Journal of Intelligent Information Systems (2021) 56:97–120
https://doi.org/10.1007/s10844-020-00608-7

On data lake architectures and metadata management

Pegdwendé Sawadogo[1] · **Jérôme Darmont**[1]

Received: 8 December 2019 / Revised: 20 March 2020 / Accepted: 22 May 2020 /
Published online: 26 January 2020

Abstract
Over the past two decades, we have witnessed an exponential increase of data production in the world. So-called big data generally come from transactional systems, and even more so from the Internet of Things and social media. They are mainly characterized by volume, velocity, variety and veracity issues. Big data-related issues strongly challenge traditional data management and analysis systems. The concept of data lake was introduced to address them. A data lake is a large, raw data repository that stores and manages all company data bearing any format. However, the data lake concept remains ambiguous or fuzzy for many researchers and practitioners, who often confuse it with the Hadoop technology. Thus, we provide in this paper a comprehensive state of the art of the different approaches to data lake design. We particularly focus on data lake architectures and metadata management, which are key issues in successful data lakes. We also discuss the pros and cons of data lakes and their design alternatives.

Keywords Data lakes · Data lake architectures · Metadata management · Metadata modeling

1 Introduction

The 21st century is marked by an exponential growth of the amount of data produced in the world. This is notably induced by the fast development of the Internet of Things (IoT) and social media. Yet, while big data represent a tremendous opportunity for various organizations, they come in such volume, speed, heterogeneous sources and structures that they exceed the capabilities of traditional management systems for their collection, storage and processing in a reasonable time (Miloslavskaya and Tolstoy 2016). A time-tested solution for big data management and processing is data warehousing. A data warehouse is indeed an integrated and historical storage system that is specifically designed to analyze data.

✉ Pegdwendé Sawadogo
pegdwende.sawadogo@univ-lyon2.fr

Jérôme Darmont
jerome.darmont@univ-lyon2.fr

[1] Université de Lyon, Lyon 2, ERIC UR 3083, Lyon, France

However, while data warehouses are still relevant and very powerful for structured data, semi-structured and unstructured data induce great challenges for data warehouses. Yet, the majority of big data is unstructured (Miloslavskaya and Tolstoy 2016). Thus, the concept of data lake was introduced to address big data issues, especially those induced by data variety.

A data lake is a very large data storage, management and analysis system that handles any data format. It is currently quite popular and trendy both in the industry and academia. Yet, the concept of data lake is not straightforward for everybody. A survey conducted in 2016 indeed revealed that 35% of the respondents considered data lakes as a simple marketing label for a preexisting technology, i.e., Apache Hadoop (Grosser et al. 2016).

Knowledge about the concept of the data lake has since evolved, but some misconceptions still exist, presumably because most of data lakes design approaches are abstract sketches from the industry that provide few theoretical or implementation details (Quix and Hai 2018). Therefore, a survey can be useful to give researchers and practitioners a better comprehension of the data lake concept and its design alternatives.

To the best of our knowledge, the only literature reviews about data lakes are all quite brief and/or focused on a specific topic, e.g., data lake concepts and definitions (Couto et al. 2019; Madera and Laurent 2016), the technologies used for implementing data lakes (Mathis 2017) or data lakes inherent issues (Giebler et al. 2019; Quix and Hai 2018). Admittedly, the report proposed in Russom (2017) is quite extensive, but it adopts a purely industrial view. Thus, we adopt in this paper a wider scope to propose a more comprehensive state of the art of the different approaches to design and exploit a data lake. We particularly focus on data lake architectures and metadata management, which lie at the base of any data lake project and are the most commonly cited issues in the literature (Fig. 1).

More precisely, we first review data lake definitions and complement the best existing one. Then, we investigate the architectures and technologies used for the implementation of data lakes, and propose a new typology of data lake architectures. Our second main focus is metadata management, which is a primordial issue to avoid turning a data lake into an inoperable, so-called data swamp. We notably classify data lake metadata and introduce the features that are necessary to achieve a full metadata system. We also discuss the pros and cons of data lakes.

Eventually, note that we do not review other important topics, such as data ingestion, data governance and security in data lakes, because they are currently little addressed in the literature, but could still presumably be the subject of another full survey.

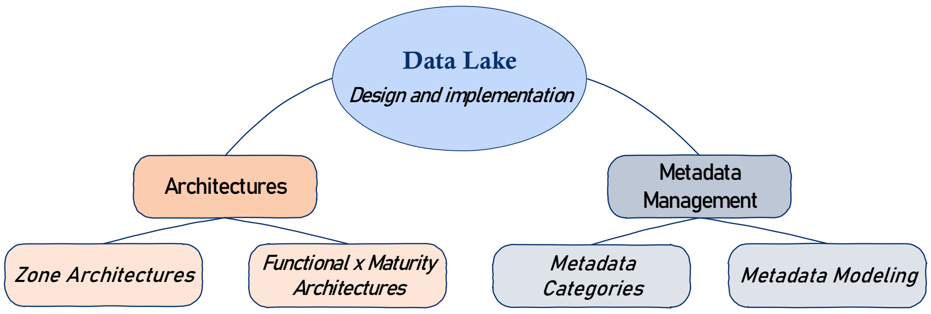

Fig. 1 Issues addressed in this paper

The remainder of this paper is organized as follows. In Section 2, we define the data lake concept. In Section 3, we review data lake architectures and technologies to help users choose the right approach and tools. In Section 4, we extensively review and discuss metadata management. Eventually, we recapitulate the pros and cons of data lakes in Section 5 and conclude the paper in Section 6 with a mind map of the key concepts we introduce, as well as current open research issues.

2 Data lake definitions

2.1 Definitions from the literature

The concept of data lake was introduced by Dixon as a solution to perceived shortcomings of datamarts, which are business-specific subdivisions of data warehouses that allow only subsets of questions to be answered (Dixon 2010). In the literature, data lakes are also refered to as data reservoirs (Chessell et al. 2014) and data hubs (Ganore 2015; Laskowski 2016), although the terms data lake are the most frequent. Dixon envisions a data lake as a large storage system for raw, heterogeneous data, fed by multiple data sources, and that allows users to explore, extract and analyze the data.

Subsequently, part of the literature considered data lakes as an equivalent to the Hadoop technology (Fang 2015; Ganore 2015; O'Leary 2014). According to this point of view, the concept of data lake refers to a methodology for using free or low-cost technologies, typically Hadoop, for storing, processing and exploring raw data within a company (Fang 2015).

The systematic association of data lakes to low cost technologies is becoming minority in the literature, as the data lake concept is now also associated with proprietary cloud solutions such as Azure or IBM (Madera and Laurent 2016; Sirosh 2016) and various data management systems such as NoSQL solutions and multistores. However, it can still be viewed as a data-driven design pattern for data management (Russom 2017).

More consensually, a data lake may be viewed as a central repository where data of all formats are stored without a strict schema, for future analyses (Couto et al. 2019; Khine and Wang 2017; Mathis 2017). This definition is based on two key characteristics of data lakes: data variety and the schema-on-read approach, also known as late binding (Fang 2015), which implies that schema and data requirements are not fixed until data querying (Khine and Wang 2017; Maccioni and Torlone 2018; Stein and Morrison 2014). This is the opposite to the schema-on-write approach used in data warehouses.

However, the variety/schema-on-read definition may be considered fuzzy because it gives little detail about the characteristics of a data lake. Thus, Madera and Laurent introduce a more complete definition where a data lake is a logical view of all data sources and datasets in their raw format, accessible by data scientists or statisticians for knowledge extraction (Madera and Laurent 2016).

More interestingly, this definition is complemented by a set of features that a data lake should include:

1. data quality is provided by a set of metadata;
2. the lake is controlled by data governance policy tools;
3. usage of the lake is limited to statisticians and data scientists;
4. the lake integrates data of all types and formats;
5. the data lake has a logical and physical organization.

2.2 Discussion and new definition

Madera and Laurent's definition of data lakes is presumably the most precise, as it defines the requirements that a data lake must meet (Section 2.1). However, some points in this definition are debatable.

The authors indeed restrain the use of the lake to data specialists and, as a consequence, exclude business experts for security reasons. Yet, in our opinion, it is entirely possible to allow controlled access to this type of users through a navigation or analysis software layer.

Moreover, we do not share the vision of the data lake as a logical view over data sources, since some data sources may be external to an organization, and therefore to the data lake. Since Dixon explicitly states that lake data come from data sources (Dixon 2010), including data sources into the lake may therefore be considered contrary to the spirit of data lakes.

Finally, although quite complete, Madera and Laurent's definition omits an essential property of data lakes: scalability (Khine and Wang 2017; Miloslavskaya and Tolstoy 2016). Since a data lake is intended for big data storage and processing, it is indeed essential to address this issue. Thence, we amend Madera and Laurent's definition to bring it in line with our vision and introduce scalability (Sawadogo et al. 2019).

