Assessing the Lakehouse: Analysis, Requirements and Definition

Jan Schneider[1], Christoph Gröger[2], Arnold Lutsch[2], Holger Schwarz[1] and Bernhard Mitschang[1]
[1]Institute of Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
[2]Robert Bosch GmbH, Borsigstraße 4, 70469 Stuttgart, Germany

Keywords: Lakehouse, Data Warehouse, Data Lake, Data Management, Data Analytics.

Abstract: The digital transformation opens new opportunities for enterprises to optimize their business processes by applying data-driven analysis techniques. For storing and organizing the required huge amounts of data, different types of data platforms have been employed in the past, with data warehouses and data lakes being the most prominent ones. Since they possess rather contrary characteristics and address different types of analytics, companies typically utilize both of them, leading to complex architectures with replicated data and slow analytical processes. To counter these issues, vendors have recently been making efforts to break the boundaries and to combine features of both worlds into integrated data platforms. Such systems are commonly called *lakehouses* and promise to simplify enterprise analytics architectures by serving all kinds of analytical workloads from a single platform. However, it remains unclear how lakehouses can be characterized, since existing definitions focus almost arbitrarily on individual architectural or functional aspects and are often driven by marketing. In this paper, we assess prevalent definitions for lakehouses and finally propose a new definition, from which several technical requirements for lakehouses are derived. We apply these requirements to several popular data management tools, such as Delta Lake, Snowflake and Dremio in order to evaluate whether they enable the construction of lakehouses.

1 INTRODUCTION

In recent years, enterprises of almost all sectors have become subject to fundamental paradigm shifts: Large-scale projects, such as in the scope of Industry 4.0 (Lasi et al., 2014), are driving the digital transformation and pursue to interleave traditional business models with digital technologies. Supported by the recent advances and the increasing maturity of AI (Davenport and Ronanki, 2018), data-driven analysis techniques can now be utilized to evaluate existing business processes, products and services by deriving insights and knowledge from collected data. This development opens new opportunities for companies to evaluate and optimize their business practices and hence gain long-term competitive advantages. For example, in manufacturing, data collected along the value chain can be used to optimize product lifecycles, taking all stages from the product development until the retirement into account. To keep up with the advances in this field and to benefit from them, enterprises need to a) collect related data, b) store and organize the resulting huge amounts of data in a structured manner and c) exploit the data by applying data-driven analysis techniques. In this context, *data platforms* take a crucial role: They allow to store data and associated metadata from all kinds of sources and hence form the technical foundation for data collection, data processing and analytics applications. While the field of data platforms has been dominated by data warehouses (Inmon W. H., 2005) and data lakes (Giebler et al., 2019) in the past, a supposedly new type has recently attracted attention: So-called *lakehouses* claim to combine the desirable characteristics of data warehouses and data lakes, allowing to serve all kinds of analytical workloads from a single platform (Armbrust et al., 2021). This development promises huge improvements regarding operational costs and the quality of analysis results, since conventional enterprise data architectures are currently rather complex and require a) the utilization of several types of data platforms in parallel to serve all kinds of workloads, b) the storage of multiple copies of the same data on different platforms, as well as c) the implementation of error-prone and often slow data pipelines for synchronizing the data between the platforms, leading to stale or inconsistent data.

Schneider, J., Gröger, C., Lutsch, A., Schwarz, H. and Mitschang, B.
Assessing the Lakehouse: Analysis, Requirements and Definition.
DOI: 10.5220/0011840500003467
In *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) - Volume 1*, pages 44-56
ISBN: 978-989-758-648-4; ISSN: 2184-4992

The prospect of addressing these problems resulted in high expectations: According to the Gartner Hype Cycle for Data Management (Feinberg et al., 2022), the lakehouse vision is about to meet the peak of expectations and will reach maturity in two to five years. Consequently, many vendors of data management tools try to take advantage of this trend and expand their products for common features of data warehouses and data lakes. Since precise definitions and distinguishing criteria are missing, it remains unclear how lakehouses can be characterized, which requirements they must necessarily fulfil and which data management tools enable the construction of lakehouses. The broad usage of "lakehouse" as a marketing term further blurs the boundaries.

In the paper at hand, we address these issues by reviewing prevalent literature and definitions for lakehouses and with the following key contributions:

- We propose a new definition that overcomes the identified issues of existing definitions,
- based on this definition, we derive eight technical requirements for lakehouses, and
- we evaluate popular data management tools, such as Delta Lake, Snowflake and Dremio by applying the derived requirements to assess whether these tools already enable the construction of full-fledged lakehouses.

The remainder of this paper first provides background information regarding the role of data warehouses and data lakes in enterprise analytics architectures. Section 3 then reviews available literature, leading to the proposal of our definition and the derivation of technical requirements in Section 4. These are subsequently applied to six popular data management tools in Section 5. Finally, conclusions regarding the investigated types of data management tools are drawn.

2 BACKGROUND

This section provides an overview on data warehouses and data lakes and discusses how they can be combined in enterprise analytics architectures.

2.1 Data Warehouses and Data Lakes

Data platforms form the technical foundation for data collection, data processing and analytics applications (Gröger, 2022). Table 1 summarizes key properties of common data warehouses and data lakes, the two most prominent kinds of analytical data platforms.

Emerged from relational databases as a more convenient solution for large-scale data analysis, data warehouses represent the more established type. They typically allow multidimensional data modelling and querying, guarantee ACID properties (Härder and Reuter, 1983) and provide advanced management capabilities, such as for data governance, time travel and zero-copy cloning (Armbrust et al., 2021). Modern data warehouses transfer these concepts to public clouds and thus provide high scalability and reduced operational costs. Due to their static, use-case specific data models, data warehouses are primarily used to answer questions that are known in advance, such as in reporting and online analytical processing (OLAP) workloads (Chaudhuri and Dayal, 1997), and barely for any kinds of advanced analytics (Bose, 2009).

