University of Stuttgart - Institute for Parallel and Distributed Systems / AS

# The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture

Corinna Giebler, Christoph Gröger, Eva Hoos, Rebecca Eichler, Holger Schwarz, Bernhard Mitschang

In: Proceedings der 19. Fachtagung für Datenbanksysteme für Business, Technologie und Web (BTW 2021)

BIBTEX:
@inproceedings{Giebler2021,
author = {Giebler, Corinna and Gr{\"{o}}ger, Christoph and Hoos, Eva and Eichler, Rebecca and Schwarz, Holger and Mitschang, Bernhard},
booktitle = { Proceedings der 19. Fachtagung für Datenbanksysteme für Business, Technologie und Web (BTW 2021)},
title = {{ The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture}},
year = {2021},
doi = {}
}



The final authenticated version is available online at https://doi.org/[insert DOI].

Herausgeber et al. (Hrsg.): BTW,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 1

# The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture

Corinna Giebler[1], Christoph Gröger[2], Eva Hoos[2], Rebecca Eichler[1], Holger Schwarz[1], Bernhard Mitschang[1]

**Abstract:** During recent years, data lakes emerged as a way to manage large amounts of heterogeneous data for modern data analytics. Although various work on individual aspects of data lakes exists, there is no comprehensive data lake architecture yet. Concepts that describe themselves as a "data lake architecture" are only partial. In this work, we introduce the data lake architecture framework. It supports the definition of data lake architectures by defining nine architectural aspects, i.e., perspectives on a data lake, such as data storage or data modeling, and by exploring the interdependencies between these aspects. The included methodology helps to choose appropriate concepts to instantiate each aspect. To evaluate the framework, we use it to configure an exemplary data lake architecture for a real-world data lake implementation. This final assessment shows that our framework provides comprehensive guidance in the configuration of a data lake architecture.

**Keywords:** Data Lake; Data Lake Architecture; Framework

## 1 Introduction

In recent years, data lakes emerged as platforms for big data management and analyses [Ma17b]. They are used in various domains, e.g., healthcare [RZ19] or air traffic [Ma17a], and enable organizations to explore the value of data using advanced analytics such as machine learning [Ma17b]. To this end, data of heterogeneous structure are stored in their raw format to allow analysis without predefined use cases.

However, implementing a data lake in practice proves challenging, as no comprehensive data lake architecture exists so far. Such an architecture specifies, e.g., the data storage or data modeling to be used. In this work, we define *comprehensive* as "*all necessary architectural aspects of a data lake and their interdependencies are covered*". An architectural aspect is a perspective on a data lake architecture, such as data modeling or infrastructure. To define a comprehensive data lake architecture, multiple such aspects have to be considered. While some concepts are called "data lake (*reference*) architecture" (e.g., in [Sh18]) by their authors, they only focus on a subset of necessary architectural aspects.

In addition, the possible applications of data lakes are very diverse. A data lake might only process batch data [Ma17a] or both batch data and data streams [MM18]. It might be limited

[1] Universität Stuttgart, IPVS, 70569 Stuttgart, Germany {firstname.lastname}@ipvs.uni-stuttgart.de
[2] Robert Bosch GmbH, 70469 Stuttgart, Germany {firstname.lastname}@de.bosch.com

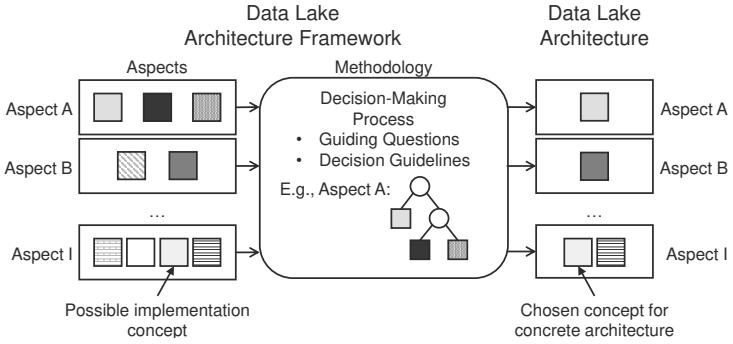


Fig. 1: The data lake architecture framework contains possible implementation concepts. To configure a data lake architecture, concrete concepts are chosen from the framework using the contained methodology.

to data scientists and advanced analytics [ML16] or additionally support traditional data warehousing [Ma17b]. Depending on the kind of data in the data lake and on the scenario in which it is used, different requirements are posed on a data lake architecture. Thus, defining a generic and universally applicable data lake architecture proves difficult. Instead, we propose the data lake architecture framework (DLAF) as a foundation for comprehensive data lake architecture development. Architecture frameworks exist in varoius domains, e.g., the Zachman framework [Za87] provides both guidance and a methodology to define an appropriate information system architecture. However, in the context of data lakes, we are not aware of such an approach. Fig. 1 depicts the connection between the framework and a configured data lake architecture. The guidance provided by the DLAF is threefold: 1) It defines the architectural aspects necessary for a data lake, e.g., data modeling, 2) it associates each aspect with a set of possible implementation concepts, e.g., data vault [Li12], and 3) it provides a methodology that helps picking appropriate concepts for a data lake architecture while also considering interdependencies between aspects, e.g., between data modeling and infrastructure. A data lake architecture derived from the framework can be understood as a DLAF instance. In this paper, we make the following contributions:

- From a categorization of literature on data lakes, we identify their necessary architectural aspects. We use these aspects to build the data lake architecture framework, which serves as a support for the configuration of a comprehensive data lake architecture.
- We present a methodology as part of the data lake architecture framework that guides the development of a specific data lake architecture from the aspects in the framework.
- We assess the data lake architecture framework with regards to its comprehensiveness and applicability. This assessment shows that the data lake architecture framework is not missing any important aspects for data lakes, and that its methodology is applicable in practice to configure a comprehensive data lake architecture.

- We show how the framework can be used to configure a comprehensive data lake architecture, to evaluate existing data lake architectures, and to extend incomprehensive data lake architectures to become comprehensive.

The remainder of this work is structured as follows: Sect. 2 discusses related work on data lake architectures and architecture frameworks. Sect. 3 describes the developed DLAF, before Sect. 4 presents the contained methodology for its use. Sect. 5 assesses the framework by analyzing existing data lake implementations and by defining an exemplary comprehensive data lake architecture for real-world industry. Finally, Sect. 6 concludes the work.

## 2 Related Work

In literature, multiple so called "data lake architectures" are proposed (e.g., in [In16; JQ17; RZ19; Sh18]). The goal of these architectures is to be generic. Sawadogo and Darmont differentiate three kinds of data lake architectures [SD20]: 1) *functional architectures* that cover data ingestion and storage, e.g., [JQ17], 2) *data maturity-based architectures*, where data are organized according to their refinement level, e.g., [Sh18], and 3) *hybrid architectures* that combine both. As functional architectures and data maturity-based architectures focus only on singular aspects of the data lake [SD20], they do not qualify as comprehensive data lake architectures. We thus only consider hybrid architectures in the remainder of this section. Sawadogo and Darmont name two hybrid architectures: Inmon's data pond architecture [In16] and Ravat's data lake functional architecture [RZ19]. To the best of our knowledge, no other generic hybrid architectures are available.

However, defining data ingestion, data storage, and data organization is not sufficient for a data lake. Examples for further aspects of importance are data modeling and metadata management [Gi19a]. Neither Inmon's nor Ravat's architecture cover these additional aspects. While there is work on both data modeling and metadata management in data lakes (e.g., [Ei20; HGQ16; Ho17; NRD18]), it focuses only on singular aspects. Overall, none of the generic data lake architectures in literature is comprehensive.

In addition to the generic architectures, there are hybrid data lake architectures in specific implementations, e.g., in [Ma17a; MM18]. These architectures are however tailored to specific use cases and are not applicable as generic data lake architectures. Thus, to the best of our knowledge, there is no guidance for defining a comprehensive data lake architecture.

## 3 Aspects forming the Data Lake Architecture Framework

To address the lack of support for configuring a comprehensive data lake architecture, we present the data lake architecture framework (DLAF) as a foundation for such a configuration. The framework consists of two parts: I) It describes necessary architectural aspects of a

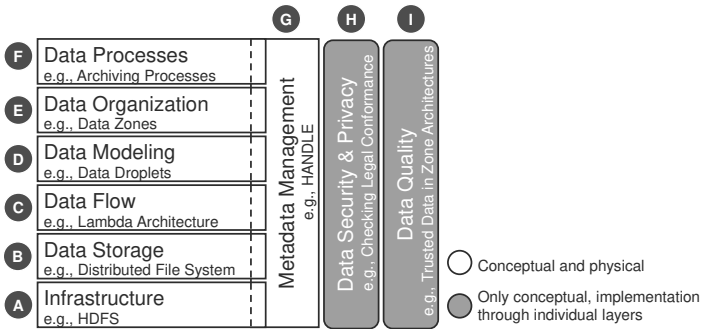


Fig. 2: The data lake architecture framework consists of nine data lake aspects that have to be considered when creating a comprehensive data lake architecture.

data lake. In doing so, it defines the scope for a comprehensive data lake architecture. This section first describes the DLAF aspects that represent the different architectural aspects (Sect. 3.1) before detailing on their interdependencies (Sect. 3.2). II) Moreover, the DLAF includes a methodology to configure a comprehensive data lake architecture, guiding the selection of appropriate concepts for each aspect. This methodology is presented in Sect. 4.

Each part of the framework in Fig. 2 represents one architectural aspect of data lakes associated with a set of concepts for its implementation (cf. Fig. 1). The architectural aspects included in the DLAF were derived by clustering the results of a thorough literature review on concepts for data lake implementation (cf. [Gi19a]). The nine resulting clusters we formulated into disjoint architectural aspects depicted in Fig. 2: A) infrastructure, B) data storage, C) data flow, D) data modeling, E) data organization, F) data processes, G) metadata management, H) data security & privacy, and I) data quality. These aspects can neither be combined further, as they describe different perspectives on the data lake, nor did we find further aspects to be considered. Aspects A-F are sorted by their abstraction degree, with infrastructure as the most physical aspect at the bottom and data processes as the most abstract aspect at the top. Aspects G-I span all of these aspects. We differentiate between aspects that consist of a concept and its physical implementation (white), and aspects that comprise only a conceptual view but are implemented through other aspects (grey). For example, if the data security & privacy aspect requires data encryption, the infrastructure has to offer this functionality. The following paragraphs explain the architectural aspects.

### 3.1 DLAF Aspects

*A) Infrastructure.* The infrastructure aspect comprises concepts for the physical implementation of the data lake. The focus lies on concrete storage systems and tools, e.g.,

HDFS[3] as distributed file system or MySQL[4] as a relational database, and their deployment on-premise or in the cloud. An example is given by [Zi15], who use *Hadoop*[5] and *DB2*[6]. As an exemplary concept for deployment, *hybrid data lakes* [Lo16] are data lakes built both on-premise and in the cloud.

*B) Data Storage.* The data storage focuses on the types of systems and tools used for data storage and processing (e.g., *file systems* and *NoSQL databases*, or *batch processing* and *stream processing tools*). In contrast to the infrastructure aspect, no specific tools are chosen. Exemplary data lakes are built on a *single distributed file system* (e.g., [Ma17a]) or on *multiple storage* systems (e.g., [Zi15]).

*C) Data Flow.* The data flow aspect covers the architecture and interaction for the two modes of data movement that may occur in a data lake: batch data and streaming data. Batch data are persistently stored in a storage system and are processed in large volumes [CY15]. In contrast, streaming (or real-time) data are continuously delivered into the data lake and typically need to be processed immediately [CY15]. Streaming data can also become batch data if it is stored for later use. Examples for data flow concepts are hybrid processing architectures such as the *Lambda architecture* [MW15] or *BRAID* [Gi18].

*D) Data Modeling.* The data modeling aspect describe whether and how data are modeled within the data lake. Typically, the modeling technique used will differ depending on the data's characteristics and their usage. Examples of data modeling techniques applicable in data lakes are *data droplets* [Ho17] or *data vault* [Li12].

*E) Data Organization.* The data organization aspect defines the conceptual set-up within the data lake. To this end, associated concepts describe what data can be found where and what state they are in (e.g., raw or pre-processed). Examples are the *data pond architecture* [In16], the *zone architectures* (e.g., [Gi20; Sh18]), *Jarke's and Quix' conceptual architecture* [JQ17], and *data meshes* for semantical data organization [De19].