Definition 1 A data lake is a scalable storage and analysis system for data of any type, retained in their native format and used *mainly* by data specialists (statisticians, data scientists or analysts) for knowledge extraction. Its characteristics include:

1. a metadata catalog that enforces data quality;
2. data governance policies and tools;
3. accessibility to various kinds of users;
4. integration of any type of data;
5. a logical and physical organization;
6. scalability in terms of storage and processing.

3 Data lake architectures and technologies

Existing reviews on data lake architectures commonly distinguish pond and zone architectures (Giebler et al. 2019; Ravat and Zhao 2019a). However, this categorization may sometimes be fuzzy. Thus, we introduce in Section 3.1 a new manner to classify data lakes architectures that we call Functional $\times$ Maturity. In addition, we present in Section 3.2 a list of possible technologies to implement a data lake. Eventually, we investigate in Section 3.3 how a data lake system can be associated with a data warehouse in an enterprise data architecture.

3.1 Data lake architectures

3.1.1 Zone architectures

Pond architecture Inmon designs a data lake as a set of data ponds (Inmon 2016). A data pond can be viewed as a subdivision of a data lake dealing with data of a specific type. According to Dixon's specifications, each data pond is associated with a specialized storage system, some specific data processing and conditioning (i.e., data transformation/preparation) and a relevant analysis service. More precisely, Inmon identifies five data ponds (Fig. 2).

1. The **raw data pond** deals with newly ingested, raw data. It is actually a transit zone, since data are then conditioned and transferred into another data pond, i.e., either the analog, application or textual data pond. The raw data pond, unlike the other ponds, is not associated with any metadata system.
2. Data stored in the **analog data pond** are characterized by a very high frequency of measurements, i.e., they come in with high velocity. Typically, semi-structured data from the IoT are processed in the analog data pond.
3. Data ingested in the **application data pond** come from software applications, and are thus generally structured data from relational Database Management Systems (DBMSs). Such data are integrated, transformed and prepared for analysis; and Inmon actually considers that the application data pond is a data warehouse.
4. The **textual data pond** manages unstructured, textual data. It features a textual disambiguation process to ease textual data analysis.
5. The purpose of the **archival data pond** is to save the data that are not actively used, but might still be needed in the future. Archived data may originate from the analog, application and textual data ponds.

Zone architectures So-called zone architectures assign data to a zone according to their degree of refinement (Giebler et al. 2019). For instance, Zaloni's data lake (LaPlante and Sharma 2016) adopts a six-zone architecture (Fig. 3).

1. The **transient loading zone** deals with data under ingestion. Here, basic data quality checks are performed.
2. The **raw data zone** handles data in near raw format coming from the transient zone.
3. The **trusted zone** is where data are transferred once standardized and cleansed.
4. From the trusted area, data move into the **discovery sandbox** where they can be accessed by data scientists through data wrangling or data discovery operations.
5. On top of the discovery sandbox, the **consumption zone** allows business users to run "what if" scenarios through dashboard tools.
6. The **governance zone** finally allows to manage, monitor and govern metadata, data quality, a data catalog and security.

However, this is but one of several variants of zone architectures. Such architectures indeed generally differ in the number and characteristics of zones (Giebler et al. 2019), e.g.,

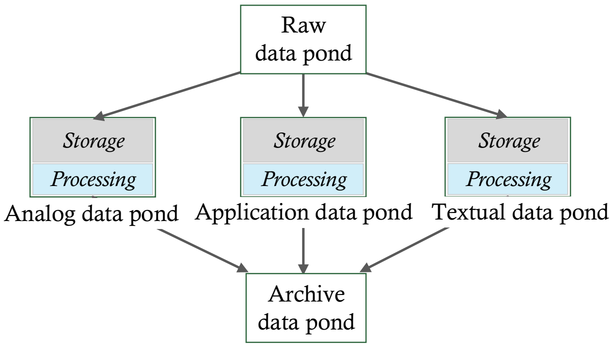

Fig. 2 Data flow in a pond architecture

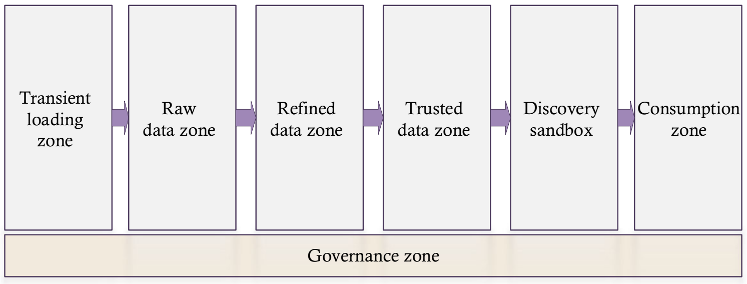

Fig. 3 Zaloni's zone architecture (LaPlante and Sharma 2016)

some architectures include a transient zone (LaPlante and Sharma 2016; Tharrington 2017; Zikopoulos et al. 2015) while others do not (Hai et al. 2016; Ravat and Zhao 2019a).

A particular zone architecture often mentioned in the data lake literature is the lambda architecture (John and Misra 2017; Mathis 2017). It indeed stands out since it includes two data processing zones: a batch processing zone for bulk data and a real-time processing zone for fast data from the IoT (John and Misra 2017). These two zones help handling fast data as well as bulk data in an adapted and specialized way.

Discussion In both pond and zone architectures, data are pre-processed. Thus, analyses are quick and easy. However, this come at the cost of data loss in the pond architectures, since raw data are deleted when transferred to other ponds. The drawbacks of the many zone architectures depend on each variant. For example, in Zaloni's architecture (LaPlante and Sharma 2016), data flow across six areas, which may lead to multiple copies of the data and, therefore, difficulties in controlling data lineage. In the Lamda architecture (John and Misra 2017), speed and batch processing components follow different paradigms. Thus, data scientists must handle two distinct logics for cross analyses (Mathis 2017), which makes data analysis harder, overall.

Moreover, the distinction of data lake architectures into pond and zone approaches is not so crisp in our opinion. The pond architecture may indeed be considered as a variant of zone architecture, since data location depends on the refinement level of data, as in zone architectures. In addition, some zone architectures include a global storage zone where raw and cleansed data are stored altogether (John and Misra 2017; Quix and Hai 2018), which contradicts the definition of zone architectures, i.e., components depend on the degree of data refinement.

3.1.2 Functional × maturity architectures

To overcome the contradictions of the pond/zone categorization, we propose an alternative way to group data lake architectures regarding the type of criteria used to define components. As a result, we distinguish functional architectures, data maturity-based architectures and hybrid architectures (Fig. 4).

Functional architectures follow some basic functions to define a lake's components. Data lake basic functions typically include (LaPlante and Sharma 2016):

1. a data ingestion function to connect with data sources;
2. a data storage function to persist raw as well as refined data;

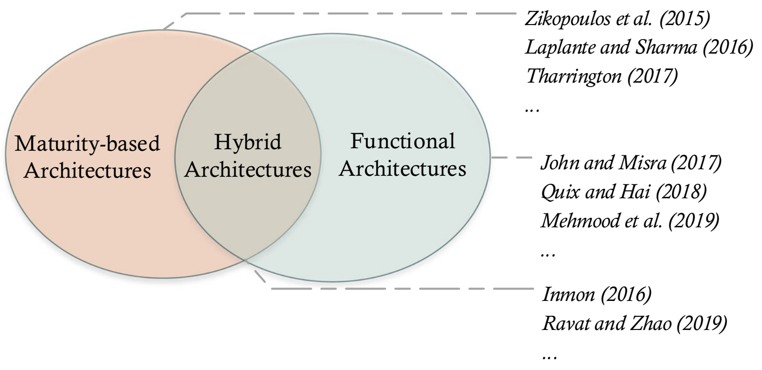

Fig. 4 Architecture typology proposal

3. a data processing function;
4. a data access function to allow raw and refined data querying.

Quix and Hai, as well as Mehmood et al., base their data lake architectures on these functions (Mehmood et al. 2019; Quix and Hai 2018). Similarly, John and Misra's lambda architecture (John and Misra 2017) may be considered as a functional architecture, since its components represent data lake functions such as storage, processing and serving.

Data maturity-based architectures are data lake architectures where components are defined regarding data refinement level. In other words, it is constituted of most zone architectures. A good representative is Zaloni's data lake architecture (LaPlante and Sharma 2016), where common basic zones are a transient zone, a raw data zone, a trusted data zone and a refined data zone (LaPlante and Sharma 2016; Tharrington 2017; Zikopoulos et al. 2015).

Hybrid architectures are data lake architectures where the identified components depend on both data lake functions and data refinement. Inmon's pond architecture is actually a hybrid architecture (Inmon 2016). On one hand, it is a data maturity-based architecture, since raw data are managed in a special component, i.e., the raw data pond, while refined data are managed in other ponds, i.e., the textual, analog and application data ponds. But on the other hand, the pond architecture is also functional because Inmon's specifications consider some storage and process components distributed across data ponds (Fig. 2). Ravat and Zhao also propose such an hybrid data lake architecture (Fig. 5 (Ravat and Zhao 2019a)).

