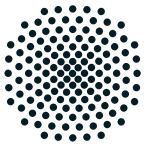


Universität Stuttgart

Architekturkonzepte zur Datenverwaltung in Data Lakes

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Corinna Giebler

aus Ostfildern

Hauptberichter: Prof. Dr. Bernhard Mitschang

Mitberichter: Prof. Dr. Ulf Leser

Tag der mündlichen Prüfung: 08. Dezember 2021

Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware

2021

Inspiration is waiting,
Rise up, don't think twice.
Put your fate in your hands
Take a chance
Roll the dice.

— *Critical Role*

Inhaltsverzeichnis

Kurzfassung

Die zunehmende Digitalisierung in zahlreichen Bereichen und die damit verbundene Vielzahl an heterogenen Daten, die gespeichert, verwaltet und analysiert werden müssen, stellen eine Herausforderung für traditionelle Datenmanagementkonzepte dar. Insbesondere gilt es, den potentiellen Wert der Daten auszunutzen und durch neue Erkenntnisse Kosten zu senken und Effizienz zu erhöhen. Um die Verwaltung und flexible Analyse der generierten Daten zu ermöglichen, wurde das Konzept des Data Lake entwickelt. Daten heterogener Struktur werden hier in ihrer Rohform gespeichert, sodass auch lange nach ihrer Erfassung beliebige Anwendungsfälle darauf realisiert werden können.

Soll allerdings ein solcher Data Lake für die praktische Nutzung in z.B. einem Unternehmen umgesetzt werden, zeigen sich zahlreiche Probleme und Lücken auf. Methodische Grundlagen sind unvollständig, vage oder fehlen ganz. So gibt es keine vollständige Data-Lake-Architektur oder einen Leitfaden, um eine solche zu erstellen. Auch fehlt es an einer passenden Datenorganisation, um die Vielzahl an Anwendungsfällen und Nutzergruppen eines unternehmensweiten Data Lake zu unterstützen.

In dieser Arbeit werden diese Lücken adressiert. Hierzu werden drei Forschungsziele formuliert: Z1-Identifikation der Eigenschaften eines Data Lake, Z2-Erstellung eines Leitfadens zur Definition einer vollständigen Data-Lake-Architektur und Z3-Erarbeitung einer internen Data-Lake-Organisation. Diese Forschungsziele werden durch insgesamt sieben Forschungsbeiträge abgedeckt. Hierfür wird zunächst in einer vollständigen Literaturrecherche das Konzept

des Data Lake identifiziert und definiert. Im zweiten Schritt stellt diese Arbeit das Data Lake Architecture Framework (DLAF) vor, welches die Definition einer vollständigen Data-Lake-Architektur ermöglicht. Abschließend bietet das Zonenreferenzmodell einen systematischen Ansatz zur Datenorganisation in Data Lakes. Die Umsetzbarkeit der erarbeiteten Lösungen wird mithilfe einer prototypischen Implementierung für ein reales Anwendungsszenario gezeigt. Eine abschließende Evaluation bestätigt, dass die entwickelten Lösungen vollständig sind, zahlreiche Vorteile bieten und so die Industrialisierung von Data Lakes unterstützen.

Abstract

Increasing digitization in numerous areas and the associated multitude of heterogeneous data that must be stored, managed and analyzed pose a challenge to traditional data management concepts. In particular, the aim is to exploit the potential value of the data and to reduce costs and increase efficiency through new insights. To enable the management and flexible analysis of the generated data, the concept of the data lake was developed. Data of heterogeneous structure is stored here in its raw form so that any use cases can be realized on it even long after it has been captured.

However, if such a data lake is to be implemented for practical use in a company, for example, numerous problems and gaps become apparent. Methodical foundations are incomplete, vague, or missing altogether. For example, there is no comprehensive data lake architecture or guideline for creating one. There is also a lack of appropriate data organization concepts to support the multitude of use cases and usergroups of an enterprise-wide data lake.

In this thesis, these gaps are addressed. To this end, three research objectives are formulated: Z1-Identifying the characteristics of a data lake, Z2-Creating a guideline for defining a comprehensive data lake architecture, and Z3-Creating an internal data lake organization. These research objectives are covered by a total of seven research contributions. To do so, a comprehensive literature review is first conducted to identify and define the concept of data lake. In the second step, this paper presents the Data Lake Architecture Framework (DLAF), which enables the definition of a comprehensive data lake architecture. Finally,

the zone reference model provides a systematic approach to data organization in data lakes. The feasibility of the developed solutions is demonstrated with the help of a prototypical implementation for a real application scenario. A final evaluation confirms that the developed solutions are complete, offer numerous advantages and thus support the industrialization of Data Lakes.