*F) Data Processes.* The data processes aspect comprises all concepts that focus on data movement and processing. Data processes can be divided into processes for *data lifecycle management* and for *data pipelining*. *Data lifecycle management processes* manage the data from creation and obtaining to disposure [DA17]. These processes have to be carefully defined and standardized within a data lake to facilitate self-service, ensure data usability, and support legal compliance. In contrast, *data pipelining processes* focus on the technical ingestion, movement, and processing of data, such as extract-transform-load (ETL) processes. They are used to describe, e.g., how data move between zones in a zone model. In contrast to the data flow aspect, the data processes aspect rather describes what is done with data instead of focusing on the characteristics of the data themselves. An exemplary data lifecycle management process for data lakes is the *archiving process* given in [Ch15]. Examples for

[3] hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
[4] www.mysql.com/de/
[5] hadoop.apache.org/
[6] www.ibm.com/analytics/db2

data pipelining can be found in most zone architectures (e.g., in [Gi20]), or in Jarke's and Quix' conceptual data lake architecture [JQ17].

*G) Metadata Management.* The metadata management aspect comprises two sub-aspects: The first one, *metadata as enabler*, overlaps the horizontal aspects in Fig. 2. In this sub-aspect, metadata enable other aspects. For example, metadata describe what zone a piece of data belongs to in a zone model or when data was created for lifecycle management. The other sub-aspect is *metadata as a feature*, depicted as the right side of the metadata management aspect in Fig. 2. This sub-aspect contains the functionalities that metadata can provide in addition to their enabler capabilities, such as business glossaries [Ba14] or semantical descriptions. For all metadata, aspects A-F have also to be defined, as metadata have to be stored, modeled, etc. Exemplary concepts are metadata management systems, such as *Constance* [HGQ16], or metadata models, such as *HANDLE* [Ei20].

*H) Data Security & Privacy.* Data security & privacy is a solely conceptual aspect. It is of great importance in a data lake, as it ensures legal conformance, alignment with business objectives, and much more [Ch15]. Exemplary concepts for this aspect are, e.g., *checking legal conformance* in the data wrangling process [Te15] or *AMNESIA* for GDPR-compliant machine learning [St20], where data security & privacy are implemented through zones.

*I) Data Quality.* The data quality aspect is also solely conceptual. Maintaining data quality is important to ensure the data's usability and prevent the data lake from turning into a data swamp [Ch14]. While there are *data quality tools*, e.g., Informatica data quality[7], these still rely on implementations of other aspects, e.g., metadata management. Data quality can also be found in, e.g., data organization, where some *zones hold trusted data* (e.g., [Gi20]).

### 3.2 Interdependencies between DLAF Aspects

The aspects of the DLAF are not independent from each other, as decisions for one aspect affect other aspects. For example, the aspect of data storage influences data modeling, as different kinds of storage systems support different kinds of modeling (e.g., relational modeling for relational databases, graphs for graph databases). This section explores the interdependencies between aspects and their implications. Fig. 3 depicts these interdependencies as a graph. They are in line with interdependencies as described in existing literature. In this graph, six of the overall nine aspects are visually grouped together for a simpler visualization. The influences between aspects depicted in this graph form the basis of the methodology presented in Sect. 4.

The graph shows that the aspects *data security & privacy* and *data quality* influence all other aspects of the DLAF (I1-I4). This is because these two aspects are not implemented directly, but instead implemented through other aspects. Thus, decisions made for data security & privacy and data quality have to be considered when choosing concepts for

[7] www.informatica.com/de/products/data-quality/informatica-data-quality.html

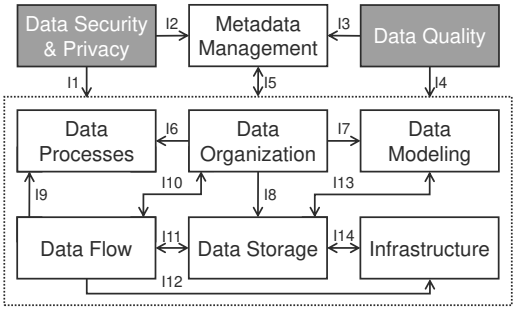


Fig. 3: The influence graph for the DLAF aspects. Directed arrows point from the influencing aspect to the influenced aspect.

all other aspects. For the aspect of *metadata management* all aspects produce and rely on metadata, e.g., metadata describing how data should be processed (I5). They influence what metadata should be collected and how it should be organized. At the same time, metadata are data and thus need to be considered when defining all other aspects (I5).

The *data organization* aspect influences data processes (I6), as data pipelining processes have to be adapted to the chosen data organization, e.g., data zones or ponds. Data organization also influences data modeling (I7), as e.g., data zone architectures typically dictate standardized data modeling in at least one zone. This means that all data in this zone are modeled according to specified rules, e.g., the rules of data vault. Furthermore, data organization influences data storage (I8), as, e.g., the data pond architecture, where data are separated by their structure, necessitates a different storage concept than, e.g., data zone architectures. The *data flow* aspect influences data processes (I9), data organization (I10), data storage (I11), and infrastructure (I12). This is because the data flow aspect comprises the different modes of data (batch and stream). Depending on the decisions made for this aspect, other aspects have to support the respective modes as well. For example, if the concept chosen for the data flow aspect includes stream processing, the concept chosen for infrastructure has to include stream processing engines. However, for data organization and data storage (I10, I11), the influence can also be reversed, as, e.g., using a batch-only data organization or choosing batch and stream processing in data storage dictates a certain data flow. The aspect of *data storage* influences and is influenced by both data modeling (I13) and infrastructure (I14). Depending on the types of storage systems chosen, certain data modeling techniques can or cannot be used. For example, if relational storage is chosen, data models should be applicable to relational schemata. At the same time, the choice of a certain data model will necessitate a different data storage concept. Data storage and infrastructure are closely connected, as one chooses the types of systems while the other defines the concrete systems and tools.

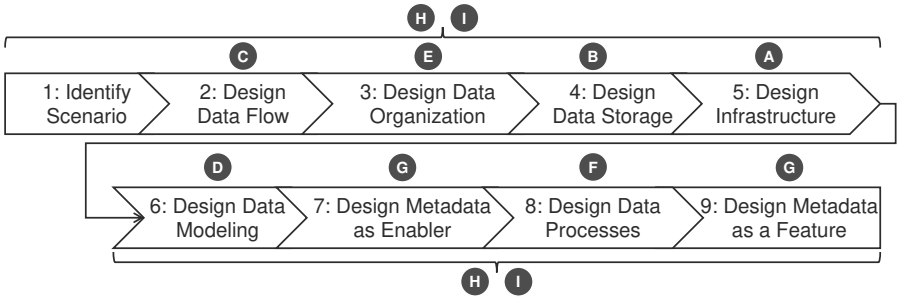


Fig. 4: To configure a comprehensive data lake architecture with the DLAF, nine steps are necessary. Each step is associated with different DLAF aspects, depicted in the circles.

## 4 Methodology for Configuring a Data Lake Architecture

To configure a comprehensive data lake architecture from the framework, specific concepts have to be chosen from the set of concepts associated with each aspect (see Fig. 2). This section provides a methodology for this task consisting of nine steps. Several of these steps directly correspond to aspects of the DLAF (see Fig. 4), and the order of the steps was determined from the aspects' interdependencies (see Fig. 3). As shown in Fig. 4, the aspects data security & privacy and data quality are associated with steps 1 through 9. This means that during all steps, these two aspects have to be taken into consideration and included accordingly. Our methodology provides guiding questions for each step that support the selection of appropriate concepts for architectural aspects. To the best of our knowledge, no further sources on such questions exists. In addition, we include typical concepts and associated decision guidelines for each aspect. However, these are not exhaustive due to the wide variety of available concepts for implementation.

*Step 1: Identify Scenario.* Understanding the key requirements of the data lake's application scenario is a crucial prerequisite for all following architecture decisions. To this end, the following questions should be answered: 1) What kind of data are managed in the data lake? What are their characteristics? This question also targets the data security & privacy and data quality requirements. 2) What time requirements are associated with data and their usage (e.g., real-time, hourly updates)? 3) How are data used (e.g., advanced analytics only or also reporting)? Further requirement elicitation should also be performed in this step.

*Step 2: Design Data Flow.* To determine a suitable data flow concept, especially question 2 from Step 1 (what time requirements are associated with data and their usage) is of relevance. If data should be both processed in real-time and in larger time intervals, both batch and stream processing should be used. Hybrid processing architectures comprise both, such as the Lambda architecture [MW15] or BRAID [Gi18]. A guiding question to this decision is: Are batch and stream processing independent from each other or should results from one be available to the other? If they are independent, the Lambda architecture is a sufficient concept. If data and results should be exchanged, BRAID offers the needed functionality.

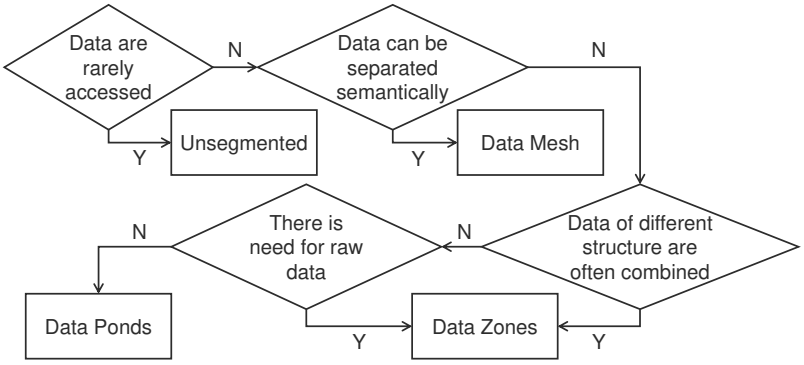


Fig. 5: The decision process for choosing an appropriate data organization concept. Combinations of concepts are not included due to space restrictions.

*Step 3: Design Data Organization.* Data organization focuses on the efficient management of data for different uses. Thus, to choose an appropriate concept for this aspect, question 3 from step 1 (how are data used) is of high importance again. Fig. 5 depicts a possible decision process for data organization, including some of the guiding questions to be asked. In our example, the concepts to choose from are no segmentation of the data, semantical data meshes [De19], Inmon's data pond architecture [In16], or data zone architectures (e.g., [Gi20]). A properly segmented and structured data lake provides more efficient access and usage than an unsegmented one, however, segmenting the data lake increases its complexity. Note there might be other suitable concepts for data organization, as well as combinations of these concepts. In addition, some of the concepts mentioned require further definition, as there are multiple variants of, e.g., the data zone architecture [Gi19a]. Concepts for data quality are included in most zone architectures and in the pond architecture. Data security & privacy however are only included in some zone architectures (e.g., [Gi20; Go16]) and not in the data pond architecture. For other data organization concepts, neither of the two aspects is considered. However, these aspects cannot be neglected.

*Step 4: Design Data Storage.* The configuration of a data storage concept depends on the kind of data to be managed (question 1 from step 1) and how they are used (question 3 from step 1). Exemplary guiding questions for this aspect are: Are multiple types of storage systems necessary? Which storage systems can support the data's characteristics? For example, relational databases provide the most appropriate support for structured data. If the managed kinds of data are widely varied, a combination of storage systems might be appropriate. When working with data ponds, each pond can be realized on a different system, e.g., an relational database for the analog data pond and a file system for the textual data pond. For this decision, exemplary guiding questions are: How are data used? What characteristics have to be supported? For example, highly connected data should be managed in a graph database, while structured data can be stored in relational databases. Further decision support can be found in e.g., [Ge17]. It is necessary to consider data security &

privacy and data quality requirements when defining the data storage aspect, as different types of data storage systems support different degrees of consistency, constraints, etc.

*Step 5: Design Infrastructure.* In this step, the defined data storage and data flow concepts are used to decide on an appropriate infrastructure for the data lake. Storage systems and data processing tools are constantly maturing, and the requirements towards infrastructure are manifold. We thus do not provide details on infrastructure decision support in this paper. However, some guiding questions are: What ingestion rates are required? Are indexes or foreign keys needed? What read/write performance should be offered? Infrastructure can then be chosen in accordance with the answers to these questions.

*Step 6: Design Data Modeling.* The answers given for question 1 and 3 from step 1 (what data are managed and how are they used) are of great importance for deciding on data modeling concepts, as they determine which data models are suitable. Exemplary guiding questions for this step are: How should structured and semi-structured data be modeled? For example, data vault [Li12] can be used to model these data in data lakes [Gi19b]. How can data be connected across systems? How can unstructured data be connected to structured data? Possible answers to these questions are data droplets [Ho17] or link-based integration [GSM14]. If zones are used as a concept for data organization, modeling concepts differ for each zone. Typically, one zone holds raw data replicated from the source, while another zone contains data in a standardized format, or even in a use-case specific format (e.g., as dimensional schema). Requirements towards data security & privacy and data quality have to be addressed, e.g., through separately treated tables for sensitive data or data models that consolidate data.

*Step 7: Design Metadata as Enabler.* The leading question to configure metadata as enabler is: What information is needed on the data to manage them meaningfully? This includes 1) metadata that are needed to reflect the concepts chosen in other aspects (e.g., metadata describing the zone of a data), and 2) metadata that are needed for the general operation of the data lake (e.g., information on lineage, access operations, or last-accessed dates). Some metadata are needed for the execution of data processes defined in step 8. Thus, it might be necessary to revisit this step during the definition of data processes to add additional metadata needed. Step 7 also includes metadata for data security & privacy and data quality, such as security classifications, a to-be-deleted date, or known quality issues. As the metadata as enabler identified in this step may vary greatly from one application scenario to another, it is impossible to provide a decisive guideline. Choosing a flexible metadata management model such as HANDLE [Ei20] is beneficial, as it can be adapted and even extended later on. If metadata management has not been considered as data in steps 1-6, step 7 is to fill these gaps. Metadata, just like other data, are in need of infrastructure, data storage, data flow, data modeling, and data organization concepts.

*Step 8: Design Data Processes.* Due to space restrictions, we cannot provide detailed guidelines for the data process configuration. Some guiding questions for this aspect are: How do data move in the data lake? How are they processed? What is the data's lifecycle?

Most data organization concepts include appropriate data process concepts, e.g., how data move and are processed between zones/ponds in data zone architectures (e.g., [Gi20]) and the data pond architecture [In16]. These processes have to be adapted and extended to fit the concepts chosen for the remaining aspects. If it turns out that data processes require further metadata, step 7 is revisited here. Data processes for data security & privacy, such as processes for accessing sensitive data, and data quality have to be chosen meaningfully to fit the application scenario's needs. The data wrangling process [Te15] and existing lifecycle management processes can serve as a base for the data process configuration.

*Step 9: Design Metadata as a Feature.* This final step includes all functionality that goes beyond the simple description of data. Metadata management systems such as data catalogs [Ch15] or data marketplaces [Mu13] offer functionalities that go beyond the scope of metadata as enabler, namely semantical data access or data purchase offers. As these additional functionalities can only be implemented with a detailed knowledge on the data lake's architecture, this step is done last. This part of the data lake can be designed quite freely. An associated guiding question is: What further benefit can metadata provide?

## 5 Assessment and Application of the DLAF

In this section, we assess the DLAF's suitability as architecture configuration guidance in two ways: 1) we analyze existing data lake implementations and sort their architectural decisions into the DLAF's aspects to demonstrate the framework's comprehensiveness (Sect. 5.1). The DLAF aids us in identifying shortcomings of existing architectures and provides pointers for improvement. 2) We assess the methodology's applicability by defining an exemplary data lake architecture using the DLAF (Sect. 5.2).

### 5.1 Comprehensiveness of DLAF

We use two real-world data lake implementations for the evaluation of the DLAF's comprehensiveness, in particular AIRPORTS DL [Ma17a] and the Smart Grid Big Data Ecosystem [MM18]. We chose these implementations because they provide detailed information on the concepts used and were evaluated using real-world data. They cover two different domains (air traffic, smart grids) and deal with different data management requirements. Neither of these papers includes a methodology for the configuration of their data lake. Tab. 1 matches the decisions made in these implementations with the DLAF aspects. Based on this categorization, we discuss the implementations' comprehensiveness and how they should be extended.

*AIRPORTS DL.* The AIRPORTS DL [Ma17a] focuses on storing surveillance data of flights, such as a plane's position or altitude. These data are combined with data from third parties, such as weather data, and are streamed into the data lake. The middle column in Tab. 1

| DLAF Layer | AIRPORTS DL [Ma17a] | Smart Grid Big Data Eco-System [MM18] |
|---|---|---|
| A. Infrastructure | Hadoop (HDFS, MapReduce), Apache Flume, Apache Spark, Apache Oozie, Apache Pig, Apache Atlas, R Studio, Shiny, Apache Sqoop | Hadoop (HDFS, MapReduce), Apache Flume, Apache Spark Streaming, Apache Spark SQL, Apache Hive, Apache Impala, Radoop, Matlab, Tableau, Google Cloud Computing |
| B. Data Storage | Single File System | Single Eco-System |
| C. Data Flow | Data are ingested as streams, but processed as batches | Based on the Lambda Architecture, data are processed as stream and as batches |
| D. Data Modeling | Raw Messages, AIRPORTS Data Model | Undefined |
| E. Data Organization | Four Zone Architecture | Two Zone Architecture for the data storage (Master data and Serving Layer) based on Lambda Architecture |
| F. Data Processes | Processing Pipeline for Messages (ETL Processes), Processes for Ingestion and Use | Processing Pipeline from the Lambda Architecture |
| G. Metadata Management | Managed by Apache Atlas | Undefined |
| H. Data Security & Privacy | Tracking manipulation of data | Undefined |
| I. Data Quality | Tracking manipulation of data, Quality through Zones | Undefined |

Tab. 1: Categorization of Architectural Decisions in Existing Data Lake Implementations

lists the architectural decisions made in this data lake implementation with respect to the aspects of the DLAF. The following paragraphs detail selected DLAF aspects. There is no explicit explanation for the *data flow* aspect in the paper, aside from data being ingested as data streams. However, data are stored before being processed. In addition, the tools used for processing (e.g., Hadoop MapReduce[8], Apache Pig[9]) are for batch processing. These decisions suggest that data are processed in batches only and not as data streams. For *data modeling*, data first are stored as raw key-value messages. Then, as they move through processing, they are transformed to fit the AIRPORTS data model, which was specifically created for this application scenario. For *data organization*, a zone architecture consisting of four zones, called layers in the paper, was chosen. These layers are 1) Raw Layer, 2) Alignment Layer, 3) Flight Leg Reconstruction Layer, and 4) Integration Layer. Data are ingested into the Raw Layer and then processed from layer to layer. Finally, data are made available in the Delivery Layer, which is not a processing layer, but an interface to the data lake. The *data processes* in this data lake implementation mostly focus on the movement of

[8] hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
[9] pig.apache.org/

data between zones. In addition, it is defined how data are ingested into the data lake and how they can be analyzed and delivered to external systems. Apache Atlas[10] is used to implement a governance system on top of the data lake, which provides *metadata management*, and basic *data security & privacy* and *data quality* by tracking data manipulation. The data quality aspect is also addressed by the zone architecture of the AIRPORTS DL, where the quality of data is increased from one zone to the other.

Overall, all architectural decisions of the AIRPORTS DL can be reflected by the aspects of the DLAF. It also shows that the AIRPORTS DL provided a concept for each of the DLAF aspects. Thus, the AIRPORTS DL architecture covers all aspects from infrastructure to data quality and thus underlines the comprehensiveness criteria.