Discussion Functional architectures have the advantage of clearly highlighting the functions to implement for a given data lake, which helps match easily with the required technologies. By contrast, data maturity-based architectures are useful to plan and organize the data lifecycle. Both approaches are thus limited, since they only focus on a unique point of view, while it is important in our opinion to take both functionality and data maturity into account when designing a data lake.

In consequence, we advocate for hybrid approaches. However, existing hybrid architecture can still be improved. For instance, in Inmon's data pond approach, raw data are deleted once they are refined. This process may induce some data loss, which is contrary to the spirit of data lakes. In Ravat and Zhao's proposal, data access seems only possible for refined data. Such limitations hint that a more complete hybrid data lake architecture is still needed nowadays.

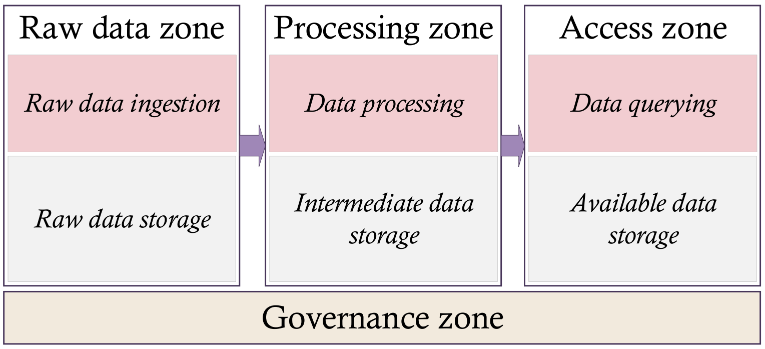

Fig. 5 Ravat and Zhao's hybrid architecture (Ravat and Zhao 2019a)

3.2 Technologies for data lakes

Most data lake implementations are based on the Apache Hadoop ecosystem (Couto et al. 2019; Khine and Wang 2017). Hadoop has indeed the advantage of providing both storage with the Hadoop Distributed File System (HDFS) and data processing tools via MapReduce or Spark. However, Hadoop is not the only suitable technology to implement a data lake. In this section, we go beyond Hadoop to review usable tools to implement data lake basic functions.

3.2.1 Data ingestion

Ingestion technologies help physically transfer data from data sources into a data lake. A first category of tools includes software that iteratively collects data through pre-designed and industrialized jobs. Most such tools are proposed by the Apache Foundation, and can also serve to aggregate, convert and clean data before ingestion. They include Flink and Samza (distributed stream processing frameworks), Flume (a Hadoop log transfer service), Kafka (a framework providing real time data pipelines and stream processing applications) and Sqoop (a framework for data integration from SQL and NoSQL DBMSs into Hadoop) (John and Misra 2017; Mathis 2017; Suriarachchi and Plale 2016).

A second category of data ingestion technologies is made of common data transfer tools and protocols (wget, rsync, FTP, HTTP, etc.), which are used by the data lake manager within data ingestion scripts. They have the key advantage to be readily available and widely understood (Terrizzano et al. 2015). In a similar way, some Application Programming Interfaces (APIs) are available for data retrieval and transfer from the Web into a data lake. For instance, CKAN and Socrata provide APIs to access a catalogue of open data and associated metadata (Terrizzano et al. 2015).

3.2.2 Data storage

We distinguish two main approaches to store data in data lakes. The first way consists in using classic databases for storage. Some data lakes indeed use relational DBMSs such as MySQL, PostgreSQL or Oracle to store structured data (Beheshti et al. 2017; Khine and Wang 2017). However, relational DBMSs are ill-adapted to semi-structured, and even more so to unstructured data. Thus, NoSQL (Not only SQL) DBMSs are usually used instead (Beheshti et al. 2017; Giebler et al. 2019; Khine and Wang 2017). Moreover, assuming that data variety is the norm in data lakes, a multi-paradigm storage system is particularly relevant (Nogueira et al. 2018). Such so-called multistore systems manage multiple DBMSs, each matching a specific storage need.

The second main way to store data and the most used is HDFS storage (in about 75% of data lakes (Russom 2017)). HDFS is a distributed storage system that offers a very high scalability and handles all types of data (John and Misra 2017). Thus, it is well suited for schema-free and bulk storage that are needed for unstructured data. Another advantage of this technology is the distribution of data that allows high fault-tolerance. However, HDFS alone is not sufficient to handle all data formats, especially structured data. Thus, it should ideally be combined with relational and/or NoSQL DBMSs.

3.2.3 Data processing

In data lakes, data processing is very often performed with MapReduce (Couto et al. 2019; John and Misra 2017; Khine and Wang 2017; Mathis 2017; Stein and Morrison 2014; Suriarachchi and Plale 2016), a parallel data processing paradigm provided by Apache Hadoop. MapReduce is well-suited to very large data, but is less efficient for fast data because it works on disk (Tiao 2018). Thus, alternative processing frameworks are used, from which the most famous is Apache Spark. Spark works like MapReduce, but adopts a full in-memory approach instead of using the file system for storing intermediate results. Thence, Spark is particularly suitable for real-time processing. Similarly, Apache Flink and Apache Storm are also suitable for real-time data processing (John and Misra 2017; Khine and Wang 2017; Mathis 2017; Suriarachchi and Plale 2016; Tiao 2018). Nevertheless, these two approaches can be simultaneously implemented in a data lake, with MapReduce being dedicated to voluminous data and stream-processing engines to velocious data (John and Misra 2017; Suriarachchi and Plale 2016).

3.2.4 Data access

In data lakes, data may be accessed through classical query languages such as SQL for relational DBMSs, JSONiq for MongoDB, XQuery for XML DBMSs or SPARQL for RDF resources (Farid et al. 2016; Fauduet and Peyrard 2010; Hai et al. 2016; Laskowski 2016; Pathirana 2015). However, this does not allow simultaneously querying across heterogeneous databases, while data lakes do store heterogeneous data, and thus typically require heterogeneous storage systems.

One solution to this issue is to adopt the query techniques from multistores (Section 3.2.2) (Nogueira et al. 2018). For example, Spark SQL and SQL++ may be used to query both relational DBMSs and semi-structured data in JSON format. In addition, the Scalable Query Rewriting Engine (SQRE) handles graph databases (Hai et al. 2018). Finally, CloudMdsQL also helps simultaneously query multiple relational and NoSQL DBMSs (Leclercq and Savonnet 2018). Quite similarly, Apache Phoenix can be used to automatically convert SQL queries into a NoSQL query language, for example. Apache Drill allows joining data from multiple storage systems (Beheshti et al. 2017). Data stored in HDFS can also be accessed using Apache Pig (John and Misra 2017).

Eventually, business users, who require interactive and user-friendly tools for data reporting and visualization tasks, widely use dashboard services such as Microsoft Power BI and Tableau over data lakes (Couto et al. 2019; Russom 2017).

3.3 Combining data lakes and data warehouses

There are in the literature two main approaches to combine a data lake and a data warehouse in a global data management system. The first approach pertains to using a data lake as the data source of a data warehouse (Section 3.3.1). The second considers data warehouses as components of data lakes (Section 3.3.2).

3.3.1 Data lake sourcing a data warehouse

This approach aims to take advantage of the specific characteristics of both data lakes and data warehouses. Since data lakes allow an easier and cheaper storage of large amount of raw data, they can be considered as staging areas or Operational Data Stores (ODSs) (Fang 2015; Russom 2017), i.e., intermediary data stores ahead of data warehouses that gather operational data from several sources before the ETL process takes place.

With a data lake sourcing a data warehouse, possibly with semi-structured data, industrialized OLAP analyses are possible over the lake's data, while on-demand, ad-hoc analyses are still possible directly from the data lake (Fig. 6).

3.3.2 Data warehouse within a data lake

As detailed in Section 3.1.1, Inmon proposes an architecture based on a subdivision of data lakes into so-called data ponds (Inmon 2016). For Inmon, structured data ponds sourced from operational applications are, plain and simple, data warehouses. Thus, this approach acts on a conception of data lakes as extensions of data warehouses.

3.3.3 Discussion

When a data lake sources a data warehouse (Section 3.3.1), there is a clear functional separation, as data warehouses and data lakes are specialized in industrialized and on-demand analyses, respectively. However, this comes with a data siloing issue.

By contrast, the data siloing syndrome can be reduced in Inmon's approach (Section 3.3.2), as all data are managed and processed in a unique global platform. Hence, diverse data can easily be combined through cross-reference analyses, which would be impossible if data were managed separately. In addition, building a data warehouse inside a global data lake may improve data lifecycle control. That is, it should be easier to track, and thus to reproduce processes applied to the data that are ingested in the data warehouse, via the data lake's tracking system.