These limitations gave birth to the idea of *data lakes*, which leverage highly scalable and low-cost storage systems, such as the Hadoop Distributed File System (HDFS) or cloud services, to store all kinds of data in their raw formats as self-contained files or objects. For this purpose, open file formats like Apache Parquet[1] are commonly utilized, which enable direct data access for applications through the interfaces of the underlying storage layer. Due to these characteristics, data lakes provide more flexibility for analyses than data warehouses, but at the cost of low robustness and a lack of management features. Furthermore, the business value of the stored data can only be exploited when extensive management of metadata is performed (Eichler et al., 2021).

Table 1: Comparison of data warehouses and data lakes.

Property	**Data Warehouse**	**Data Lake**
Workloads:	Reporting, OLAP	Advanced analytics
Users:	Business users, data analysts	Data scientists
Data access:	Query language, data export	Direct access on storage
Data independence:	Physical, partly logical	Weak
Guarantees:	ACID	Weak
Schema:	On-write	On-read
Data type:	Mainly structured	All types
Addressing:	Relational	Via metadata
Data granularity:	Aggregated	Raw and aggregated
Data Storage:	RDBMS	Object storage
Flexibility:	Low	High
Mgt. features:	Advanced	Rudimentary
Analysis questions:	Known in advance	Not known in advance

[1] https://parquet.apache.org

In summary, it can be stated that both types of data platforms show rather contrary properties and hence target different fields of analytical applications.

2.2 Integration Patterns

While data warehouses and data lakes address different analytical workloads, companies often need to leverage both types in parallel (Gröger, 2021), e.g. for generating business reports, feeding user recommendation systems and for the training and application of machine learning models. In industrial practice, we currently see four common patterns for integrating the capabilities of data warehouses and data lakes into enterprise analytics architectures, which are depicted in Figure 1. In addition to these, also variants exist.

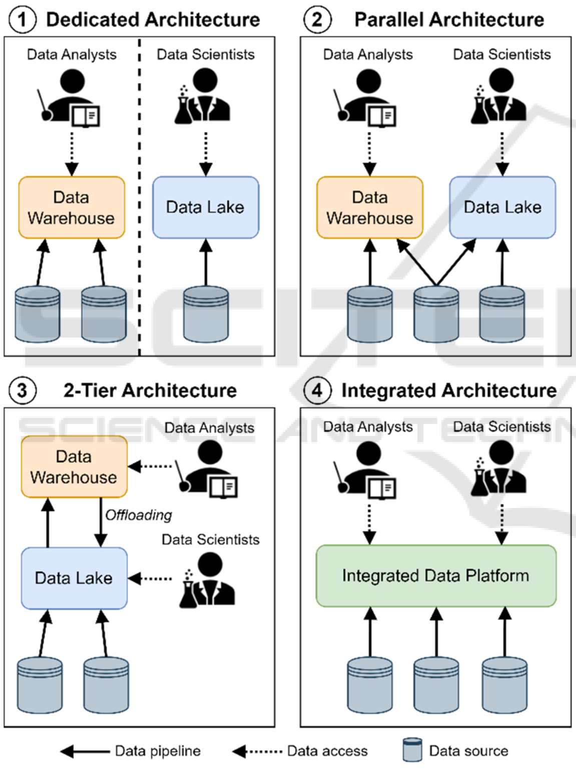

Figure 1: Integration patterns for combining data warehouses and data lakes in enterprise analytics architectures.

In ①, a data warehouse and a data lake are used independently. This pattern assumes that the data of each data source is of interest for either reporting and OLAP or advanced analytics, but not for both. Consequently, each data record is only ingested into one of both data platforms, depending on its relevance for the different types of analytics. The pattern stands out for its simplicity, but is strictly limited to scenarios where the source data can be appropriately split into disjoint subsets for both workloads, which is rarely the case. Furthermore, it is inflexible, because the decisions on how to split the source data require upfront assumptions regarding the analysis questions.

Similarly, ② also employs an independent data warehouse and data lake; however, the source data is not split up anymore and instead ingested into both data platforms where necessary. This way, relevant data can be exploited for reporting, OLAP and advanced analytics in parallel, while other data can solely reside on one of the platforms, depending on the intended analyses. This approach is more flexible than ①, but also requires the replication of data to both platforms, which prevents the formation of a single source of truth, provokes additional storage costs, may cause inconsistencies between both copies of data and requires several pipelines for data ingestion.

The 2-tier architecture in ③ appears to be the currently most commonly used integration pattern (Armbrust et al., 2021). Here, all source data is first ingested into the data lake and subsequently prepared for analytical evaluation. A second data pipeline then copies or moves data from the data lake to the data warehouse, where it can be exploited in the scope of reporting and OLAP workloads. Optionally, another pipeline can offload data that is no longer required by the data warehouse back to the data lake to improve query performance and storage costs (Oreščanin and Hlupić, 2021). However, this pattern possesses severe drawbacks (Armbrust et al., 2021): The additional data pipelines increase the overall complexity of the architecture and the required data conversions render them error-prone. In addition, they cause additional delays, leading to stale data in the data warehouse.

Finally, ④ represents the vision of a single data platform that combines desirable characteristics and features of both worlds such that all types of analytical workloads can be served. This way, no data replication or additional pipelines for transferring the data between platforms are needed. How such a solution may look like and which requirements it must necessarily meet is discussed in the following sections.

3 RELATED WORK

First, work related to the general concepts of lakehouses, associated technologies, implementations and practical applications is discussed. The second part of this section then elaborates on existing definitions for lakehouses and shows why they are insufficient.

3.1 Conceptual Work

The term "lakehouse" was presumably used for the first time by Alonso (Alonso, 2016), where it describes *“a solution for the analytical framework in the middle point [...] between classical [data warehouses] and [data lakes]”* that allows to combine the schema-on-write and schema-on-read paradigms by using flexible schemas. However, the work does not discuss how other properties of data warehouses and data lakes (cf. Table 1) could be combined as well and hence positions itself considerably far from today's notion of a lakehouse. About four years later, the lakehouse idea took shape with the emergence of the open source framework Delta Lake[2], which intends to allow the construction of integrated data platforms that combine characteristics of modern data lakes with comfortable management features of traditional data warehouses (cf. ④ in Figure 1). The underlying concepts and technologies are explained in the accompanying paper by Databricks (Armbrust et al., 2020), which refers to Delta Lake as a novel kind of *“ACID table storage layer over cloud object stores”*. The term “lakehouse” gained further popularity with their subsequent paper (Armbrust et al., 2021), which discusses issues of typical enterprise analytics architectures and emphasizes the benefits of an integrated lakehouse platform in comparison to the established 2-tier architecture (cf. ③ in Figure 1). The paper focuses on Delta Lake, but also refers to similar frameworks, such as Apache Hudi[3] and Apache Iceberg[4].

Most of the currently available literature on lakehouses has adopted the descriptions and elementary concepts presented in the two previously mentioned papers, promoting them to cornerstones for research related to the lakehouse vision. However, also other perspectives exist: Oreščanin and Hlupić (Oreščanin and Hlupić, 2021; Hlupić et al., 2022) use the term “lakehouse” to describe an architecture similar to the 2-tier approach, in which data is transferred between a data lake and a data warehouse and propose the integration of a virtualization layer that provides uniform data access to the users. According to Azeroual et al. (Azeroual et al., 2022), lakehouse-like characteristics can be achieved by combining data lakes with practices of data wrangling. Others argue that modern, cloud-based data warehouses already represent feature-rich lakehouses, since they have adopted common features of data lakes, including the ingestion of streaming data, support for semi-structured data and means for querying data on external cloud storages (Hansen, 2021; Eckerson, 2020). In contrast, Inmon et al. (Inmon et al., 2021) argue that lakehouses are always built on top of existing data lakes.

Due to the different views, Raina and Krishnamurthy (Raina and Krishnamurthy, 2022) conclude that the term “lakehouse” should only be used to describe the general vision of combining both worlds, rather than to categorize individual tools. However, we believe that lakehouses add value over prevalent enterprise analytics architectures and indeed possess characteristics that distinguish them from traditional data warehouse or data lake solutions (cf. Section 4).

For the construction of lakehouses, Behm at al. (Behm et al., 2022) propose a vectorized query engine for the Databricks ecosystem that provides increased performance for SQL queries on tables in open file formats. Fourny et al. (Fourny et al., 2021) developed a language and library which allows to define tasks for data preparation and the management of machine learning models on lakehouses in a declarative manner. Oreščanin and Hlupić (Oreščanin and Hlupić, 2021) suggest to leverage process control frameworks for orchestrating data flows in lakehouses.

Current proposals for the implementation of lakehouses are often based on Delta Lake and the Databricks ecosystem, like the one used by Begoli et al. (Begoli et al., 2021) for the management of biomedical research data, or on public cloud services (L'Esteve, 2022; Shiyal, 2021). In contrast, Tovarňák et al. (Tovarňák et al., 2021) utilize a more diverse technology stack, including Apache Iceberg, Apache Spark[5], Trino[6] and other tools for telemetry analysis.

Due to its popularity and the broad variety of perspectives, technologies and implementations related to the lakehouse vision, a precise characterization is necessary. However, the existing definition attempts are not sufficient, as pointed out in the following.

3.2 Prevalent Lakehouse Definitions

An obvious definition for lakehouses can be derived from the portmanteau word "lakehouse" itself: It suggests the fusion of key characteristics of data warehouses and data lakes, some of which are listed in Table 1, into a common architecture (cf. Shiyal, 2021; Raina and Krishnamurthy, 2022; Alonso, 2016; Eckerson, 2020; Hansen, 2021)). However, it remains unclear which properties must necessarily be present

[2] https://delta.io

[3] https://hudi.apache.org

[4] https://iceberg.apache.org

[5] https://spark.apache.org

[6] https://trino.io

and/or whether this architecture needs to reflect a single data platform or can also be implemented as several tiers (cf. ③ in Figure 1).

The most often cited definition is given by Armbrust et al. (Armbrust et al., 2021), who define a lakehouse as *"data management system based on lowcost and directly-accessible storage that also provides traditional analytical DBMS management and performance features such as ACID transactions, data versioning, auditing, indexing, caching, and query optimization."* The first two characteristics, low-cost storage and direct data access, refer to attributes of data lakes and have a mandatory character in this definition. In contrast, the second part only provides examples for common management and performance features of data warehouses that are also desirable for lakehouses, but does not actually demand any of them. Depending on the interpretation of this definition, a) platforms based on the Delta Lake framework, b) instances of Apache Hive[7] on top of the HDFS, c) a cloud object storage combined with an external SQL query engine and even d) modern data warehouses that support tables on external storages could be considered lakehouses. However, all of these approaches show rather different qualities, e.g. with respect to read and write access, provided guarantees and stream processing capabilities. Furthermore, this definition does not explain why the listed properties were selected and how they can achieve benefits over prevalent enterprise analytics architectures.

According to Gartner (Feinberg et al., 2022), a lakehouse represents a *"converged infrastructure environment that combines the semantic flexibility of a data lake with the production optimization and delivery of a data warehouse"* and *"supports the full progression of data from raw, unrefined [to] optimized data for consumption."* This definition reflects a business strategy perspective rather than a technical one. "Semantic flexibility" and "production optimization and delivery" are rather abstract properties that do not provide a sharp outline of the lakehouse paradigm and are difficult to verify in practice. The second part of the definition refers to the so-called *Delta architecture* (Leano, 2020), which is claimed to represent an alternative to the Lambda (Warren and Marz, 2015) and Kappa (Kreps, 2014) architectures by unifying batch and stream processing. While we consider this an important implication of lakehouses (cf. Section 4.3.8), Gartner remains too abstract and merely describes a process that can already be implemented on conventional data lakes by leveraging processing engines and zone models (Giebler et al., 2020).

[7] https://hive.apache.org

Hansen (Hansen, 2021) defines a lakehouse as "*architectural approach for managing all [types of data] and supporting [all] data workloads (Data Warehouse, BI, AI/ML, and Streaming)*", which emphasizes the intended usage of lakehouses rather than detailed functional characteristics. However, the definition is too broad to delineate a distinct concept, as all of the integration patterns (cf. Figure 1) can be considered "architectural approaches" that satisfy the two prerequisites mentioned in this definition.

