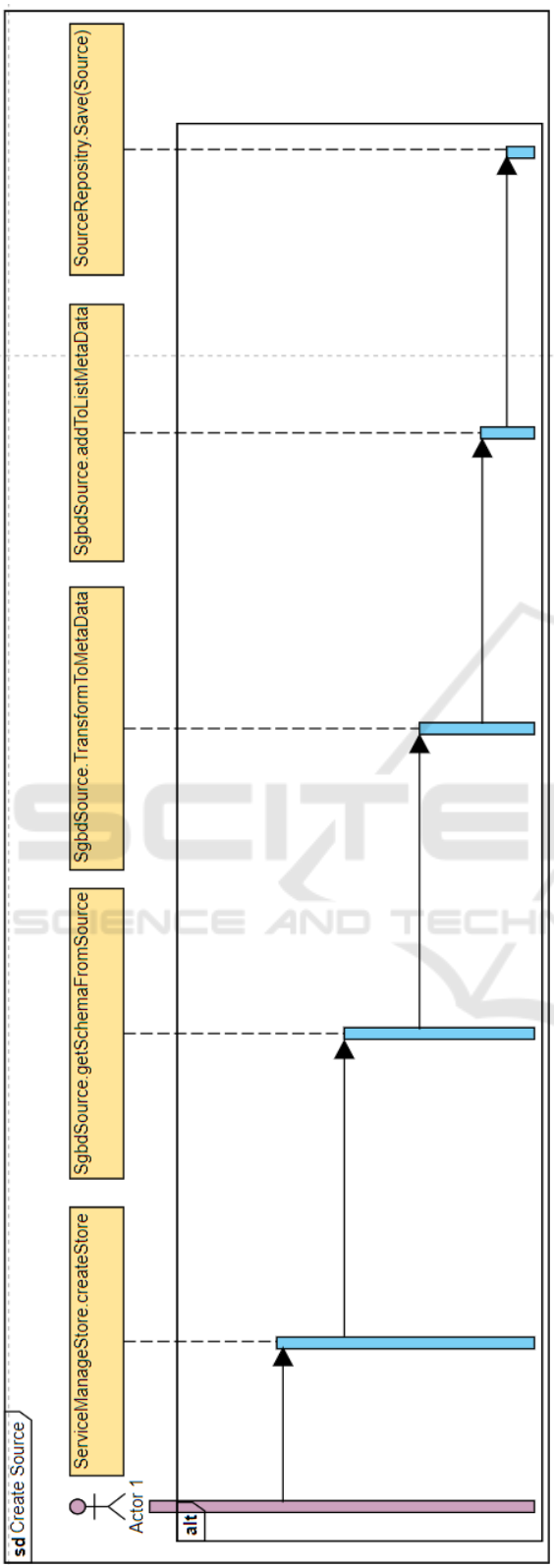

Figure 5: Sequence Diagram.

dedicated transformation model, using for each type a specific algorithm.

4 EXPERIMENTATION

To experiment our approach we used a docker which is a software that will allow us to launch different applications in different containers. The objective is to simulate the real environment of a company with its different databases. For this we will take three containers that represent our sources:

- Container number 1: Relational database server;
- Container number 2: Hdfs server contains csv files;
- Container number 3: Cassandra server;
- Fourth container contains our DL layer.

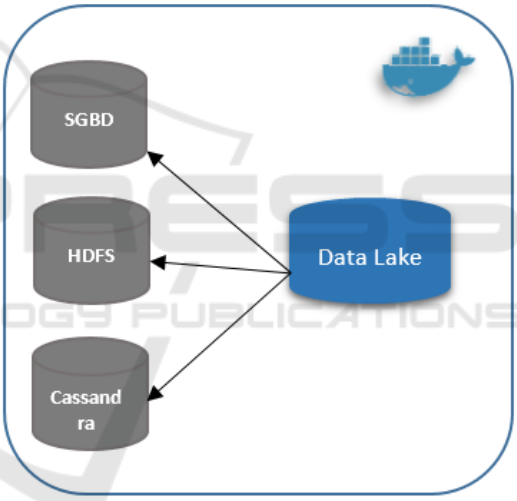

Figure 6: Experimentation Schema.