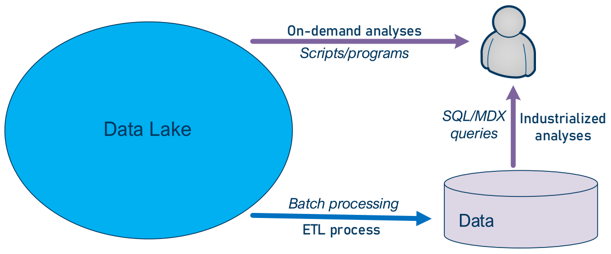

Fig. 6 Data lake and data warehouse architecture

4 Metadata management in data lakes

Data ingested in data lakes bear no explicit schema (Miloslavskaya and Tolstoy 2016), which can easily turn a data lake into a data swamp in the absence of an efficient metadata system (Suriarachchi and Plale 2016). Thence, metadata management plays an essential role in data lakes (Laskowski 2016; Khine and Wang 2017). In this section, we detail the metadata management techniques used in data lakes. First, we identify the metadata that are relevant to data lakes. Then, we review how metadata can be organized. We also investigate metadata extraction tools and techniques. Finally, we provide an inventory of desirable features in metadata systems.

4.1 Metadata categories

We identify in the literature two main typologies of metadata dedicated to data lakes. The first one distinguishes functional metadata, while the second classifies metadata with respect to structural metadata types.

4.1.1 Functional metadata

Oram introduces a metadata classification in three categories, with respect to the way they are gathered (Oram 2015).

1. **Business metadata** are defined as the set of descriptions that make the data more understandable and define business rules. More concretely, these are typically data field names and integrity constraints. Such metadata are usually defined by business users at the data ingestion stage.
2. **Operational metadata** are information automatically generated during data processing. They include descriptions of the source and target data, e.g., data location, file size, number of records, etc., as well as process information.
3. **Technical metadata** express how data are represented, including data format (e.g., raw text, JPEG image, JSON document, etc.), structure or schema. The data structure consists in characteristics such as names, types, lengths, etc. They are commonly obtained from a DBMS for structured data, or via custom techniques during the data maturation stage.

Diamantini et al. enhance this typology with a generic metadata model (Diamantini et al. 2018) and show that business, operational and technical metadata sometimes intersect. For instance, data fields relate both to business and technical metadata, since they are defined in data schemas by business users. Similarly, data formats may be considered as both technical and operational metadata, and so on (Fig. 7).

4.1.2 Structural metadata

In this classification, Sawadogo et al. categorize metadata with respect to the "objects" they relate to Sawadogo et al. (2019). The notion of object may be viewed as a generalization of the dataset concept (Maccioni and Torlone 2018), i.e., an object may be a relational or spreadsheet table in a structured data context, or a simple document (e.g., XML document, image file, video file, textual document, etc.) in a semi-structured or unstructured data context. Thence, we use the term "object" in the remainder of this paper.

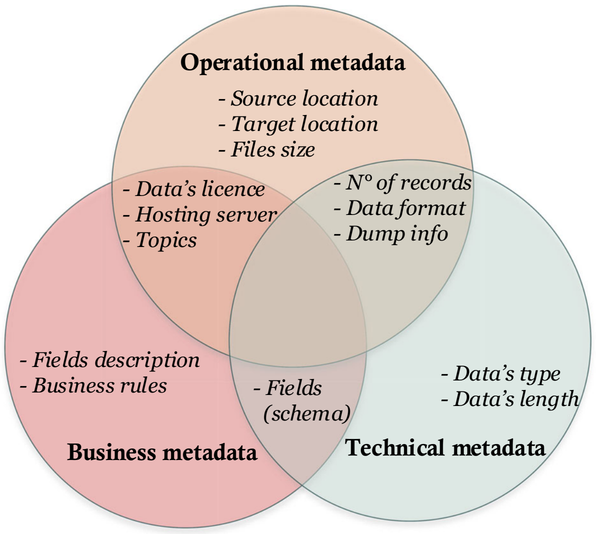

Fig. 7 Functional metadata model (Diamantini et al. 2018)

Intra-object metadata belong to a set of characteristics associated with single objects in the lake. They are subdivided into four main subcategories.

1. **Properties** provide an object's general description. They are generally retrieved from the filesystem as key-value pairs, e.g., file name and size, location, date of last modification, etc.
2. **Previsualization and summary metadata** aim to provide an overview of the content or structure of an object. For instance, metadata can be extracted data schemas for structured and semi-structured data, or wordclouds for textual data.
3. **Version and representation metadata** are made of altered data. When a new data object o' is generated from existing object o in the data lake, o' may be considered as metadata for o. Version metadata are obtained through data updates, while representation metadata come from data refining operations. For instance, a refining operation may consist of vectorizing a textual document into a bag-of-words for further automatic processing.
4. **Semantic metadata** involve annotations that describe the meaning of data in an object. They include such information as title, description, categorization, descriptive tags, etc. They often allow data linking. Semantic metadata can be either generated using semantic resources such as ontologies, or manually added by business users (Hai et al. 2016; Quix et al. 2016).

Inter-object metadata represent links between two or more objects. They are subdivided into three categories.

1. **Object groupings** organize objects into collections. Any object may be associated with several collections. Such links can be automatically deduced from some intra-object metadata such as tags, data format, language, owner, etc.
2. **Similarity links** express the strength of likeness between objects. They are obtained via common or custom similarity measures. For instance, Maccioni and Torlone (2018) define the affinity and joinability measures to express the similarity between semi-structured objects.

3. **Parenthood links** aim to save data lineage, i.e., when a new object is created from the combination of several others, these metadata record the process. Parenthood links are thus automatically generated during data joins.

Global metadata are data structures that provide a context layer to make data processing and analysis easier. Global metadata are not directly associated with any specific object, but potentially concern the entire lake. There are three subcategories of global metadata.

1. **Semantic resources** are knowledge bases such as ontologies, taxonomies, thesauri, etc., which notably help enhance analyses. For instance, an ontology can be used to automatically extend a term-based query with equivalent terms. Semantic resources are generally obtained from the Internet or manually built.
2. **Indexes** (including inverted indexes) enhance term-based or pattern-based data retrieval. They are automatically built and enriched by an indexing system.
3. **Logs** track user interactions with the data lake, which can be simple, e.g., user connection or disconnection, or more complex, e.g., a job running.

4.1.3 Discussion

Oram's metadata classification is the most cited, especially in the industrial literature (Diamantini et al. 2018; LaPlante and Sharma 2016; Ravat and Zhao 2019b; Russom 2017), presumably because it is inspired from metadata categories from data warehouses (Ravat and Zhao 2019a). Thus, its adoption seems easier and more natural for practitioners who are already working with it.

Yet, we favor the second metadata classification, because it includes most of the features defined by Oram's. Business metadata are indeed comparable to semantic metadata. Operational metadata may be considered as logs and technical metadata are equivalent to previsualization metadata. Hence, the structural metadata categorization can be considered as an extension, as well as a generalization, of the functional metadata classification.

Moreover, Oram's classification is quite fuzzy when applied in the context of data lakes. Diamantini et al. indeed show that functional metadata intersect (Section 4.1.1) (Diamantini et al. 2018). Therefore, practitioners who do not know this typology may be confused when using it to identify and organize metadata in a data lake.

Table 1 summarizes commonalities and differences between the two metadata categorizations presented above. The comparison addresses the type of information both inventories provide.

Table 1 Comparison of Oram's and Sawadogo et al's metadata categories

Type of information	Functional metadata	Structural metadata
Basic characteristics of data (size, format, etc.)	✓	✓
Data semantics (tags, descriptions, etc.)	✓	✓
Data history	✓	✓
Data linkage		✓
User interactions		✓

4.2 Metadata modeling

There are in the literature two main approaches to represent a data lake's metadata system. The first, most common approach, adopts a graph view, while the second exploits data vault modeling.

4.2.1 Graph models

Most models that manage data lake metadata systems are based on a graph approach. We identify three main subcategories of graph-based metadata models with respect to the main features they target.

Data provenance-centered graph models mostly manage metadata tracing, i.e., the information about activities, data objects and users who interact with a specific object (Suriarachchi and Plale 2016). In other words, they track the pedigree of data objects (Halevy et al. 2016). Provenance representations are usually built using a directed acyclic graph (DAG) where nodes represent entities such as users, roles or objects (Beheshti et al. 2017; Hellerstein et al. 2017). Edges are used to express and describe interactions between entities, e.g., through a simple timestamp, activity type (read, create, modify) (Beheshti et al. 2017), system status (CPU, RAM, bandwith) (Suriarachchi and Plale 2016) or even the script used (Hellerstein et al. 2017). For instance Fig. 8a shows a basic provenance model with nodes representing data objects and edges symbolizing operations. Data provenance tracking helps ensure the traceability and repeatability of processes in data lakes. Thus, provenance metadata can be used to understand, explain and repair inconsistencies in the data (Beheshti et al. 2017). They may also serve to protect sensitive data, by detecting intrusions (Suriarachchi and Plale 2016).