Danksagungen

Diese Dissertation entstand im Rahmen eines Forschungsprojekts mit einem weltweit agierenden Industriepartner am Institut für Parallele und Verteilte Systeme (Abteilung Anwendersoftware) der Universität Stuttgart.

An dieser Stelle möchte ich mich herzlich bei all jenen bedanken, die mich über die Erstellung dieser Arbeit hinweg begleitet und unterstützt haben. Besonderer Dank gilt an dieser Stelle meinem Doktorvater Prof. Dr. Bernhard Mitschang, welcher durch fortwährende Unterstützung diese Arbeit ermöglicht hat. Insbesondere möchte ich mich an dieser Stelle für die Diskussionen, die wissenschaftliche Ausrichtung, sowie die Organisation des Forschungsprojektes bedanken. Prof. Dr. Ulf Leser danke ich für die Arbeit als Mitberichter. Weiterer Dank gilt meinen Betreuern Holger Schwarz vom IPVS, sowie Christoph Gröger, Eva Hoos und Thomas Müller. Der regelmäßigen Austausch, die tiefen Diskussionen und die Einblicke in sowohl die wissenschaftliche als auch die praktische Welt haben diese Arbeit und auch mich selbst immens bereichert. Auch möchte ich mich an dieser Stelle für die Ermutigungen, den persönlichen Beistand und all die Tipps und Tricks bedanken, die das Promovieren deutlich angenehmer gestaltet haben.

Ich möchte mich bei meinen Kollegen am IPVS bedanken, für den Austausch auf wissenschaftlicher und auch persönlicher Ebene. Dabei möchte ich besonders hervorheben (in alphabetischer Reihenfolge): Ana Franco da Silva, Christoph Stach, Frank Steimle (auch für die oft sehr notwendigen Kaffeepausen), Julian Ziegler (auch als Bürokollege und Brainstorm-Opfer), Manuel

Fritz (auch für das gegenseitige Paperlesen), Mathias Mormul (auch für die Schreibworkshops und den Richter), Michael Behringer, Pascal Hirmer, Peter Reimann, Rebecca Eichler und Sarah Oppold (auch für das gemeinsame organisieren von Gruppenaktivitäten und konspirativer Frauensachen). An dieser Stelle sei auch Sven Michalczyk vom KIT gedankt für den Austausch über die Promotion hinweg. Auch meinen Kollegen beim Industriepartner gilt mein tiefer Dank, unter anderem aber nicht ausschließlich Ingo Hollenbeck, Melanie Dold, Leonie Haug, Stefanie Layer, Matthias Link, Arnold Lutsch, Matthias Osswald, Yannick Sigwalt und Yulia Svetashova. Ebenfalls möchte ich mich bei der Verwaltung des IPVS bedanken, insbesondere bei Christine Well und Stefanie Palmer, sowie bei meinen Studenten Simone Schmidt, Fabian Geiger und Marc Altvater, deren Arbeiten zum Inhalt dieser Dissertation beigetragen haben.

Mein Dank gilt meinen Eltern und meinen Schwestern, die mich über mein Studium und auch meine Promotion hinweg immer unterstützt haben. Ich möchte auch Franziska Tomschi danken, für fast neun Jahre wundervolle Freundschaft, die mir immer den Rücken stärkt, auch über die Promotion hinweg. Auch möchte ich ihr an dieser Stelle noch einmal herzlich für das Titelbild zu dieser Arbeit danken. Mein Dank gilt auch Nicole Ondrusch und meinem E-Team, die mich durch hilfreiche Tipps und Unterstützung nicht nur wissenschaftlich, sondern auch außerhalb bereichert haben. Last but most certainly not least I want to thank the lovely beans that I met online, who have grown into a close-knit little family: Blitz, Kati, Kirby, Lallie, and usual. Your support means the world to me and I can’t express how thankful I am to have all of you.