Similar to Hansen, our definition (cf. Section 4.1) also focuses on the analytical workloads lakehouses must be able to serve, but adds further constraints that reflect the promised benefits in comparison to currently operated enterprise analytics architectures.

4 DEFINITION AND REQUIREMENTS FOR LAKEHOUSES

This section proposes a definition for lakehouses that addresses the issues of prevalent definitions as discussed in Section 3. Next, several technical requirements are derived that allow to verify whether given data platforms represent full-fledged lakehouses.

4.1 Defining the Lakehouse

As shown in Table 1, most of the characteristics of typical data warehouses and data lakes are rather contrary, for example with respect to data access, data independence, and the storage type. For this reason, a straight-forward merge of both concepts into one universal data platform that preserves all desirable properties is not possible. Instead, data warehouses and data lakes typically need to give up on some of their characteristics in order to be able to adopt features from the respective other platform. For instance, a data warehouse that should support direct data access on the storage layer like data lakes, must give up its data independence and instead utilize open file formats. If it should support semi-structured data as well, it must relax its relational design and consistency guarantees. Similarly, a data lake that is supposed to provide ACID properties has to limit its support for direct data access, since read and write operations must now be performed according to a specific protocol that ensures data integrity.

On the other side, the combination of both types of data platforms can also lead to new, emergent characteristics that neither typical data warehouses nor data lakes possess. In summary, they represent the additional value of lakehouses for enterprises in comparison to the other patterns sketched in Figure 1 and may include the ability to satisfy all analytical workloads from a single data platform and reduced maintenance efforts. However, drawbacks may emerge as well, such as an increased risk of vendor lock-ins. Figure 2 illustrates how the characteristics of lakehouses can be composed. In this figure, defining lakehouses means to decide which mandatory characteristics the green set must include. While prevalent definitions do not select these in a structured manner (cf. Section 3.2), we first shift the focus to a higher level of abstraction and instead of individual characteristics consider the different kinds of analytical workloads that are expected to be executed on lakehouses. Secondly, since each type of workload is associated with functional requirements, we use the workloads to derive mandatory characteristics in a top-down manner.

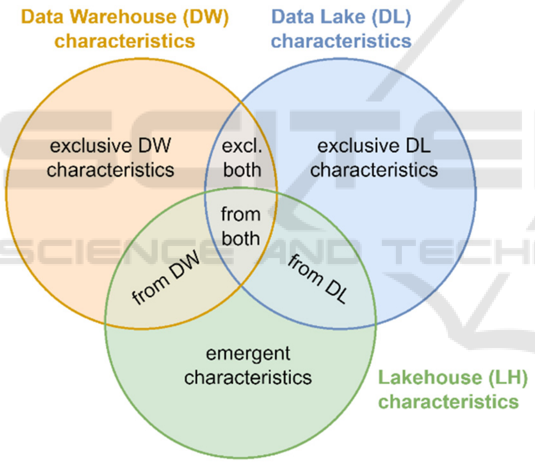
Data Warehouse (DW) characteristics
Data Lake (DL) characteristics
exclusive DW characteristics
excl. both
exclusive DL characteristics
from both
from DW
from DL
emergent characteristics
Lakehouse (LH) characteristics

Figure 2: Venn diagram illustrating how the characteristics of lakehouses are composed.

Despite the different perspectives on the lakehouse paradigm, it appears to be common sense that the fundamental motivation is to simplify existing enterprise analytics architectures and to reduce their complexity. We believe that this vision can only be achieved when a lakehouse consists of a single, integrated platform that can run the typical workloads of both data warehouses and data lakes, since architectures with multiple data platforms a) prevent the formation of a single source of truth, b) require additional data pipelines that need to be maintained, c) require additional data transformations that may cause inconsistencies and d) tend generally to become complex and error-prone, which undermines the promises of the lakehouse paradigm. Consequently, we argue that the 2-tier architecture (cf. ③ in Figure 1) does not represent a lakehouse. Based on these considerations, we propose the following definition:

Definition 1. *A lakehouse is an integrated data platform that leverages the same storage type and data format for reporting and OLAP, data mining and machine learning, as well as streaming workloads.*

Reporting and OLAP refer to the primary workload of data warehouses, while the combination of data mining and machine learning, as well as streaming represent typical data lake workloads. All three workloads are discussed in detail in Section 4.2 and are subsequently used to derive technical requirements. With its additional constraints, the definition ensures that lakehouses use the same type of storage (e.g. object storages) and the same data format (e.g. Parquet) to serve all of the listed workloads. As a consequence, the data must not be replicated to different types of storages or transformed into other formats for these purposes. Section 4.3.1 explains this in more detail.

The definition deliberately does not make any statement about non-functional properties of lakehouses, since these mainly distinguish more suitable from less suitable lakehouse systems for the respective application scenario, but have no major influence on the underlying type of data platform.

4.2 Analytical Workload Characteristics

This section characterizes the three analytical workloads that a lakehouse must be able to serve according to our definition. Table 2 summarizes the results.

4.2.1 Reporting and OLAP

Reporting refers to the production, delivery and management of reports (Vaisman and Zimányi, 2022), i.e. static or interactive overviews of business facts, such as key performance indicators (KPIs) and corresponding visualizations (Zheng, 2017). For the automatic generation of reports, predefined queries are typically employed and periodically executed against the stored data (Vaisman and Zimányi, 2022).

This workload is supplemented by OLAP, which intends to enable interactive analyses by providing fast, intuitive, multi-user and scalable query capabilities based on multidimensional data models (Pendse and Creeth, 1995). Together, reporting and OLAP allow to extract descriptive statistics from the stored data in order to support business decisions. For example, the periodic calculation of the first pass yield within a manufacturing company, broken down to the level of machines and quarters, can guide decisions regarding the acquisition and replacement of manufacturing machines. To enable efficient query processing and report generation, the data must be available in structured and table-like form, which requires the definition and enforcement of schemas to ensure data integrity and quality. While low response times are desirable for OLAP, batch-like processing is generally sufficient, since the performed analyses are typically not time-critical and do not need to happen immediately in response to external events.

4.2.2 Data Mining and Machine Learning

Both data mining and machine learning are broad sub-disciplines of the field of advanced analytics (Bose, 2009). Data mining is the process of discovering patterns and other forms of knowledge in large data sets (Han et al., 2022). Typical data mining techniques include classification approaches, clustering analysis, association analysis and regression analysis. The goal of machine learning is to develop learning algorithms that are capable of building models from data, which can then be used to generate predictions on new observations (Zhou, 2021). Although there is an overlap between the techniques used in data mining and machine learning, the focus of data mining lies on finding new patterns and inferring knowledge, while machine learning tries to generalize from patterns in collected data in order to generate prediction models for unseen data. For example, in the context of manufacturing, data mining techniques can be applied to find usage patterns for products and derive possible design optimizations from them, whereas machine learning may allow to create models for the predictive maintenance of machines. Since most of the associated techniques and algorithms are too complex to be expressed using query languages and due to the volume of data that needs to be analysed, data mining and machine learning usually require direct read access to the data on the storage layer. This also provides high flexibility for data mining, as analysis questions are often not known in advance and only arise after the discovery of first patterns in the collected data. The data that is supposed to be analysed can be of arbitrary types, including semi-structured and even unstructured data (Gröger et al., 2014). This workload has no strict timing requirements, because data mining and the training of models are rather slow processes that involve human experimentation and produce results that are valid until a further iteration comes up with an updated version of the model.

4.2.3 Streaming

In the context of analytical workloads, *streaming* subsumes all analysis techniques for near-real-time reporting and stream analytics (Kejariwal et al., 2017). The goals of near-real-time reporting are similar to those of batch reporting (cf. Section 4.2.1), with the difference that the reports are usually replaced by dashboards whose business facts and visualizations must be updated within minutes. Due to the time-consuming nature of most data-driven analysis techniques, results cannot be continuously re-calculated and instead must be incrementally updated as new data arrives. Hence, near-real-time reporting requires different approaches than batch reporting.

Stream analytics refers to techniques for the analysis of data that arrives in continuous data streams, including algorithms for data filtering, pattern detection and clustering (Kejariwal et al., 2017). Data for streaming workloads is typically either structured or semi-structured, since most streaming tools cannot handle unstructured data well. In the scope of a manufacturing company, the application of machine learning models to arriving data for predictive maintenance and the updating of dashboards for shop floor operators are typical examples of streaming workloads. Traditional data warehouses are designed for batch processing and operations on large data volumes and hence not optimized for small incremental data changes that occur with high frequency, which renders them unsuitable for streaming. Instead, streaming workloads have been mainly executed on data lakes so far, e.g. by applying the Lambda or Kappa architecture (Giebler et al., 2021).

Table 2: Comparison of the analytical lakehouse workloads.

Characteristics	**Reporing/ OLAP**	**DM/ML**	**Streaming**
Analytics types:	Descript., diagnostic	Diagnostic, predictive, prescriptive	Descriptive, diagnostic, predictive
Users:	Business users, data analysts	Data scientists	Operators, analysts
Data access:	Via query language	Direct access on storage	Direct acc. on stream storage
Timing:	Batch	Batch	Near-real-time
Data types:	Structured	All types	Structured, semi-struct.
User concurrency:	High	Low	Low

4.3 Derived Technical Requirements

Based on the previously described analytical workloads, we identified eight technical requirements that a lakehouse must satisfy in order to comply with our lakehouse definition. Table 3 provides an overview over them and indicates how strongly they were influenced by the different workloads.

4.3.1 R1: Same Storage Type and Data Format

Following up on our lakehouse definition, this requirement demands that all data and metadata is solely stored on a single type of highly scalable storage and that all data (excluding metadata) is stored using the same data format. Examples of storage types include object storages, such as provided by Amazon S3[8] or Azure Blob Storage[9], and highly scalable file systems, like the HDFS, while Parquet and CSV represent common data formats. As the lakehouse paradigm promises to simplify enterprise analytic architectures and hence to overcome drawbacks of the first three integration patterns (cf. Figure 1), R1 does not allow to replicate data or metadata to different storage types (e.g. from object storage to RDBMS) or to transform it to different data file formats (e.g. from Parquet to CSV). However, the parallel utilization of multiple storage systems of the same type, e.g. several cloud object storages from different providers, is not restricted, as well as the replication of data to other storages of the same type for ensuring availability. Furthermore, data may be replicated and stored in different versions and with different schemas on the same type of storage, which allows the implementation of data processing pipelines. While all data stored in the lakehouse must leverage the same data format, metadata may be stored in different formats, since it typically shows a lower volume and serves different purposes. With the goals of avoiding complex data integration tasks and keeping enterprise analytics architectures simple and scalable, R1 is relevant for all three types of analytical workloads.

4.3.2 R2: CRUD for all Types of Data

Data mining and machine learning applications are not necessarily limited to structured data, but can also operate on semi-structured or unstructured data (cf. Section 4.2.2). For this reason, lakehouses must be able to store all kinds of data, similar to data lakes. While the possibility of writing data to storage (*C* of CRUD) and retrieving the stored data (*R*) is crucial for all types of analytical data platforms, it may also be necessary to update (*U*) and delete (*D*) data, e.g. due to changes of privacy policies or because the stored data turns out to be erroneous and hence needs to be repaired or removed. As a result, the ingestion, retrieval, updating and deleting of all kinds of data must be supported by the lakehouse at least on the level of data collections (cf. R3). This means that the lakehouse must at least allow to create, retrieve, update and delete entire data collections. R8 later refines this requirement for stream processing.

4.3.3 R3: Relational Data Collections

In data lakes, structured data is typically broken down to multiple files and stored in open and column-oriented file formats, such as Parquet. This enables more efficient column-wise aggregations and direct data access. However, just dumping the data as multiple files and providing direct read access is not sufficient for lakehouses, since especially the reporting and

Table 3: Overview about the identified lakehouse requirements and the workloads from which they were mainly derived.

Requirement		Influencing workloads		
#	Name	Reporting/OLAP	DM/ML	Streaming
R1	Same storage type and data format	●	●	●
R2	CRUD for all types of data	●	●	●
R3	Relational data collections	●	◐	◐
R4	Query language	●	○	○
R5	Consistency guarantees	●	◐	○
R6	Isolation and atomicity	●	○	●
R7	Direct read access	○	●	◐
R8	Unified batch and stream processing	○	○	●

● strong influence ◐ medium influence ○ no influence

[8] https://aws.amazon.com/s3/

[9] https://azure.microsoft.com/products/storage/blobs/

OLAP workload relies on the relational processing of data, which requires a higher degree of structure that associates the stored files with their context. Hence, lakehouses must provide concepts that allow to compose structured data to relations on the logical level, such that multiple files in the storage system can jointly represent a cohesive data collection with relational properties, such as a table-like structure (cf. (Codd, 1990)). This can be achieved e.g. by storing and managing technical metadata that contains information about the available relations, their column names and the data files holding their tuples. Also streaming applications can benefit from relational data collections, since they simplify the handling and addressing of data sources and sinks.

4.3.4 R4: Query Language

To support typical OLAP tasks, a lakehouse must offer at least a declarative, structured data query language (DQL) that allows to query at least the stored structured data in a relational manner. Such a language is necessary, because OLAP queries often have to be created in an experimental manner and with high frequency. Although additional language elements, such as those of a data management language (DML), would be desirable as well, these are not mandatory, since the associated operations could also be issued in other ways, e.g. via an API. Besides OLAP, a DQL can also be helpful for specifying the business facts that are supposed to be included into reports.

4.3.5 R5: Consistency Guarantees

As discussed for R3, structured data is typically stored as data files in column-oriented formats, such as Parquet. Since a relational data collection can consist of multiple data files, it is necessary for reporting and OLAP, but partly also for data mining, to enforce the consistency of the data within and across these files. Otherwise, aggregations and filter operations on the data are not possible in a meaningful and reliable manner. Hence, a lakehouse must provide means to enforce the consistency of data across data collections with respect to its structure. This can be achieved e.g. by employing schema validation and constraint checking. However, it is up to the implementation of the respective lakehouse to decide whether these guarantees should be enforced when new data is ingested into a data collection or when it is queried.

4.3.6 R6: Isolation and Atomicity

In order to be able to run different types of workloads and tasks in parallel, precautions must be taken to prevent lost updates and other anomalies that can arise during concurrent data accesses. This is especially relevant for the generation of reports and OLAP analyses that may be executed in parallel to write and update operations on the same data collections, but is also a prerequisite for unified batch and stream processing (cf. R8). Thus, a lakehouse must ensure atomicity and isolation (Härder and Reuter, 1983) at least for the structured data and at least on the level of data collections (cf. R3), such that incomplete or intermediate results cannot be accidentally read during concurrently executed operations that affect the same data collections. This can be implemented in various ways, e.g. via serialization techniques.

4.3.7 R7: Direct Read Access

As described in Section 4.2.2, data mining and machine learning tasks typically require direct access to the data on the storage layer, which is naturally provided by data lakes. Similarly, also lakehouses must allow unmediated read access to all stored data and metadata and leverage open, standardized file formats, so that the data and metadata can be accessed directly on the storage layer without needing to export the data. Being able to directly access the metadata is crucial, since the metadata describes the context of the stored data, links data files to data collections and may be required to ensure isolation (cf. R6). While the possibility of being able to modify the data directly on the storage layer would be desirable as well, this is not demanded, as it would likely conflict with R5 and R6 that typically need to enforce specific protocols for write access.

4.3.8 R8: Unified Batch and Stream Processing

In order to support streaming workloads, lakehouses must be able to deal with continuous streams of data, i.e. allow the ingestion of data from data streams into data collections and provide stored or updated data rapidly to stream consumers. However, since other workloads may be executed on a lakehouse as well, it may be desirable to combine stream processing with batch-based processing steps. Since R6 already assures isolation and atomicity, this is possible without risking to run into concurrency anomalies and to read intermediate results. For this reason, lakehouses provide new opportunities for breaking the boundaries between batch and stream processing and interleaving both techniques as needed, because data collections can act as sinks and sources for both batch and stream processing. However, stream processing typically requires incremental changes to small batches or even single records of data with high frequency and in near-real-time (cf. Section 4.2.3). Hence, in summary, a lakehouse must a) support the near-real-time execution of at least append, update and read operations on single records of structured data at a high rate, i.e. several operations within a second, b) allow to interleave batch processing and stream processing tasks while ensuring data integrity in accordance with R6 and c) be able to provide the updated contents of data collections as data source to external batch and stream processing tools and to ingest data from these tools into data collections as data sink.

In our opinion, the support for external batch and stream processing tools, such as Apache Spark for batch processing and Spark Structured Streaming or Apache Flink[10] for stream processing, is crucial, since both batch and streaming tasks typically require sophisticated implementations with a broad range of features and high flexibility that can barely be provided by individual platform-internal solutions.

5 TOOL EVALUATION

We applied the previously described requirements to six popular data management tools in order to evaluate whether they enable the construction of lakehouses that comply with our definition. While the requirements could also be met by combing several different tools into one data platform, we considered each of them separately, as off-the-shelf solutions are generally preferred over custom compositions and hence more promising for industrial application. However, as the frameworks Delta Lake, Apache Hudi and Apache Iceberg do not represent self-contained data platforms and are instead designed as enhancements for existing processing engines, we evaluated them in combination with Apache Spark as the hosting infrastructure. The assessment is primarily based on the available online documentations of the tools in their latest version at the time of evaluation, and in some cases supplemented by insights gained via prototyping. Table 4 summarizes the results for each tool. For those satisfying all eight requirements, we conclude that they enable to build lakehouses (cf. Table 4). Snowflake[11] was evaluated twice, one time by using internal tables and one time with external tables, as they show different characteristics.

Based on our evaluation, we conclude that currently only Delta Lake, Apache Hudi and Apache Iceberg enable the construction of lakehouses. All three frameworks operate on top of cloud object stores or the HDFS and also use them to store metadata in the JSON format. This metadata contains information about the available tables, their structure and also includes a log that tracks additions and deletions of data files in order to provide isolation and atomicity. In addition, these frameworks support SQL queries through the hosting processing engine, allow schema validation and offer comprehensive integration points for common batch and stream processing tools, e.g. Apache Spark, Apache Flink and Apache Kafka[12].

Snowflake represents a modern type of data warehouse and provides several features and characteristics that go beyond those of traditional data warehouses, including cloud deployment, the separation between compute and storage nodes and additional management capabilities for semi-structured data (Dageville et al., 2016). Nevertheless, Snowflake internally still relies on a proprietary, non-open format for storing the data in order to accelerate query processing, which prevents direct read access (cf. R7). Snowflake supports batch processing pipelines via a

Table 4: Evaluation results for six popular data management tools. The numbered columns indicate which requirements are satisfied by each tool, while the last column concludes whether they enable to build lakehouses.

Tool	**Version**	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**Lakehouses?**
Delta Lake	2.1.0	✓	✓	✓	✓	✓	✓	✓	✓	✓
Apache Hudi	0.12.1	✓	✓	✓	✓	✓	✓	✓	✓	✓
Apache Iceberg	1.0.0	✓	✓	✓	✓	✓	✓	✓	✓	✓
Snowflake with internal tables	6.31.1	✓	✓	✓	✓	✓	✓	✗	✗	✗
Snowflake with external tables	6.31.1	✗	✗	✓	✓	✗	✗	✗	✗	✗
Dremio	23.0.1	✗	✓	✓	✓	✗	✗	✗	✗	✗
Trino	394	✗	✓	✓	✓	✗	✗	✗	✗	✗

[10] https://flink.apache.org
[11] https://snowflake.com
[12] https://kafka.apache.org

convolute of periodically executed tasks and change data capture, allows Spark batch jobs to query and write data and can also ingest streaming data from Kafka and Spark Structured Streaming. However, the streaming ingestion is limited to append operations and streaming from Snowflake tables, i.e. using them as sources for stream processing tools is not supported at the time of evaluation.

When using external tables instead of internal ones, the data resides on a directly accessible, third-party cloud storage in an open format and can still be queried like an ordinary table within Snowflake. However, the metadata remains in Snowflake's metadata store (cf. R1), which is an instance of FoundationDB[13] and does not provide unmediated access (cf. R7). Furthermore, external tables are read-only and Snowflake provides no means for inserting, updating or deleting their data, which is a missing prerequisite for R2, R5, R6 and R8.

Dremio[14] is advertised as a lakehouse platform that brings self-service analytics and data warehouse functionality to data lakes. In contrast, Trino describes itself as a distributed SQL query engine which allows to query large datasets that are possibly distributed over several data sources. Despite their different focus, Dremio and Trino show several similarities from a high-level perspective: Both tools are used on top of self-contained data platforms, such as data lakes or even relational or NoSQL databases, and are hence not responsible for ingesting, storing and organizing data themselves. Instead, they are built around an SQL-based query engine and allow to query the data on these platforms. Although Dremio and Trino provide a few rudimentary DML operations for some of the supported storages, data manipulation is generally supposed to take place on the data platforms themselves. For this reason, Dremio and Trino are neither capable of providing consistency guarantees (cf. R5), nor atomicity and isolation (cf. R6) for the underlying storage layers. Dremio stores its metadata as key-value pairs within an instance of RocksDB[15], which is a different type of storage as employed for the actual data (cf. R1) and also impedes direct metadata access (cf. R7). With Trino, the storage location of the metadata depends on the storage systems that are used as data sources and the connectors that Trino provides for them. The Hive connector[16] allows Trino to access data in open file formats that resides on highly scalable storage systems, such as instances of the HDFS or cloud object stores. However, this connector also requires a Hive Metastore[17] for metadata management, which represents a different type of storage (cf. R1) and does not provide unmediated access (cf. R7). Since Dremio and Trino possess only little control over the data that resides on the data platforms, they also cannot provide support for unified batch and stream processing in accordance with R8. Hence, we do not consider them as tools that enable the construction of lakehouses.

6 CONCLUSIONS

In this paper, we first elaborated on the motivation for the recently emerging lakehouse paradigm and assessed different perspectives and definitions that are available in literature. As we found the existing definitions insufficient, we proposed a new definition based on the promises and key benefits of lakehouses in comparison to prevalent enterprise analytics architectures. This definition allowed us to derive eight technical requirements, which can be used to verify whether given data platforms represent full-fledged lakehouses. We subsequently applied these requirements to six popular data management tools and investigated to which degree they support the construction of lakehouses that comply with our definition.

As a result of this evaluation, we found that of the reviewed tools, only Delta Lake, Apache Hudi and Apache Iceberg were able to satisfy all of our requirements. These tools represent feature-rich frameworks that operate on top of highly scalable and directly accessible storages and leverage additional metadata to enhance them for lightweight data warehousing capabilities. Hence, the resulting data platforms can be considered advanced data lakes that follow the pattern "Integrated Architecture" as shown in Figure 1 and allow to serve all kinds of analytical workloads.

In contrast, the other assessed tools, including Snowflake and Dremio, provide only individual enhancements for data lakes, such as SQL query capabilities. Thus, they need to be complemented by other frameworks in order to become able to meet all requirements and allow the construction of lakehouses.

In future work, we plan to expand our evaluation to further data management tools, to investigate the suitability and maturity of lakehouse concepts for industrial applications and to assess the implications of the so-called Delta architecture.

[13] https://foundationdb.org
[14] https://dremio.com
[15] http://rocksdb.org
[16] https://trino.io/docs/current/connector/hive.html
[17] https://hive.apache.org

REFERENCES

Alonso, P. J. (2016, October). *SETA, a suite-independent agile analytical framework.* Master thesis, Polytechnic Univ. of Catalonia, BarcelonaTech.

Armbrust, M., Das, T., Sun, L., & others. (2020). Delta lake: high-performance ACID table storage over cloud object stores. *Proceedings of the VLDB Endowment, 13*, 3411–3424.