Similarity-centered graph models describe the metadata system as an undirected graph where nodes are data objects and edges express a similarity between objects. Such a similarity can be specified either through a weighted or unweighted edge. Weighted edges show the similarity strength, when a formal similarity measure is used, e.g., affinity and joinability (Maccioni and Torlone 2018) (Fig. 8b). In contrast, unweighted edges serve to simply detect whether two objects are connected (Farrugia et al. 2016). Such a graph design allows network analyses over a data lake (Farrugia et al. 2016), e.g., discovering communities or calculating the centrality of nodes, and thus their importance in the lake. Another use

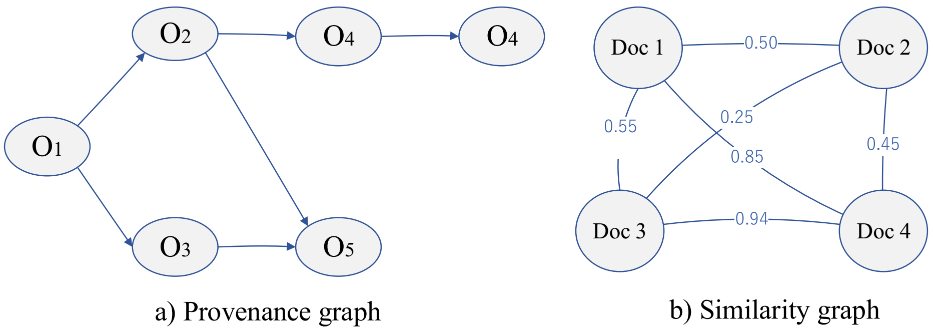

Fig. 8 Sample graph-based metadata models

of data similarity may be to automatically recommend to lake users some data related to the data they currently observe (Maccioni and Torlone 2018).

Composition-centered graph models help decompose each data object into several inherent elements. The lake is viewed as a DAG where nodes represent objects or attributes, e.g., columns, tags, etc., and edges from any node A to any node B express the constraint $B \subseteq A$ (Diamantini et al. 2018; Halevy et al. 2016; Nargesian et al. 2018). This organization helps users navigate through the data (Nargesian et al. 2018). It can also be used as a basis to detect connections between objects. For instance, Diamantini et al. (2018) used a simple string measure to detect links between heterogeneous objects by comparing their respective tags.

4.2.2 Data vault

A data lake aims at ingesting new data possibly bearing various structures. Thus, its metadata system needs to be flexible to easily tackle new data schemas. Nogueira et al. propose the use of a data vault to address this issue (Nogueira et al. 2018). Data vaults are indeed alternative logical models to data warehouse star schemas that, unlike star schemas, allow easy schema evolution (Linstedt 2011). Data vault modeling involves three types of entities (Hultgren 2016).

1. A **hub** represents a business concept, e.g., customer, vendor, sale or product in a business decision system.
2. A **link** represents a relationship between two or more hubs.
3. **Satellites** contain descriptive information associated with a hub or a link. Each satellite is attached to a unique hub or link. In contrast, links or hubs may be associated with any number of satellites.

In Nogueira et al.'s proposal, metadata common to all objects, e.g., title, category, date and location, are stored in hubs; while metadata specific to some objects only, e.g., language for textual documents or publisher for books, are stored in satellites (Fig. 9). Moreover, any new type of object would have its specific metadata stored in a new satellite.

4.2.3 Discussion

Data vault modeling is seldom associated with data lakes in the literature, presumably because it is primarily associated with data warehouses. Yet, this approach ensures metadata schema evolutivity, which is required to build an efficient data lake. Another advantage of data vault modeling is that, unlike graph models, it can be intuitively implemented in a relational DBMS. However, several adaptations are still needed for this model to deal with data linkage as in graph models.

Graph models, though requiring more specific storage systems such as RDF or graph DBMSs, are still advantageous because they allow to automatically enrich the lake with information that facilitate and enhance future analyses. Nevertheless, the three subcategories of graph models need to be all integrated together for this purpose. This remains an open issue because at most two of these graph approaches are simultaneously implemented in metadata systems from the literature (Diamantini et al. 2018; Halevy et al. 2016). The MEDAL metadata model does include all three subcategories of graph models (Sawadogo et al. 2019), but is not implemented yet.

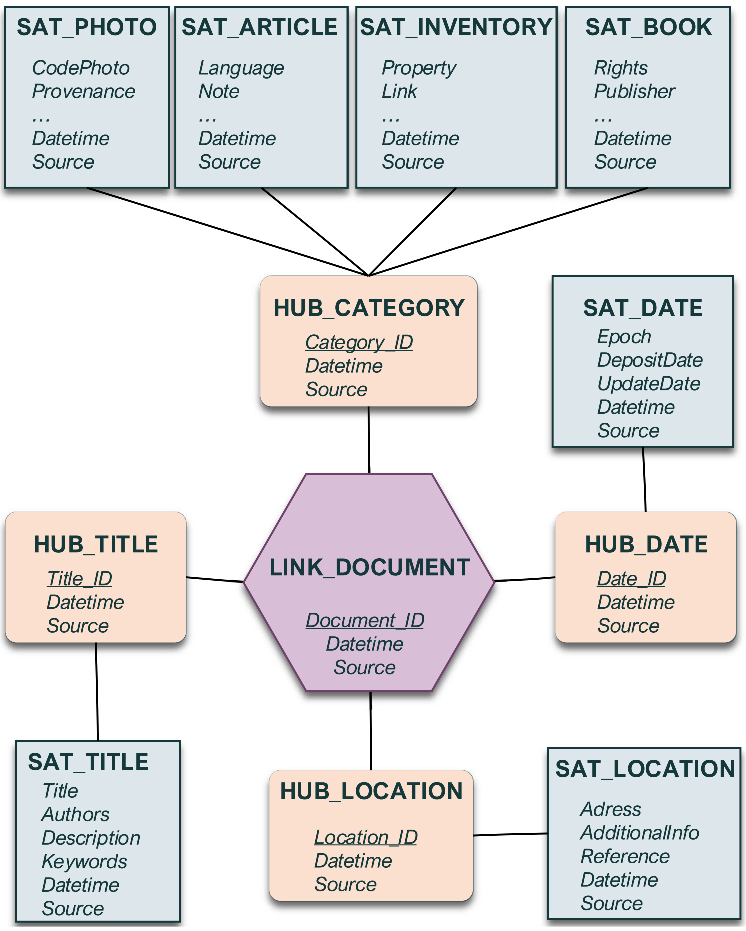

Fig. 9 Sample metadata vault model (Nogueira et al. 2018)

4.3 Metadata generation

Most of the data ingestion tools from Section 3.2.1 can also serve to extract metadata. For instance, Suriarachchi and Plale use Apache Flume to retrieve data provenance in a data lake (Suriarachchi and Plale 2016). Similarly, properties and semantic metadata can be obtained through specialized protocols such as the Comprehensive Knowledge Archive Network (CKAN), an open data storage management system (Terrizzano et al. 2015).

A second kind of technologies is more specific to metadata generation. For instance, Apache Tika helps detect the MIME type and language of objects (Quix et al. 2016). Other tools such as Open Calais and IBM's Alchemy API can also enrich data through inherent entity identification, relationship inference and event detection (Farid et al. 2016).

Ad-hoc algorithms can also generate metadata. For example, Singh et al. show that Bayesian models allow detecting links between data attributes (Singh et al. 2016). Similarly, several authors propose algorithms to discover schemas or constraints in semi-structured data (Beheshti et al. 2017; Klettke et al. 2017; Quix et al. 2016).

Last but not least, Apache Atlas (The Apache Software Foundation 2019), a widely used metadata framework (Russom 2017), features advanced metadata generation methods through so-called hooks, which are native or custom scripts that rely on logs to generate metadata. Hooks notably help Atlas automatically extract lineage metadata and propagate tags on all derivations of tagged data.

4.4 Features of data lake metadata systems

A data lake made inoperable by lack of proper metadata management is called a data swamp (Khine and Wang 2017), data dump (Inmon 2016; Suriarachchi and Plale 2016) or one-way data lake (Inmon 2016), with data swamp being the most common terms. In such a flawed data lake, data are ingested, but can never be extracted. Thus, a data swamp is unable to ensure any analysis. Yet, to the best of our knowledge, there is no objective way to measure or compare the efficiency of data lake metadata systems. Therefore, we first introduce in this section a list of expected features for a metadata system. Then, we present a comparison of eighteen data lake metadata systems with respect to these features.

4.4.1 Feature identification

Sawadogo et al. identify six features that a data lake metadata system should ideally implement to be considered comprehensive (Sawadogo et al. 2019).