We used for the implementation of our application, the Framework Spring, with its variant Spring Boot that allows us to launch a Rest API to deploy our application.

For the creation of a store we send the following request using curl:
curl -d ’”name”: ”store1”’ -H ’Content-Type: application/json’ http://localhost:8082/spring-rest/store/add

Also, for the creation of sgdb type sources we send the following request:
curl -d ’”Type”: ”mysql”, ”Parameter”:[”url”: ”mysql://localhost:3306/testSource”, ”login”: ”admin”, ”password”: ”admin”]’ -H ’Content-Type: application/json’ http://localhost:8082/spring-rest/Srource/add

In this experiment, we check the good functioning of our application, which aims to collect and manage the metadata of the chosen sources. We also provide other interfaces to help the user manipulating, visualizing and managing the store and the sources being created.

5 CONCLUSION AND FUTURE WORK

In this article, we presented a DL literature review, a DL architecture proposal called Virtual DL, and a conceptual model that defines a DL as a set of stores. We also define an experimentation framework to access our model. To integrate data sources, we used bridge patterns and create metadata that describe each targeted data source. The goal of this proposal is to avoid the migration of the whole digital and IT infrastructure of the company to a single server as a single operation. This a single-operation relocation of resources would be too expensive. Our solution offers companies the opportunity to gradually transform their information system without supporting an exorbitant cost while still benefiting from the entire available data. The entire cost will be gradually absorbed as the relocation of resources goes on. As a future work, we believe that it is necessary to manage a global schema that comprehends organization's data, and create a dictionary of relationships between internal and external data that handles the problem of heterogeneous data.

REFERENCES

A. Beheshti, B. Benatallah, R. N. and Tabebordbar, A. (2018). Corekg: a knowledge lake service.

A. Beheshti, B. Benatallah, R. N. V. M. C. H. X. and Zhao, X. (2017). Coredb: a data lake service.

A. Farrugia, R. C. and Thompson, S. (2016). Towards social network analytics for understanding and managing enterprise data lakes.

Dixon, J. (2010). Pentaho, hadoop, and data lakes.

Dixon, J. (2014). Data lakes revisited.

E. Gamma, R. Helm, R. J. and Vlissides, J. (1994). Design patterns : Elements of reusable object-oriented software.

I. D. Nogueira, M. Romdhane, J. D. U. d. L. and Ea, E. (2018). Modélisation des métadonnées d'un data lake en data vault.

Kachaoui, J. and Belangour, A. (2019). Challenges and benefits of deploying big data storage solution.

L. Chen, J. Shao, Z. Y. J. S. F. W. and Zhuang, Y. (2015). Raise: A whole process modeling method for unstructured data management.

Llave, M. R. (2018). Data lakes in business intelligence: reporting from the trenches.

M. Farid, A. Roatis, I. F. I. H.-F. H. and Chu, X. (2016). Clams: Bringing quality to data lakes.

M. Wibowo, S. S. and Shamsuddin, S. M. (2017). Machine learning in data lake for combining data silos.

Madera, C. and Laurent, A. (2016). The next information architecture evolution: the data lake wave.

Miloslavskaya, N. and Tolstoy, A. (2016). Big data, fast data and data lake concepts.

Quix, C. and Hai, R. (2018). Data lake.

R. Hai, S. G. and Quix, C. (2016). Constance: An intelligent data lake system.

Sawadogo, T. K. and Darmont, J. (2019). Metadata management for textual documents in data lakes.

Wang, L. (2017). Heterogeneous data and big data analytics.

Z. Shang, Y. Liu, G. L. and Feng, J. (2016). K-join: Knowledge-aware similarity join.

Zagan, E. and Danubianu, M. (2019). From data warehouse to a new trend in data architectures – data lake.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS