The Data Platform Evolution: From Data Warehouses over Data Lakes to Lakehouses

Jan Schneider[1], Christoph Gröger[2] and Arnold Lutsch[2]

[1]*Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany*
[2]*Robert Bosch GmbH, Borsigstraße 4, 70469 Stuttgart, Germany*

Abstract
The continuously increasing availability of data and the growing maturity of data-driven analysis techniques have encouraged enterprises to collect and analyze huge amounts of business-relevant data in order to exploit it for competitive advantages. To facilitate these processes, various platforms for analytical data management have been developed: While data warehouses have traditionally been used by business analysts for reporting and OLAP, data lakes emerged as an alternative concept that also supports advanced analytics. As these two common types of data platforms show rather contrary characteristics and target different user groups and analytical approaches, enterprises usually need to employ both of them, resulting in complex, error-prone and cost-expensive architectures. To address these issues, efforts have recently become apparent to combine features of data warehouses and data lakes into so-called lakehouses, which pursue to serve all kinds of analytics from a single data platform. This paper provides an overview on the evolution of analytical data platforms from data warehouses over data lakes to lakehouses and elaborates on the vision and characteristics of the latter. Furthermore, it addresses the question of what aspects common data lakes are currently missing that prevent them from transitioning to lakehouses.

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

1. Introduction

Within the course of the digital transformation of society and economy, the importance of data for enterprises is continuously growing. Due to the ever-increasing affordability of smart devices and sensors in the scope of the Internet of Things [1], as well as a wide range of other upcoming technologies for capturing data about products, shop floors, suppliers, customers and other entities, enterprises have gained manifold opportunities for collecting business-related data along their value chains. By leveraging data-driven analysis techniques, this data can be exploited for evaluating and optimizing products and business processes and hence constitutes a key factor for continuous development and improvement. However, in order to be able to derive valuable insights and knowledge from huge amounts of collected data, this data needs to be organized and prepared in a systematic manner, along with metadata that describes the context in which the data was created and processed [2]. *Platforms for analytical data management* can support these tasks, as they are specifically developed for the storage, management, processing and provisioning of data from all types of data sources that is supposed to be made available for different types of analytics applications [3]. In practice, especially the traditional data warehouses and the more recent data lakes have become the predominant types of data platforms. With so-called *lakehouses*, a supposedly new kind of data platform has recently attracted attention: They are driven by the vision of combining the characteristics and features of data warehouses and data lakes, which are perceived as complementary, into integrated data platforms. With the prospect of being able to serve all kinds of analytical workloads from one universally applicable platform, lakehouses promise to simplify and improve existing enterprise analytics architectures, which commonly needed to operate data warehouses and data lakes in parallel and hence suffered from high operational costs, slow analytical processes, as well as a low trustworthiness of analysis results [4]. Over the past years, a variety of technologies have emerged or evolved with the intention to address these issues and hence to enable the construction of lakehouse-like data platforms, such as Delta Lake[1], Dremio[2] or Snowflake[3]. As indicated by our evaluation of several data management tools [5], frameworks that operate on top of data lakes and pursue to enhance them for typical features of data warehouses appear to be particularly promising in this regard, including Delta Lake, Apache Hudi[4] and Apache Iceberg[5]. This paper first provides an overview on the evolution of data platforms and explains the vision

GvDB'23: 34th Workshop on Foundations of Database Systems, June 07–09, 2023, Hirsau, Germany
{firstname.lastname}@ipvs.uni-stuttgart.de (J. Schneider); {firstname.lastname}@de.bosch.com (C. Gröger); {firstname.lastname}@de.bosch.com (A. Lutsch)

[1]https://delta.io
[2]https://www.dremio.com
[3]https://www.snowflake.com
[4]https://hudi.apache.org
[5]https://iceberg.apache.org

behind the lakehouse paradigm. Section 3 then elaborates on the characteristics of lakehouses, which are compared to the architecture of a typical data lake in Section 4. This way, several aspects are identified that conventional data lakes need to address in order to be able to complete the transition to lakehouses.

2. Evolution of Data Platforms

Between 1960 and 1970, the first databases appeared and the relational data model [6] was developed. The purpose of these databases was primarily to provide data management capabilities for applications and were accordingly designed for workloads where rather simple read and write operations have to be performed on large datasets with high frequency. However, many of these databases are less suitable for analytics applications, where large amounts of historical data have to be sporadically analyzed with rather complex queries in order to derive insights and knowledge that can then be used for guiding business decisions. For this reason, data platforms for analytical data management have been developed, which support the systematic long-term storage, management and querying of data for analytical purposes.

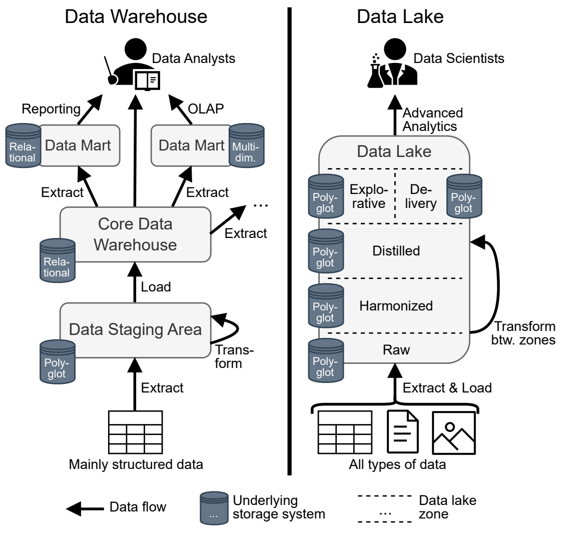

Figure 1: Comparison of typical high-level architectures of data warehouses (left) and data lakes (right).

Data warehouses [7] represent the most established type of analytical data platform and emerged from relational database systems in the 1980s. They are primarily designed for the management of structured data, impose well-defined and possibly multi-dimensional data models [8], often provide ACID guarantees and tend to offer features that go beyond those of conventional relational databases, such as time travel and data governance capabilities. The left side of Fig. 2 shows the common architecture of a data warehouse, based on the reference architecture by Bauer and Günzel [9]. Data Warehouses are typically designed specifically for a given application scenario and employ a Extract-Transform-Load (ETL) process, where the data is first extracted from the data sources, then prepared and transformed into the target schema in a dedicated *data staging area* and finally load into the *core data warehouse*, which is responsible for the long-term storage of all data. While the data staging area can leverage different types of storage systems, such as relational or NoSQL databases, the core data warehouse typically relies on relational databases. Due to the large amount of data that resides in the core data warehouse, it can be reasonable to extract parts of the data and to make it available in dependent data marts [10], which then allow to speed up downstream analyses. For example, some data marts may be based on relational databases and optimized for reporting, while other data marts employ multi-dimensional databases in order to support Online Analytical Processing (OLAP) [10]. By using appropriate query languages, data analysts can perform their analyses either on individual data marts or directly on the core data warehouse. As data warehouses employ complex, static data models, store pre-processed data instead of the raw data and leverage proprietary data formats that impede direct data access, they are mostly suited for analysis questions that are already known in advance and provide only very limited support for data mining and machine learning. Moreover, since data warehouses are primarily optimized for the batch processing of huge amounts of data, they can barely be used for streaming applications [11] that rely on the near-realtime execution of simple data operations with high frequency. With the goal of making data warehouses more flexible, there have been various attempts to enable the storage of structured raw data. For example, data vault [12] represents a data modeling approach that facilitates the easy incorporation of changes to the data schema without requiring adjustments to the structure of existing tables and hence accommodates the variability of raw data.

The continuously increasing demand for organizing and analyzing semi-structured and unstructured data led to the emergence of data lakes [13] in about 2010. Data lakes are based on the idea of collecting raw data from the data sources and deciding at a later point how this data can be processed and analyzed. This leads to a Extract-Load-Transform (ELT) process, where the data is first extracted and load into the data lake and subsequently prepared and transformed in order to make it accessible for different types of analytics applications. As a result, data lakes manage not only preprocessed and pre-aggregated data, but also raw data, which allows to increase the efficiency of re-occurring analyses while still maintaining a high level of flexibility. As indicated on the right-hand side of Fig. 2, data lakes typically impose a polyglot architecture, in which several different systems for data storage and data processing are utilized, including relational and NoSQL databases, distributed file systems, batch and stream processing engines and event hubs. By applying zone models, the architecture is commonly divided into zones that reflect different degrees of data processing and governance policies [14]. Instead of proprietary file formats, data lakes tend to leverage open file formats, such as Apache Parquet[6] or Apache ORC[7]. These formats enable tabular data representations and provide further optimizations in terms of data compression and query processing. These aspects and the possibility to directly access the data on the underlying storage systems enable the execution of data mining and machine learning applications on top of data lakes. By integrating stream storage and stream processing systems, such as Apache Spark and Apache Kafka[8], respectively, into the architecture and by applying well-established architecture patterns like the Lambda [15] or Kappa [16] architecture, data lakes are also suitable for near-realtime reporting and streaming analytics.

Due to these complementary alignments of data warehouses and data lakes, enterprises tend to employ complex analytics architectures in which both types of data platforms are operated in parallel This approach commonly results in several shortcomings [4], such as data replication across multiple storage systems and the need for continuously transferring, transforming and synchronizing the data between the involved data platforms, which likely leads to high operational costs and inconsistent or erroneous data. In addition, the necessary movement of data extends the time until analysis results are available. Vendors of various data management tools have recognized these problems and recently developed products that pursue to close the gap between data warehouses and data lakes: On the one hand, modern and possibly cloud-based data warehouses like Snowflake are evolving in order to support the management of unstructured data, the stream ingestion of near-realtime data, as well the querying of data that is stored in open formats on external, third-party storage systems. On the other hand, frameworks and query engines like Apache Hudi, Apache Iceberg, Dremio and Trino[9] are emerging that can be used to enhance data lakes by typical features of data warehouses and hence make analyses more convenient. This observable convergence of data warehouses and data lakes contributed to the coining of the term "lakehouse" and its underlying vision.

[6]https://parquet.apache.org
[7]https://orc.apache.org
[8]https://kafka.apache.org
[9]https://trino.io

3. The Lakehouse Paradigm

Although there is a widespread agreement that lakehouses represent amalgamations of data warehouses and data lakes, different opinions in literature exist about how the architecture of lakehouses should look like and what characteristics these data platforms must necessarily possess. For example, many authors consider lakehouses as integrated data platforms that are based on directly-accessible storage, such as distributed file systems or object storages and can also provide typical features of data warehouses like ACID transactions [4]. However, others argue that a two-tier architecture consisting of self-contained data warehouses and data lakes that are potentially connected by an integration layer for unified data access can also constitute a lakehouse [17]. In our work [5], we assessed different views and definitions of the lakehouse paradigm and finally derived a new definition that reflects the additional value of lakehouses for enterprises in comparison to conventional data platforms. From our perspective, lakehouses are beneficial for enterprises when they contribute to simplifying enterprise analytics architectures by providing a single source of truth, limiting the variety of involved technologies and hence reducing the number of required data movement and transformation processes. Accordingly, we define a lakehouse as *"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."* [5]. Fig. 3 illustrates how such a data platform may look like. First of all, the term "integrated platform" expresses that a lakehouse should not be considered as a loose amalgamation of standalone data warehouses and data lakes, but rather as a single, self-contained data platform. Limiting the architecture to one type of storage, e.g. to a distributed file system, and one data format, e.g. to Apache Parquet, eliminates the need for additional data movement and transformation processes within the lakehouse and therefore reduces the complexity and error-proneness of the overall architecture. Furthermore, it supports the formation of a single source of truth, as the same data may no longer be replicated between different systems with varying characteristics. Finally, the definition emphasizes that lakehouses must support all typical analytical workloads of data warehouses and data lakes, so that data analysts and data scientists can use a lakehouse instead of the former data platforms.

Based on this definition and the characteristics of the workloads mentioned therein, we derived a total of eight technical requirements that lakehouses should fulfill [5]:

R1: Same type of storage and data format Lakehouses must employ only a single type of storage for all data and metadata and use only one format for the data.

R2: CRUD for all types of data Lakehouses must support the ingestion, retrieval, updating and deletion of

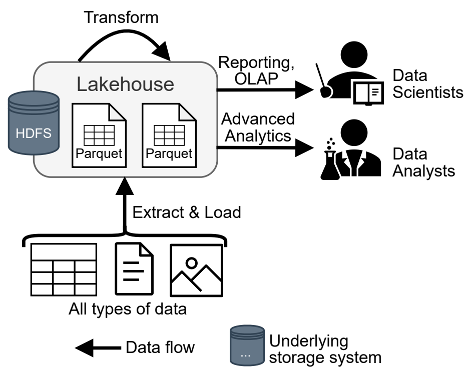

Figure 2: Example of a lakehouse that uses the HDFS as storage system and Apache Parquet as data format.

all kinds of data at least on the level of data collections.

R3: Relational data collections Lakehouses must provide means to abstract from the stored data files and to represent them as cohesive data collections with relational properties on the logical level.

R4: Query language Lakehouses must offer a declarative, structured query language that allows to query the data in a relational manner.

R5: Consistency Guarantees Lakehouses must provide consistency guarantees for the data, such as schema validation, which can either be enforced on data ingestion or when the data is queried.

R6: Isolation and Atomicity Similar to relational database systems, lakehouses must provide isolation and atomicity for data operations in order to ensure the consistency of the data and to support concurrency.

R7: Direct read access Lakehouses must provide direct access to the data and metadata on the underlying storage system and must employ open data formats only.

R8: Unified batch and stream processing Lakehouses must support record-wise data operations in near-realtime and allow to treat data collections as sources and sinks for batch and stream processing.

These requirements can be achieved in various ways, for example by opening existing data warehouses and driving them into the direction of data lakes or by developing technologies that enhance data lakes for common features and characteristics of data warehouses.

4. Transitioning from Data Lakes to Lakehouses

In the course of our evaluation of several data management tools [5], frameworks for data lakes like Delta Lake, Apache Hudi and Apache Iceberg appeared to be particularly promising for the fulfillment of the aforementioned requirements and thus for the construction of lakehouses. These frameworks basically act as libraries for highly scalable batch and stream processing engines, such as Apache Spark[10] or Apache Flink[11] and implement data access protocols that control how these engines read data from and write data to storage systems (cf. [18]). In addition, they manage technical metadata, which allows them to represent datasets as relational data collections and track additions, updates and deletions of data.

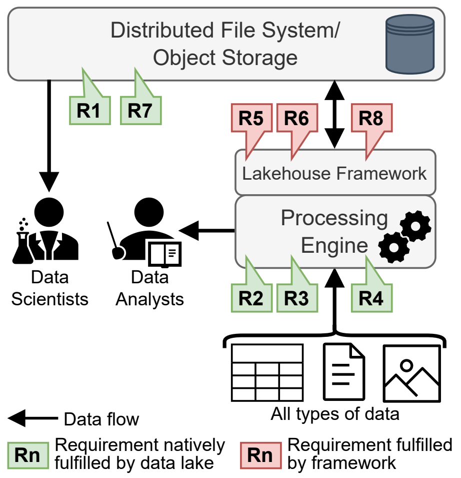
Distributed File System/ Object Storage
R1
R7
R5
R6
R8
Lakehouse Framework
Processing Engine
R2
R3
R4
Data Scientists
Data Analysts
All types of data
Data flow
Rn Requirement natively fulfilled by data lake
Rn Requirement fulfilled by framework

Figure 3: Typical architecture of a data lake that can transition to a lakehouse by adding a corresponding framework.

Fig. 4 shows the conceptual architecture of a data lake as it can often be encountered in practice. It essentially consists of a storage system, which can be either a distributed file system or an object storage that persists the data as data files in an open file format. A batch and stream processing engine can read data from the storage system, process it and then write the results back to the storage system. Hence, the data lake is supposed to store the raw data next to pre-processed and aggregated data. This processing engine is also used to ingest data and data analysts can leverage it in order to query the data via a query language like SQL. For data mining and machine learning, data science applications can directly access the data on the storage system. Without the lakehouse framework that is depicted in Fig. 4, the data lake would already satisfy the requirements R1, R2, R3, R4, and R7. R3 is satisfied because many processing engines like Apache Spark already enable relational data abstraction, so that multiple data files that reside on the storage system can collectively represent the contents of a table. R5 and R6 are not met, since processing engines

[10]https://spark.apache.org
[11]https://flink.apache.org

usually do not provide means for enforcing the internal consistency of a table, nor do they guarantee atomicity and isolation when performing operations on the data. Although processing engines like Apache Spark generally support the batch and stream processing of data that resides on a distributed file system or object storage, R8 is often not met, because especially engines that apply micro-batching are often not optimized for simple data operations that occur at high frequency, which results in the creation of many small data files when streaming data needs to be materialized. This high number of data files prevents the efficient querying of data, as many files have to be read and consolidated [18]. To solve this issue, a dedicated stream storage system, such as Apache Kafka, could be leveraged, but this would in turn increase the complexity of the data lake and in particular violate R1, as it represents another type of storage system.

When integrating a lakehouse framework into the processing engine, the previously unmet requirements R5, R6, and R8 can be satisfied [5]: As these frameworks provide means for enforcing the inner consistency of data collections, such as schema validation and constraint checking, R5 can be fulfilled. Furthermore, they use the collected technical metadata in order to implement data access protocols that achieve atomicity and at least snapshot isolation [19] via multi-version concurrency control [20] (cf. R6). By offering various optimizations, such as different table types that are either designed for frequent reads or writes, as well as compaction techniques for data and metadata, these frameworks avoid the creation of many small data and metadata files and hence increase the efficiency of stream processing (cf. R8).

5. Conclusion

By assessing the properties of a typical data lake architecture and comparing them to requirements that are relevant for lakehouses, it became apparent that it lacks consistency guarantees, atomicity and isolation for data operations, as well as optimizations for stream processing in order to complete the transition to a lakehouse. While the lakehouse approach looks promising, its concepts and technologies have not reached maturity yet and hence require further research, for example in terms of data modeling and the suitability of different architectures.

References

[1] O. Vermesan, P. Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, River publishers, 2013.

[2] DAMA International, DAMA-DMBOK: Data Management Body of Knowledge, second ed., Technics Publications, 2017.

[3] C. Gröger, Industrial Analytics – An Overview, it - Information Technology 64 (2022) 55–65.

[4] M. Armbrust, A. Ghodsi, R. Xin, et al., Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics, in: 11th CIDR, 2021.

[5] J. Schneider, C. Gröger, A. Lutsch, et al., Assessing the Lakehouse: Analysis, Requirements and Definition, in: Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS), 2023, pp. 44–56.

[6] E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the ACM 13 (1970) 377–387.

[7] W. H. Inmon, Building the Data Warehouse, John Wiley & Sons, 2005.

[8] R. Kimball, M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, third ed., John Wiley & Sons, 2013.

[9] A. Bauer, H. Günzel, Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung, dpunkt.verlag, 2013.

[10] H. Baars, H.-G. Kemper, Business Intelligence & Analytics, fourth ed., Springer Vieweg, 2021.

[11] T. Akidau, S. Chernyak, R. Lax, Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing, O'Reilly Media, 2018.

[12] D. Linstedt, M. Olschimke, Building a Scalable Data Warehouse with Data Vault 2.0, Elsevier Science & Technology Books, 2015.

[13] C. Giebler, C. Gröger, E. Hoos, et al., Leveraging the Data Lake: Current State and Challenges, in: Big Data Analytics and Knowledge Discovery, Springer International Publishing, 2019.

[14] C. Giebler, C. Gröger, E. Hoos, et al., A Zone Reference Model for Enterprise-Grade Data Lake Management, in: 24th Internat. Enterprise Distributed Object Computing Conference (EDOC), 2020.

[15] J. Warren, N. Marz, Big Data: Principles and Best Practices of Scalable Realtime Data Systems, Simon and Schuster, 2015.

[16] J. Kreps, Questioning the Lambda Architecture, 2014. URL: https://www.oreilly.com/radar/questioning-the-lambda-architecture/.

[17] D. Oreščanin, T. Hlupić, Data Lakehouse - A Novel Step in Analytics Architecture, in: 44th International Convention on Information, Communication and Electronic Technology (MIPRO), 2021.

[18] M. Armbrust, T. Das, L. Sun, et al., Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores, Proc. VLDB Endow. 13 (2020).

[19] G. Weikum, G. Vossen, Transactional Information Systems, Elsevier, 2001.

[20] P. Jain, P. Kraft, C. Power, et al., Analyzing and Comparing Lakehouse Storage Systems, CIDR, 2023.