1. **Semantic Enrichment (SE)** is also known as semantic annotation (Hai et al. 2016) or semantic profiling (Ansari et al. 2018). It involves adding information such as title, tags, description and more to make the data comprehensible (Terrizzano et al. 2015). This is commonly done using knowledge bases such as ontologies (Ansari et al. 2018). Semantic annotation plays a vital role in data lakes, since it makes the data meaningful by providing informative summaries (Ansari et al. 2018). In addition, semantic metadata could be the basis of link generation between data (Quix et al. 2016). For instance, data objects with the same tags could be considered linked.
2. **Data Indexing (DI)** is commonly used in the information retrieval and database domains to quickly find a data object. Data indexing is done by building and enriching some data structure that enables efficient data retrieval from the lake. Indexing can serve for both simple keyword-based retrieval and more complex querying using patterns. All data, whether structured, semi-structured or unstructured, benefit from indexing (Singh et al. 2016).
3. **Link Generation (LG)** consists in identifying and integrating links between lake data. This can be done either by ingesting pre-existing links from data sources or by detecting new links. Link generation allows additional analyses. For instance, similarity links can serve to recommend to lake users data close to the data they currently use (Maccioni and Torlone 2018). In the same line, data links can be used to automatically detect clusters of strongly linked data (Farrugia et al. 2016).
4. **Data Polymorphism (DP)** is the simultaneous management of several data representations in the lake. A data representation of, e.g., a textual document, may be a tag cloud or a vector of term frequencies. Semi-structured and unstructured data need to be at least partially transformed to be automatically processed (Diamantini et al. 2018). Thus, data polymorphism is relevant as it allows to store and reuse transformed data.

This makes analyses easier and faster by avoiding the repetition of certain processes (Stefanowski et al. 2017).

5. **Data Versioning (DV)** expresses a metadata system's ability to manage update operations, while retaining the previous data states. It is very relevant to data lakes, since it ensures process reproducibility and the detection and correction of inconsistencies (Bhattacherjee and Deshpande 2018). Moreover, data versioning allows branching and concurrent data evolution (Hellerstein et al. 2017).
6. **Usage Tracking (UT)** consists in managing information about user interactions with the lake. Such interactions are commonly creation, read and update operations. This allows to transparently follow the evolution of data objects. In addition, usage tracking can serve for data security, either by explaining data inconsistencies or through intrusion detection. Usage tracking and data versioning are related, since update interactions often induce new data versions. However, they are distinct features as they can be implemented independently (Beheshti et al. 2017; Suriarachchi and Plale 2016).

4.4.2 Metadata system comparison

We present in Table 2 a comparison of eighteen state-of-the-art metadata systems and models with respect to the features they implement (Sawadogo et al. 2019). We distinguish metadata models from implementations. Models are indeed quite theoretical and describe the conceptual organization of metadata. In contrast, implementations follow a

Table 2 Comparison of data lake metadata systems (Sawadogo et al. 2019)

System	Type	SE	DI	LG	DP	DV	UT
SPAR (Fauduet and Peyrard 2010)	♦♯	✓	✓	✓			✓
Alrehamy and Walker (2015)	♦	✓		✓			
Terrizzano et al. (2015)	♦	✓	✓			✓	✓
Constance (Hai et al. 2016)	♦	✓	✓				
GEMMS (Quix et al. 2016)	♢	✓					
CLAMS (Farid et al. 2016)	♦	✓					
Suriarachchi and Plale (2016)	♢				✓		✓
Singh et al. (2016)	♦	✓	✓	✓	✓		
Farrugia et al. (2016)	♦			✓			
GOODS (Halevy et al. 2016)	♦	✓	✓	✓		✓	✓
CoreDB (Beheshti et al. 2017)	♦		✓				✓
Ground (Hellerstein et al. 2017)	♢♯	✓	✓			✓	✓
KAYAK (Maccioni and Torlone 2018)	♦	✓	✓	✓			
CoreKG (Beheshti et al. 2018)	♦	✓	✓	✓	✓		✓
Diamantini et al. (2018)	♢	✓		✓	✓		
Mehmood et al. (2019)	♦	✓	✓				
CODAL (Sawadogo et al. 2019)	♦	✓	✓	✓	✓		
MEDAL (Sawadogo et al. 2019)	♢	✓	✓	✓	✓	✓	✓

♦ : Implementations ♢ : Metadata models

♯ : Model or implementation akin to a data lake more operational approach, but are usually little detailed, mainly focusing on a description of the resulting system instead of the applied methodology. This comparison also considers metadata systems that are not explicitly associated with the concept of data lake by their authors, but whose characteristics allow to be considered as such, e.g., the Ground metadata model (Hellerstein et al. 2017).

The comparison shows that the most comprehensive metadata system with respect to the features we propose is MEDAL, with all features covered. However, it is not implemented yet. The next best systems are GOODS and CoreKG, with five out of six features implemented. However, they are black box metadata systems, with few details on metadata conceptual organization. Thus, the Ground metadata model may be preferred, since it is much more detailed and almost as complete (four out of six features).

Eventually, two of the six features defined in Section 4.4.1 may be considered advanced. Data polymorphism and data versioning are indeed mainly found in the most complete systems such as GOODS, CoreKG and Ground. Their absence from most of metadata systems can thus be attributed to implementation complexity.

5 Pros and cons of data lakes

In this section, we account for the benefits of using a data lake instead of more traditional data management systems, but also identify the pitfalls that may correspond to these expected benefits.

An important motivating feature in data lakes is **cheap storage**. Data lakes are ten to one hundred times less expensive to deploy than traditional decision-oriented databases. This can be attributed to the usage of open-source technologies such as HDFS (Khine and Wang 2017; Stein and Morrison 2014). Another reason is that the cloud storage often used to build data lakes reduces the cost of storage technologies. That is, the data lake owner pays only for actually used resources. However, the use of HDFS may still fuel **misconceptions**, with the concept of data lake remaining ambiguous for many potential users. It is indeed often considered either as a synonym or a marketing label closely related to the HDFS technology (Alrehamy and Walker 2015; Grosser et al. 2016).

Another feature that lies at the core of the data lake concept is **data fidelity**. Unlike in traditional decision-oriented databases, original data are indeed preserved in a data lake to avoid any data loss that could occur from data preprocessing and transformation operations (Ganore 2015; Stein and Morrison 2014). Yet, data fidelity induces a high risk of **data inconsistency** in data lakes, due to data integration from multiple, disparate sources without any transformation (O'Leary 2014).

One of the main benefits of data lakes is that they allow exploiting and analyzing **unstructured data** (Ganore 2015; Laskowski 2016; Stein and Morrison 2014). This is a significant advantage when dealing with big data, which are predominantly unstructured (Miloslavskaya and Tolstoy 2016). Moreover, due to the schema-on-read approach, data lakes can comply with any data type and format (Cha et al. 2018; Ganore 2015; Khine and Wang 2017; Madera and Laurent 2016). Thence, data lakes enable a wider range of analyses than traditional decision-oriented databases, i.e., data warehouses and datamarts, and thus show better **flexibility and agility**. However, although the concept of data lake dates back from 2010, it has only been put in practice in the mid-2010's. Thus, implementations vary, are still maturing and there is a **lack of methodological and technical standards**, which sustains confusions about data lakes. Finally, due to the absence of an explicit schema, **data access services** and APIs are essential to enable knowledge extraction in a data lake. In other words, a data access service is a must to successfully build a data lake (Alrehamy and Walker 2015; Inmon 2016), while such a service is not always present.

Next, an acclaimed advantage of data lakes over data warehouses is **real-time data ingestion**. Data are indeed ingested in data lakes without any transformation, which avoids any time lag between data extraction from sources and their ingestion in the data lake (Ganore 2015; Laskowski 2016). But as a consequence, a data lake requires an efficient **metadata system** for ensuring data access. However, the problem lies in the “how”, i.e., the use of inappropriate methods or technologies to build the metadata system can easily turn the data lake into an inoperable data swamp (Alrehamy and Walker 2015).

More technically, data lakes and related analyses are typically implemented using distributed technologies, e.g., HDFS, MapReduce, Apache Spark, Elasticsearch, etc. Such technologies usually provide a **high scalability** (Fang 2015; Miloslavskaya and Tolstoy 2016). Furthermore, most technologies used in data lakes have replication mechanisms, e.g., Elasticsearch, HDFS, etc. Such technologies allow a high resilience to both hardware and software failure and enforce **fault tolerance** (John and Misra 2017).

Eventually, data lakes are often viewed as sandboxes where analysts can “play”, i.e., access and prepare data so as to perform various, specific, **on-the-fly analyses** (Russom 2017; Stein and Morrison 2014). However, such a scenario requires **expertise**. Data lake users are indeed typically data scientists (Khine and Wang 2017; Madera and Laurent 2016), which contrasts with traditional decision systems, where business users are able to operate the system. Thus, a data lake induces a greater need for specific, and therefore more expensive, profiles. Data scientists must indeed master a wide knowledge and panoply of technologies.

Moreover, with the integration in data lakes of structured, semi-structured and unstructured, expert data scientists can discover links and **correlations between heterogeneous data** (Ganore 2015). Data lakes also allow to easily integrate data “as is” from external sources, e.g., the Web or social media. Such external data can then be associated with proprietary data to generate new knowledge through **cross-analyses** (Laskowski 2016). However, several statistical and Artificial Intelligence (AI) approaches are not originally designed for parallel operations, nor for streaming data, e.g., K-means or K-Nearest Neighbors. Therefore, it is necessary to **readjust classical statistical and AI approaches** to match the distributed environments often used in data lakes (O’Leary 2014), which sometimes proves difficult.