*Smart Grid Big Data Eco-System.* The second data lake implementation analyzed is applied in a scenario of smart grids as part of a smart grid big data eco-system [MM18]. The data to be stored and analyzed in this scenario are highly diverse, including sensor data from, e.g., farms and consumers, but also images and videos from plant security cameras. These data are enriched with data from additional sources, e.g., weather data. Data are ingested into the data lake as a stream. The right column of Tab. 1 summarizes the architectural decisions in this data lake. The *data storage* of this data lake is realized as a single system, namely the Hadoop eco-system as seen in *infrastructure*, including HDFS, Hive[11], and Impala[12]. The *data flow* concept of this data lake is based on the lambda architecture [MW15]: Data ingested as data stream are forwarded to both batch processing, where they are stored persistently, and to stream processing, where they are processed in real-time. The results from both processing modes are combined in a Serving Layer. The usage of the lambda architecture influences the *data organization* concept. According to the lambda architecture, the data lake is divided into two zones: a Raw Zone, where data are persistently stored before processing, and the Serving Layer that holds the processing results. Similarly, *data processes* are given by the lambda architecture.

While all architectural decisions of this implementation could be assigned to a DLAF aspect, this analysis shows that the architecture of the smart grid data lake is not comprehensive. There are no concepts for *data modeling*, *metadata management*, *data security & privacy*, or *data quality*. However, without these concepts, a data lake risks turning into a data swamp, where data are unusable [Ch14]. Using the DLAF, this data lake can be re-designed including extensive metadata management to also address security and quality.

### 5.2 Application of the DLAF Methodology

In the second part of our assessment, we configure a data lake architecture using the DLAF and its methodology introduced in Fig. 2 and Fig. 4. In doing so, we evaluate the

[10] atlas.apache.org/
[11] hive.apache.org/
[12] impala.apache.org/

| Step | Resulting Decision |
|---|---|
| 1: Identify Scenario | Structured, semi-structured, and unstructured data |
| | Batch and stream processing |
| | Both advanced and traditional analyses |
| 2: Data Flow | Hybrid Processing Architecture BRAID |
| 3: Data Organization | Zone Reference Model |
| 4: Data Storage | Multi-Storage System |
| 5: Infrastructure | Hadoop (HDFS, MapReduce), Kafka, MySQL, Apache Spark, . . . , partially Cloud-based |
| 6: Data Modeling | Data Vault, Link-Based Integration |
| 7: Metadata as Enabler | Metadata Types based on HANDLE |
| 8: Data Processes | Organization Specific Processes |
| 9: Metadata as Feature | Data Catalog |

Tab. 2: Overview of Resulting Decisions in the Definition of an Exemplary Data Lake Architecture

applicability of the DLAF based on a real-world industry case from a large, globally active manufacturer. The manufacturer's business is highly diverse, with business domains ranging from manufacturing to quality management to finance. To increase efficiency and reduce costs, an enterprise-wide data lake is implemented to employ data analytics in everyday business. Tab. 2 provides an overview over the decisions made for the data lake architecture. The following paragraphs describe the decision process that led to these solutions. The data flow and data organization of the resulting data lake architecture is depicted in Fig. 6.

*Step 1: Identify Scenario.* In the use case introduced above, a wide variety of data are used for a multitude of different projects and analyses. To create a proper base for the definition of an exemplary data lake architecture, we refer to the questions defined in Sect. 4, Step 1. Based on these answers, the remaining steps are performed to configure a data lake architecture suited for the manufacturer's needs.

1) Throughout the entire business, data are collected from various source systems, such as enterprise resource planning systems and manufacturing execution systems, or the IoT. These data managed in the data lake are highly diverse, not only in structure (e.g., structured product data, semi-structured sensor data, and unstructured computer aided design (CAD) files), but also in their characteristics. These characteristics range from highly sensitive master data to voluminous IoT data of unknown quality.

2) Various time requirements exist in the data lake. In regular intervals, data are extracted from source systems and transferred to the data lake, where they are processed in periodic batches. Some of these data should be available within an hour, others need to be processed once a day or less. At the same time data from sensors arrives as data stream. These data should be processed immediately, to enable quick and timely reactions to, e.g., malfunctions in the manufacturing process. Also, they should be stored for later use as batch data. Thus, both batch and stream processing are of importance in this data lake.

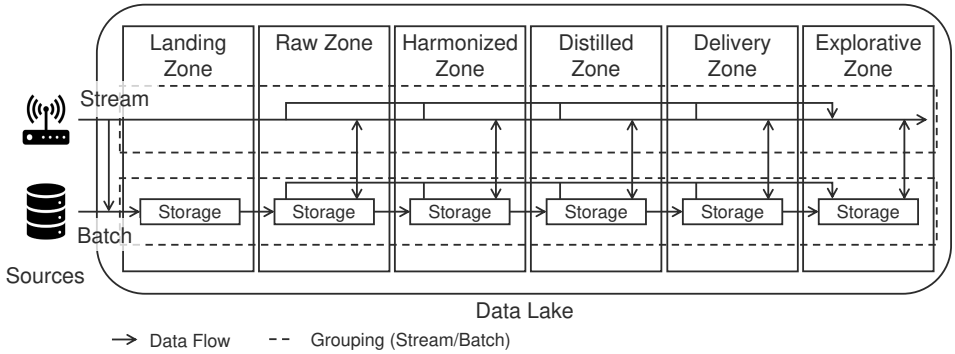


Fig. 6: An excerpt of the resulting data lake architecture, including data organization and data flow.

3) The data lake is the enterprise's central data repository that is accessed for a wide variety of use cases. These range from advanced analytics [Bo09], e.g., data mining and machine learning, to traditional reporting and online analytical processing (OLAP). For example, data in the data lake might be used to train a machine learning model to improve the quality of manufactured products, and to create a report for the supervisor of a specific plant.

*Step 2: Design Data Flow.* As specified in Step 1, the manufacturer relies on both batch and stream processing for the used data. Thus, the chosen data flow concept has to support both of these processing modes. For this exemplary data lake architecture, we decided to use the hybrid processing architecture BRAID [Gi18] as the data flow concept. In this architecture, data ingested as a stream are both forwarded to a persistent storage and to a stream processing engine. This behavior is similar to that of the Lambda Architecture as briefly described in Sect. 5.1. However, the BRAID architecture allows to use results from batch processing in stream processing. For example, a machine learning model can be trained on batch data and used to classify data from the data stream. In addition, results can be stored persistently and are available for later use. These characteristics of BRAID align with the manufacturer's requirements. Data ingested in batches are stored in the persistent storage and processed in batches. This data flow concept is depicted in Fig. 6.