Armbrust, M., Ghodsi, A., Xin, R., & others. (2021). Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. *Proceedings of CIDR.*

Azeroual, O., Schöpfel, J., Ivanovic, D., & others. (2022). Combining Data Lake and Data Wrangling for Ensuring Data Quality in CRIS. *CRIS2022: 15th International Conference on Current Research Information Systems.*

Begoli, E., Goethert, I., & Knight, K. (2021). A Lakehouse Architecture for the Management and Analysis of Heterogeneous Data for Biomedical Research and Mega-biobanks. *2021 IEEE International Conference on Big Data*, (pp. 4643–4651).

Behm, A., Palkar, S., Agarwal, U., & others. (2022). Photon: A Fast Query Engine for Lakehouse Systems. *Proceedings of the 2022 Internat. Conf. on Management of Data*, (pp. 2326–2339).

Bose, R. (2009, March). Advanced analytics: opportunities and challenges. *Industrial Management & Data Systems, 109*, 155–172.

Chaudhuri, S., & Dayal, U. (1997, March). An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec., 26*, 65–74.

Codd, E. F. (1990). *The relational model for database management: version 2.* Addison-Wesley Longman Publishing Co., Inc.

Codd, E. F., Codd, S. B., & Salley, C. T. (1993). Providing OLAP (on-line analytical processing) to user-analysts. *An IT Mandate. White Paper. Arbor Software Corporation, 4.*

Dageville, B., Cruanes, T., Zukowski, M., Antonov, V., Avanes, A., Bock, J., Unterbrunner, P. (2016, June). The Snowflake Elastic Data Warehouse. *Proceedings of the 2016 International Conference on Management of Data.* ACM.

Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. *Harvard business review, 96*, 108–116.

Eckerson, W. (2020, June 8). All Hail, the Data Lakehouse! (If Built on a Modern Data Warehouse). Retrieved December 8, 2022, from https://www.eckerson.com/articles/all-hail-the-data-lakehouse-if-built-on-a-modern-data-warehouse

Eichler, R., Giebler, C., Gröger, C., & others. (2021). Modeling metadata in data lakes—A generic model. *Data & Knowledge Engineering, 136*, 101931.

Feinberg, D., Russom, P., & Showell, N. (2022, June). *Hype Cycle for Data Management.* Gartner Inc.

Fourny, G., Dao, D., Cikis, C. B., & others. (2021). RumbleML: program the lakehouse with JSONiq. arXiv.

Giebler, C., Gröger, C., Hoos, E., & others. (2019). Leveraging the data lake: Current state and challenges. *Internat. Conf. on Big Data Analytics and Knowledge Discovery*, (pp. 179–188).

Giebler, C., Gröger, C., Hoos, E., & others. (2020). A Zone Reference Model for Enterprise-Grade Data Lake Management. *IEEE 24th Internat. Enterprise Distributed Object Computing Conf.*, (pp. 57-66).

Giebler, C., Gröger, C., Hoos, E., & others. (2021). The Data Lake Architecture Framework. *BTW 2021.*

Gröger, C. (2021). There is no AI without data. *Communications of the ACM, 64*, 98–108.

Gröger, C. (2022). Industrial analytics–An overview. *it-Information Technology.*

Gröger, C., Schwarz, H., & Mitschang, B. (2014). The Manufacturing Knowledge Repository. *Proceedings of the 16th International Conference on Enterprise Information Systems*, (pp. 39-51).

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques.* Morgan kaufmann.

Hansen, J. (2021, April 1). Selling the Data Lakehouse. Retrieved December 8, 2022, from https://medium.com/snowflake/a9f25f67c906

Härder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM computing surveys (CSUR), 15*, 287–317.

Hlupić, T., Oreščanin, D., Ružak, D., & others. (2022). An Overview of Current Data Lake Architecture Models. *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology*, (pp. 1082–1087).

Inmon, B., Levins, M., & Srivastava, R. (2021, October). *Building the Data Lakehouse.* TECHNICS PUBN LLC.

Inmon, W. H. (2005). *Building the data warehouse.* John wiley & sons.

Kejariwal, A., Kulkarni, S., & Ramasamy, K. (2017). Real Time Analytics: Algorithms and Systems. arXiv.

Kreps, J. (2014, July). Questioning the Lambda Architecture. Retrieved December 8, 2022, from https://www.oreilly.com/radar/questioning-the-lambda-architecture/

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014, June). Industry 4.0. *Business & Information Systems Engineering, 6*, 239–242.

Leano, H. (2020, November). Delta vs. Lambda: Why Simplicity Trumps Complexity for Data Pipelines. Retrieved December 8, 2022, from https://www.databricks.com/blog/2020/11/20/delta-vs-lambda-why-simplicity-trumps-complexity-for-data-pipelines.html

L'Esteve, R. (2022, July). *The Azure Data Lakehouse Toolkit.* Apress.

Oreščanin, D., & Hlupić, T. (2021). Data Lakehouse - a Novel Step in Analytics Architecture. *44th Internat. Conv. on Information, Communication and Electronic Technology*, (pp. 1242-1246).

Pendse, N., & Creeth, R. (1995). The OLAP report. *Business Intelligence*.

Raina, V., & Krishnamurthy, S. (2022). *Building an Effective Data Science Practice.* apress, Springer.

Shiyal, B. (2021, June). *Beginning Azure Synapse Analytics.* Apress.

Tovarňák, D., Raček, M., & Velan, P. (2021). Cloud Native Data Platform for Network Telemetry and Analytics. *17th Internat. Conference on Network and Service Management*, (pp. 394-396).

Vaisman, A., & Zimányi, E. (2022). Data Warehouse Concepts. In *Data Warehouse Systems: Design and Implementation* (pp. 45-74). Berlin, Heidelberg: Springer Berlin Heidelberg.

Warren, J., & Marz, N. (2015). *Big Data: Principles and best practices of scalable realtime data systems.* Simon and Schuster.

Zheng, J. G. (2017, November). Data Visualization in Business Intelligence. In *Global Business Intelligence* (pp. 67–81). Routledge.

Zhou, Z.-H. (2021). *Machine learning.* Springer Nature.