6 Conclusion

In this survey paper, we establish a comprehensive state of the art of the different approaches to design, and conceptually build a data lake. First, we state the definitions of the data lake concept and complement the best existing one. Second, we investigate alternative architectures and technologies for data lakes, and propose a new typology of data lake architectures. Third, we review and discuss the metadata management techniques used in data lakes. We notably classify metadata and introduce the features that are necessary to achieve a full metadata system. Fourth, we discuss the pros and cons of data lakes. Fifth, we summarize by a mind map the key concepts introduced in this paper (Fig. 10).

Eventually, in echo to the topics we chose *not* to address in this paper (Section 1), we would like to open the discussion on important current research issues in the field of data lakes.

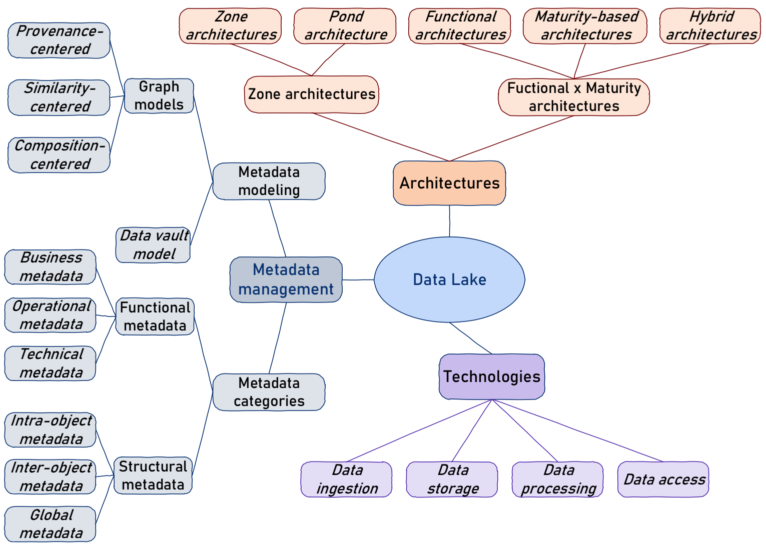

Fig. 10 Key concepts investigated in this survey

Data integration and transformation have long been recurring issues. Though delayed, they are still present in data lakes and made even more challenging by big data volume, variety, velocity and lack of veracity. Moreover, when transforming such data, User-Defined Functions (UDFs) must very often be used (MapReduce tasks, typically). In ETL and ELT processes, UDFs are much more difficult to optimize than classical queries, an issue that is not addressed yet by the literature (Stefanowski et al. 2017).

With data storage solutions currently going beyond HDFS in data lakes, **data interrogation** through metadata is still a challenge. Multistores and polystores indeed provide unified solutions for structured and semi-structured data, but do not address unstructured data, which are currently queried separately through index stores. Moreover, when considering data gravity (Madera and Laurent 2016), virtual data integration becomes a relevant solution. Yet, mediation approaches are likely to require new, big data-tailored query optimization and caching approaches (Quix and Hai 2018; Stefanowski et al. 2017).

Unstructured datasets although unanimously acknowledged as ubiquitous and sources of crucial information, are very little specifically addressed in data lake-related literature. Index storage and text mining are usually mentioned, but there is no deep thinking about global querying or analysis solutions. Moreover, exploiting other types of unstructured data but text, e.g., images, sounds and videos, is not even envisaged as of today.

Again, although all actors in the data lake domain stress the importance of **data governance** to avoid a data lake turning into a data swamp, data quality, security, life cycle management and metadata lineage are viewed as risks rather than issues to address *a priori* in data lakes (Madera and Laurent 2016). Data governance principles are indeed currently seldom turned into actual solutions.

Finally, **data security** is currently addressed from a technical point of view in data lakes, i.e., through access and privilege control, network isolation, e.g., with Docker tools (Cha et al. 2018), data encryption and secure search engines (Maroto 2018). However, beyond these issues and those already addressed by data governance (integrity, consistency, availability) and/or related to the European General Data Protection Regulation (GDPR), by storing and cross-analyzing large volumes of various data, data lakes allow mashups that potentially induce serious breaches of data privacy (Joss 2016). Such issues are still researched as of today.

Acknowledgments The research accounted for in this paper was funded by the Université Lumière Lyon 2 and the Auvergne-Rhône-Alpes Region through the COREL and AURA-PMI projects, respectively. The authors also sincerely thank the anonymous reviewers of this paper for their constructive comments and suggestions.

References

Alrehamy, H., & Walker, C. (2015). Personal data lake with data gravity pull. In *IEEE 5Th international conference on big data and cloud computing(BDCloud 2015), Dalian, China, IEEE computer society washington, vol. 88, pp. 160–167*. https://doi.org/10.1109/BDCloud.2015.62.

Ansari, J.W., Karim, N., Decker, S., Cochez, M., Beyan, O. (2018). Extending data lake metadata management by semantic profiling. In *2018 Extended semantic web conference (ESWC 2018), Heraklion, Crete, Greece, pp. 1–15*.

Beheshti, A., Benatallah, B., Nouri, R., Chhieng, V.M., Xiong, H., Zhao, X. (2017). CoreDB: A Data Lake Service. In *2017 ACM On conference on information and knowledge management (CIKM 2017), Singapore, Singapore, ACM, pp. 2451–2454*. https://doi.org/10.1145/3132847.3133171.

Beheshti, A., Benatallah, B., Nouri, R., Tabebordbar, A. (2018). CoreKG: A knowledge lake service. *Proceedings of the VLDB Endowment*, *11*(12), 1942–1945. https://doi.org/10.14778/3229863.3236230.

Bhattacherjee, S., & Deshpande, A. (2018). RSTore: A distributed multi-version document store. In *IEEE 34Th international conference on data engineering (ICDE), Paris, France, pp. 389–400*. https://doi.org/10.1109/ICDE.2018.00043.

Cha, B., Park, S., Kim, J., Pan, S., Shin, J. (2018). International network performance and security testing based on distributed abyss storage cluster and draft of data lake framework. *Hindawi Security and Communication Networks*, *2018*, 1–14. https://doi.org/10.1155/2018/1746809.

Chessell, M., Scheepers, F., Nguyen, N., van Kessel, R., van der Starre, R. (2014). Governing and managing big data for analytics and decision makers. IBM.

Couto, J., Borges, O., Ruiz, D., Marczak, S., Prikladnicki, R. (2019). A mapping study about data lakes: an improved definition and possible architectures. In *31St international conference on software engineering and knowledge engineering (SEKE 2019), Lisbon, Portugal, pp. 453–458*. https://doi.org/10.18293/SEKE2019-129.

Diamantini, C., Giudice, P.L., Musarella, L., Potena, D., Storti, E., Ursino, D. (2018). A new metadata model to uniformly handle heterogeneous data lake sources. In: New trends in databases and information systems - ADBIS 2018 Short Papers and Workshop, Budapest, Hungary, pp. 165–177. https://doi.org/10.1007/978-3-030-00063-9_17.

Dixon, J. (2010). Pentaho, Hadoop, and data lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/.

Fang, H. (2015). Managing Data Lakes in Big Data era: What's a data lake and why has it became popular in data management ecosystem. In *5Th annual IEEE international conference on cyber technology in automation, control and intelligent systems (CYBER 2015), Shenyang, China, IEEE, pp. 820–824*. https://doi.org/10.1109/CYBER.2015.7288049.

Farid, M., Roatis, A., Ilyas, I.F., Hoffmann, H.F., Chu, X. (2016). CLAMS: Bringing quality to data lakes. In *2016 International conference on management of data (SIGMOD 2016), San Francisco, CA, USA, ACM, pp. 2089–2092*. https://doi.org/10.1145/2882903.2899391.

Farrugia, A., Claxton, R., Thompson, S. (2016). Towards social network analytics for understanding and managing enterprise data lakes. In *Advances in social networks analysis and mining (ASONAM 2016), San Francisco, CA, USA, IEEE, pp. 1213–1220*. https://doi.org/10.1109/ASONAM.2016.7752393.

Fauduet, L., & Peyrard, S. (2010). A data-first preservation strategy: Data management in spar. In: 7th international conference on preservation of digital objects (iPRES 2010), Vienna, Autria, pp. 1–8. http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/fauduet-13.pdf.

Ganore, P. (2015). Introduction To The Concept Of Data Lake And Its Benefits. https://www.esds.co.in/blog/introduction-to-the-concept-of-data-lake-and-its-benefits.

Giebler, C., Gröger, C., Hoos, E., Schwarz, H., Mitschang, B. (2019). Leveraging the data lake - current state and challenges. In *Proceedings of the 21st international conference on big data analytics and knowledge discovery (DaWaK* (p. 2019). Austria: Linz.

Grosser, T., Bloeme, J., Mack, M., Vitsenko, J. (2016). Hadoop and data lakes: Use cases, benefits and limitations business application research center – BARC GmbH.