*Step 3: Design Data Organization.* Because data are frequently accessed and used, a segmentation of the data lake into different portions is needed in this scenario. As data of different structure are often combined and the availability of raw data is crucial, we decided to use a data zone architecture. Due to the decisions made for the data flow aspect, this architecture needs to support both batch and stream processing. We chose the zone reference model [Gi20] as it provides fitting zones for the envisioned use cases for both batch and stream processing. It also contains concepts for both data quality and data security, e.g., the protected part, or varying access rights for different zones. Fig. 6 depicts the zone reference model and its interaction with batch data and streaming data.

| Zone | Characteristics that influence Data Modeling | Data Modeling Technique |
|---|---|---|
| Landing Zone | Temporary | Raw Format |
| Raw Zone | Large amounts of data | Raw Format |
| Harmonized Zone | Standardized Modeling Technique | Data Vault (Raw Vault), Link-based Integration |
| Distilled Zone | Standardized Modeling Technique, Use Case Dependance | Data Vault (Business Vault), Link-based Integration |
| Delivery Zone | Prepared for specific tools | Modeling according to needs |
| Explorative Zone | Modeling done by data scientist | Modeling according to needs |

Tab. 3: Overview of Data Modeling Decisions

*Step 4: Design Data Storage.* As the data to be managed is highly diverse, the data storage concept for this data lake comprises multiple different storage systems. That way, data can be managed where their characteristics and usage are supported best. For example, sensor data are stored in time series databases that support effective time-oriented queries (such as aggregations over time) while unstructured data are stored in a distributed file system.

*Step 5: Design Infrastructure.* As mentioned in step 4, multiple different storage systems and tools should be used. We chose various tools for storage and processing from the Hadoop ecosystem, e.g., HDFS and Apache Spark. Other systems, RDBMS and NoSQL databases alike, are added to this core to support more data characteristics. In addition, parts of the data lake are realized on a cloud-based structure to give third parties access to the stored data, such as suppliers or even end customers.

*Step 6: Design Data Modeling.* The usage of a data zone architecture for the data organization results in different modeling techniques in the zones. This is to support the required characteristics of the zones in the zone reference model. While data in the Landing Zone and Raw Zone are kept in their original format, Data Vault is used for structured data in the Harmonized Zone and the Distilled Zone. Data Vault allows flexible, use-case-independent, and scalable modeling of data in data lakes [Gi19b]. In addition, link-based integration [GSM14] is used to link structured and semi-structured data to unstructured data. The Harmonized Zone uses Raw Vault, while the Distilled Zone is modeled in Business Vault to include business logic. Finally, data in the Delivery Zone and the Explorative Zone are modeled according to specific needs.

*Step 7: Design Metadata as Enabler.* To handle all the stored data and to enable their usage, metadata management is needed. As metadata are also data, steps 1-6 have to be performed for them as well. Metadata may be structured or semi-structured and are ingested in the same way as the data it belongs to (e.g., as data stream for streaming data). The data flow concept for metadata thus is the same as for normal data. The data organization is unsegmented for metadata, as they span across the zones. For data storage, we decided to manage metadata in a graph database to support their highly connected structure (e.g., lineage metadata is

connected to data sources, operations, and resulting data). As infrastructure, we decided on Neo4J[13]. The metadata are modeled using HANDLE, which can represent, but is not limited to, lineage metadata, zone affiliations, and access information [Ei20]. This way, data security & privacy and data quality metadata can be stored, too.

*Step 8: Design Data Processes.* Data processes need to be specified in two sub-aspects, data lifecycle processes and data pipelining processes.

1) Data lifecycle processes in the scenario are defined in accordance with [DA17]. These processes manage data in all steps of the data lifecycle, ranging from creation over storage, use, and enhancement, to disposal. In all of these steps, metadata are captured and stored with the data, e.g., lineage metadata about data's creation, or metadata on who accessed data. Due to space reasons, we cannot discuss the aspect of lifecycle management in more detail. However, appropriate measures to comply with the data security & privacy and data quality concepts are taken, such as access control and change management.

2) Data pipelining processes are heavily intertwined with the data zone model used in data organization. Data are ingested and buffered in the Landing Zone before extract-transform-load (ETL) processes forward them to the Raw Zone. From there on, further ETL processes move the data into the other zones. These ETL processes apply transformations to the data to make them fit for the zone they are moving into. For example, data may be transformed according to data vault when moving from the Raw Zone to the Harmonized Zone. These processes are also responsible to realize the defined data security & privacy and data quality concepts. For example, personal data moving from the Landing Zone into the Raw Zone have to be anonymized. Similarly, data moving from the Raw Zone to the Harmonized Zone must follow certain quality guidelines.

*Step 9: Design Metadata as a Feature.* The final step is to specify concepts for metadata as a feature. In the scenario, we use three concepts that provide features in addition to those of metadata as enabler, namely a data catalog [Ch14] to allow access of data.

This completes the configuration of an exemplary data lake architecture using the DLAF. As can be seen in the description above, the usage of the DLAF and the associated methodology enabled a structured decision-making process. In the industry case, the DLAF provided guidance on what aspects to include and how to choose appropriate concepts for their implementation. It thus ensured that every aforementioned aspect is included in the data lake architecture and that their interdependencies are considered. For example, we could define data modeling with respect to the chosen zone model, or adjust the data processes to the chosen metadata management. The DLAF enabled interdisciplinary collaboration between domain experts, IT, and data scientists at the manufacturer's site by providing a common understanding of what a data lake architecture should comprise. Overall, the definition of this exemplary data lake architecture shows both the guidance DLAF provides as well as its applicability.

[13] neo4j.com/

## 6 Conclusion and Future Work

While various concepts refer to themselves as data lake architectures, none of them covers all aspects necessary for a functional data lake. Thus, we developed the data lake architecture framework (DLAF) to support the definition of a scenario-specific data lake architecture. The DLAF consists of nine data lake aspects to be considered, their interdependencies, and a methodology to choose appropriate concepts for each aspect. The evaluation showed that the DLAF can be applied in two ways: 1) It can be used to identify missing aspects in existing data lake implementations and provide pointers towards re-design of the architecture. Our discussion of existing real-world data lake architectures showed that important aspects had been forgotten during the architecture's definition, such as metadata management. We showed that the DLAF supports not only the evaluation of existing data lake architectures to identify such shortcomings, but also their extension towards comprehensiveness. 2) The DLAF can be used to define a novel comprehensive data lake architecture. We used it in a real-world industry case. The DLAF enables a structured, step-by-step decision process, while providing decision support for choosing appropriate concepts. As interdependencies between aspects are considered by the DLAF methodology, the concepts of the resulting data lake architecture are well-matched to each other.