Hai, R., Geisler, S., Quix, C. (2016). Constance: an intelligent data lake system. In *International conference on management of data (SIGMOD 2016), San Francisco, CA, USA, ACM Digital Library, pp. 2097–2100*. https://doi.org/10.1145/2882903.2899389.

Hai, R., Quix, C., Zhou, C. (2018). Query rewriting for heterogeneous data lakes. In: 22nd European conference on advances in databases and information systems (ADBIS 2018), Budapest, Hungary, LNCS, vol. 11019, pp. 35–49. Springer. https://doi.org/10.1007/978-3-319-98398-1_3.

Halevy, A.Y., Korn, F., Noy, N.F., Olston, C., Polyzotis, N., Roy, S., Whang, S.E. (2016). Goods: organizing google's datasets. In *Proceedings of the 2016 international conference on management of data (SIGMOD 2016), San Francisco, CA, USA, pp. 795–806*. https://doi.org/10.1145/2882903.2903730.

Hellerstein, J.M., Sreekanti, V., sGonzalez, J.E., Dalton, J., Dey, A., Nag, S., Ramachandran, K., Arora, S., Bhattacharyya, A., Das, S., Donsky, M., Fierro, G., She, C., Steinbach, C., Subramanian, V., Sun, E. (2017). Ground: A data context service. In: 8th Biennial Conference on Innovative Data Systems Research (CIDR 2017), Chaminade, CA, USA. http://cidrdb.org/cidr2017/papers/p111-hellerstein-cidr17.pdf.

Hultgren, H. (2016). Data Vault modeling guide: Introductory guide to data vault modeling. Genessee Academy, USA.

Inmon, B. (2016). Data Lake architecture: Designing the Data Lake and avoiding the garbage dump. Technics Publications.

John, T., & Misra, P. (2017). Data Lake for enterprises: Lambda architecture for building enterprise data systems. Packt Publishing.

Joss, A. (2016). The rise of the GDPR data lake. https://blogs.informatica.com/2016/06/16/rise-gdpr-data-lake/.

Khine, P.P., & Wang, Z.S. (2017). Data lake: A new ideology in big data era. In *4Th international conference on wireless communication and sensor network (WCSN 2017), Wuhan, China, ITM web of conferences, vol. 17, pp. 1–6*. https://doi.org/10.1051/itmconf/2018170302.

Klettke, M., Awolin, H., Stürl, U., Müller, D., Scherzinger, S. (2017). Uncovering the evolution history of data lakes. In *2017 IEEE International conference on big data (BIGDATA 2017), Boston, MA, USA, pp. 2462–2471*. https://doi.org/10.1109/BigData.2017.8258204.

LaPlante, A., & Sharma, B. (2016). Architecting data lakes data management architectures for advanced business use cases. O'Reilly Media Inc.

Laskowski, N. (2016). Data lake governance: A big data do or die. https://searchcio.techtarget.com/feature/Data-lake-governance-A-big-data-do-or-die.

Leclercq, E., & Savonnet, M. (2018). A tensor based data model for polystore: an application to social networks data. In *Proceedings of the 22nd international database engineering & applications symposium (IDEAS 2018), Villa San Giovanni, Italy, pp. 110–118*. https://doi.org/10.1145/3216122.3216152.

Linstedt, D. (2011). Super charge your data warehouse: Invaluable data modeling rules to implement your data. Vault CreateSpace Independent Publishing.

Maccioni, A., & Torlone, R. (2018). KAYAK: A framework for just-in-time data preparation in a data lake. In: International Conference on Advanced Information Systems Engineering (CAiSE 2018), Tallin, Estonia, pp. 474–489. https://doi.org/10.1007/978-3-319-91563-0_29.

Madera, C., & Laurent, A. (2016). The next information architecture evolution: the data lake wave. In *8Th international conference on management of digital ecosystems (MEDES 2016), Biarritz, France, pp. 174–180*. https://doi.org/10.1145/3012071.3012077.

Maroto, C. (2018). Data lake security – four key areas to consider when securing your data lake. https://www.searchtechnologies.com/blog/data-lake-security.

Mathis, C. (2017). Data lakes. *Datenbank-Spektrum*, *17*(3), 289–293. https://doi.org/10.1007/s13222-017-0272-7.

Mehmood, H., Gilman, E., Cortes, M., Kostakos, P., Byrne, A., Valta, K., Tekes, S., Riekki, J. (2019). Implementing big data lake for heterogeneous data sources. In *2019 IEEE 35Th international conference on data engineering workshops (ICDEW), pp. 37–44*. https://doi.org/10.1109/ICDEW.2019.00-37.

Miloslavskaya, N., & Tolstoy, A. (2016). Big data, fast data and data lake concepts. In *7Th annual international conference on biologically inspired cognitive architectures (BICA 2016), NY, USA, Procedia Computer Science, vol. 88, pp. 1–6*. https://doi.org/10.1016/j.procs.2016.07.439.

Nargesian, F., Pu, K.Q., Zhu, E., Bashardoost, B.G., Miller, R.J. (2018). Optimizing Organizations for Navigating Data Lakes. arXiv:1812.07024.

Nogueira, I., Romdhane, M., Darmont, J. (2018). Modeling data lake metadata with a data vault. In *22Nd international database engineering and applications symposium (IDEAS 2018), Villa San Giovanni, Italia* (pp. 253-261). New York: ACM.

O'Leary, D.E. (2014). Embedding AI and crowdsourcing in the big data lake. *IEEE Intelligent Systems*, *29*(5), 70–73. https://doi.org/10.1109/MIS.2014.82.

Oram, A. (2015). Managing the data lake. Zaloni.

Pathirana, N. (2015). Modeling industrial and cultural heritage data. Master's thesis, université lumière Lyon 2 France.

Quix, C., & Hai, R. (2018). *Data lake*, (pp. 1–8). Berlin: Springer International Publishing. https://doi.org/10.1007/978-3-319-63962-8_7-1.

Quix, C., Hai, R., Vatov, I. (2016). Metadata extraction and management in data lakes with GEMMS. *Complex Systems Informatics and Modeling Quarterly*, *9*, 289–293. https://doi.org/10.7250/csimq.2016-9.04.

Ravat, F., & Zhao, Y. (2019). Data lakes: Trends and perspectives. In *30Th international conference on database and expert systems applications (DEXA* (p. 2019). Austria: Linz.

Ravat, F., & Zhao, Y. (2019). Metadata management for data lakes. In *23Rd european conference on advances in databases and information systems (ADBIS* (p. 2019). Slovenia: Bled.

Russom, P. (2017). Data lakes purposes, Practices, Patterns, and Platforms. TDWI research.

Sawadogo, P.N., Kibata, T., Darmont, J. (2019). Metadata management for textual documents in data lakes. In *21St international conference on enterprise information systems (ICEIS 2019), Heraklion, Crete, Greece, pp. 72–83*. https://doi.org/10.5220/0007706300720083.

Sawadogo, P.N., Scholly, E., Favre, C., Ferey, É., Loudcher, S., Darmont, J. (2019). Metadata systems for data lakes: models and features. In *BI And big data applications - ADBIS 2019 Short Papers and Workshop, Bled, Slovenia*.

Singh, K., Paneri, K., Pandey, A., Gupta, G., Sharma, G., Agarwal, P., Shroff, G. (2016). Visual bayesian fusion to navigate a data lake. In *19Th international conference on information fusion (FUSION 2016), Heidelberg, Germany, IEEE, pp. 987–994*.

Sirosh, J. (2016). The intelligent data lake. https://azure.microsoft.com/fr-fr/blog/the-intelligent-data-lake/.

Stefanowski, J., Krawiec, K., Wrembel, R. (2017). Exploring complex and big data. *International Journal of Applied Mathematics and Computer Science*, *27*(4), 669–679. https://doi.org/10.1515/amcs-2017-0046.

Stein, B., & Morrison, A. (2014). The enterprise data lake: Better integration and deeper analytics. PWC Technology Forecast. http://www.smallake.kr/wp-content/uploads/2017/03/20170313_074222.pdf.

Suriarachchi, I., & Plale, B. (2016). Crossing analytics systems: A case for integrated provenance in data lakes. In *12Th IEEE international conference on e-science (e-science 2016), Baltimore, MD, USA, pp. 349–354*. https://doi.org/10.1109/eScience.2016.7870919.

Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E. (2015). Data Wrangling: The Challenging Journey from the Wild to the Lake. In: 7th Biennial conference on innovative data systems research (CIDR 2015), Asilomar, CA, USA, pp. 1–9. http://cidrdb.org/cidr2015/Papers/CIDR15_Paper2.pdf.

Tharrington, M. (2017). The dzone guide to big data, data science & advanced Analytics. DZone.

The Apache Software Foundation (2019). Apache atlas – data governance and metadata framework for Hadoop. https://atlas.apache.org/.

Tiao, S. (2018). Object storage for big data: What Is It? And Why Is It Better? https://blogs.oracle.com/bigdata/what-is-object-storage.

Zikopoulos, P., deRoos, D., Bienko, C., Buglio, R., Andrews, M. (2015). Big data bayond the hype. McGraw-Hill Education.