For future work, we plan to further apply and evaluate the developed data lake architecture in practice. Furthermore, the implications of the DLAF for an enterprise-wide usage across multiple data lakes should be investigated.

## References

[Ba14] Ballard, C. et al.: Information Governance Principles and Practices for a Big Data Landscape. IBM, 2014.

[Bo09] Bose, R.: Advanced analytics: opportunities and challenges. Industrial Management & Data Systems (IDMS) 109/2, pp. 155–172, Mar. 2009.

[Ch14] Chessell, M. et al.: Governing and Managing Big Data for Analytics and Decision Makers. IBM, 2014.

[Ch15] Chessell, M. et al.: Designing and Operating a Data Reservoir. IBM, 2015.

[CY15] Casado, R.; Younas, M.: Emerging trends and technologies in big data processing. Concurrency and Computation: Practice and Experience 27/8, pp. 2078–2091, June 2015.

[DA17] DAMA: DAMA-DMBOK: Data Management Body of Knowledge. Technics Publications, 2017.

[De19] Dehghani, Z.: How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh, 2019, visited on: 05/27/2019.

[Ei20] Eichler, R. et al.: HANDLE - A Generic Metadata Model for Data Lakes. In: Proceedings of the 22nd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK2020). 2020.

[Ge17] Gessert, F. et al.: NoSQL database systems: a survey and decision guidance. Computer Science - Research and Development 32/3-4, pp. 353–365, July 2017.

[Gi18] Giebler, C. et al.: BRAID - A Hybrid Processing Architecture for Big Data. In: Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018). SCITEPRESS - Science and Technology Publications, pp. 294–301, 2018.

[Gi19a] Giebler, C. et al.: Leveraging the Data Lake - Current State and Challenges. In: Proceedings of the 21st International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2019). 2019.

[Gi19b] Giebler, C. et al.: Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned. In: Proceedings of the 38th Conference on Conceptual Modeling (ER 2019). 2019.

[Gi20] Giebler, C. et al.: A Zone Reference Model for Enterprise-Grade Data Lake Management. In: Proceedings of the 24th IEEE Enterprise Computing Conference (EDOC 2020). 2020.

[Go16] Gorelik, A.: The Enterprise Big Data Lake. O'Reilly Media, Inc., 2016.

[GSM14] Gröger, C.; Schwarz, H.; Mitschang, B.: The Deep Data Warehouse: Link-Based Integration and Enrichment of Warehouse Data and Unstructured Content. In: Proceedings of the 2014 IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC 2014). IEEE, pp. 210–217, Sept. 2014.

[HGQ16] Hai, R.; Geisler, S.; Quix, C.: Constance: An Intelligent Data Lake System. In: Proceedings of the 2016 International Conference on Management of Data (SIGMOD'16). Pp. 2097–2100, 2016.

[Ho17] Houle, P.: Data Lakes, Data Ponds, and Data Droplets, Online, 2017.

[In16] Inmon, B.: Data Lake Architecture - Designing the Data Lake and avoiding the Garbage Dump. Technics Publications, 2016.

[JQ17] Jarke, M.; Quix, C.: On Warehouses, Lakes, and Spaces: The Changing Role of Conceptual Modeling for Data Integration. In (Cabot, J. et al., eds.): Conceptual Modeling Perspectives. Springer International Publishing AG, chap. 16, pp. 231–245, 2017.

[Li12] Linstedt, D.: Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. 2012.

[Lo16] Lock, M.: Maximizing your Data Lake with a Cloud or Hybrid Approach, tech. rep., 2016.

[Ma17a] Martínez-Prieto, M. A. et al.: Integrating Flight-related Information into a (Big) data lake. In: Proceedings of the 36th IEEE/AIAA Digital Avionics Systems Conference (DASC). IEEE, 2017.

[Ma17b] Mathis, C.: Data Lakes. Datenbank-Spektrum 17/3, pp. 289–293, Nov. 2017.

[ML16] Madera, C.; Laurent, A.: The Next Information Architecture Evolution: The Data Lake Wave. In: Proceedings of the 8th International Conference on Management of Digital EcoSystems (MEDES). ACM Press, New York, New York, USA, pp. 174–180, 2016.

[MM18] Munshi, A. A.; Mohamed, Y. A.-R. I.: Data Lake Lambda Architecture for Smart Grids Big Data Analytics. IEEE Access 6/, pp. 40463–40471, 2018.

[Mu13] Muschalle, A. et al.: Pricing Approaches for Data Markets. In: International Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE 2012). Pp. 129–144, 2013.

[MW15] Marz, N.; Warren, J.: Big Data - Principles and best practices of scalable real-time data systems. Manning Publications Co., 2015.

[NRD18] Nogueira, I.; Romdhane, M.; Darmont, J.: Modeling Data Lake Metadata with a Data Vault. In: Proceedings of the 22nd International Database Engineering Applications Symposium (IDEAS 2018). 2018.

[RZ19] Ravat, F.; Zhao, Y.: Data Lakes: Trends and Perspectives. In: Proceedings of the 30th International Conference on Database and Expert Systems Applications (DEXA 2019). Pp. 304–313, 2019.

[SD20] Sawadogo, P.; Darmont, J.: On data lake architectures and metadata management. Journal of Intelligent Information Systems/, 2020.

[Sh18] Sharma, B.: Architecting Data Lakes - Data Management Architectures for Advanced Business Use Cases. O'Reilly Media, Inc., 2018.

[St20] Stach, C. et al.: AMNESIA: A Technical Solution towards GDPR-compliant Machine Learning. In: Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020). Pp. 21–32, 2020.

[Te15] Terrizzano, I. et al.: Data Wrangling: The Challenging Journey from the Wild to the Lake. In: Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR'15). 2015.

[Za87] Zachman, J. A.: A framework for information systems architecture. IBM Systems Journal 26/3, pp. 276–292, 1987.

[Zi15] Zikopoulos, P. et al.: Big Data Beyond the Hype. McGraw-Hill Education, 2015, ISBN: 978-0-07-184466-6.