Kapitel 1

Einleitung

Mit der zunehmenden Digitalisierung in zahlreichen Bereichen, z. B. der Industrie oder dem Gesundheitswesen, werden datengetriebene Prozessoptimierung und Kostensenkung ermöglicht. Hierzu wird eine Vielzahl von Daten gesammelt, die sehr voluminös und oftmals heterogen, d. h. von unterschiedlicher Struktur (strukturiert, semi-strukturiert, unstrukturiert), sind. Man spricht hierbei von *Big Data*. Big Data werden durch verschiedene Eigenschaften charakterisiert, unter anderem durch hohes Volumen, hohe Vielfalt in den Daten, hohe Erfassungsgeschwindigkeit und den potentiellen Wert, den die Daten enthalten. Man spricht in diesem Fall von potentiellem Wert, da zum Erfassungszeitpunkt nicht immer klar ist, wie und ob sie in späteren Anwendungsfällen wertbringend genutzt werden können. Um den, wenn auch nur potentiell, enthaltenen Wert auszunutzen, müssen die Daten gespeichert, verwaltet und verarbeitet werden. Herkömmliche Konzepte, wie das Data Warehouse, sind ungeeignet für die Verwaltung dieser Daten, insbesondere da das Data Warehouse lediglich einen bereinigten und aufbereiteten Ausschnitt der Daten enthält, dessen Nutzen bekannt ist [Mad15]. Andere Daten und Informationen gehen bei dieser Aufbereitung verloren. Als neues Konzept für die Verwaltung und Nutzung von Big Data wurde darum der Data Lake entwickelt [Dix10]. Die benötigten Architekturen zum Aufbau eines solchen Data Lake bilden den Fokus der vorliegenden

Arbeit, die durch dieses Kapitel eingeleitet wird. Hierzu detailliert Abschnitt 1.1 Hintergrund und Motivation der Arbeit. Abschnitt 1.2 leitet die sich daraus ergebenden Forschungsfragen und -ziele ab und formuliert die in dieser Arbeit erarbeiteten Forschungsbeiträge. Schließlich gibt Abschnitt 1.3 eine Überblick über die weitere Struktur der Arbeit.

1.1 Hintergrund und Motivation

Neue Möglichkeiten der Datenerfassung, -speicherung und -verarbeitung verändern den Stellenwert von Daten in Unternehmen und Organisationen. Schon längst gelten sie als strategisches Kapital und bestimmen maßgeblich die Entwicklung verschiedenster Domänen, von der Industrie über das Gesundheitswesen bis hin zur Forschung [Cao17]. Zu dieser Stellung trägt vor allem der potentielle Wert bei, den die erfassten Daten innehaben [TSRC15]. So können beispielsweise Sensormesswerte aus Produktionsanlagen dazu verwendet werden, tiefere Einblicke in Produktionsprozesse zu erhalten und diese vorausschauend zu steuern [LKY14].

Die traditionellen Systeme und Werkzeuge für die Speicherung und Verwaltung von Daten, etwa Data Warehouses, stießen mit dem hohen Volumen und der ebenso hohen Heterogenität der Big Data schnell an ihre Grenzen. Daher wurde eine Vielzahl neuer Konzepte entwickelt, um den potentiellen Wert der Daten ausschöpfen zu können. Der *Data Lake* ist eines dieser Konzepte, welches 2010 zum ersten Mal Erwähnung fand [Dix10]. In diesem Konzept werden alle Daten eines Unternehmens oder einer Organisation in ihrem Rohformat abgespeichert und stehen daher für beliebige Anwendungsfälle zur Verfügung [Mat17]. Durch die Speicherung der Daten in Rohform gehen keinerlei Informationen verloren und ihr Wert bleibt für spätere Analysen unverändert erhalten. Insbesondere muss dabei der spätere Verwendungszweck der Daten zum Zeitpunkt der Datenerfassung nicht bekannt sein. So ermöglicht ein Data Lake explorative und fortgeschrittene Analysen [Bos09] und bilden damit eine Grundlage für datengetriebene Effizienzsteigerung und Kostensenkung.

Für die Realisierung eines Data Lake wird eine *vollständige Data-Lake-Architektur* benötigt, in welcher geeignete Konzepte für die Umsetzung des Data Lake in Verbindung gesetzt werden. Eine Data-Lake-Architektur gilt nach

Definition dieser Arbeit dann als vollständig, wenn sie alle für die Funktion relevanten *Aspekte* eines Data Lake abdeckt. Ein Aspekt ist in dieser Arbeit definiert als eine Perspektive auf einen Data Lake, wie beispielsweise die verwendeten Datenspeicher, die Datenmodellierung, die Datenorganisation oder das Metadatenmanagement. Jeder Aspekt ist dabei mit einer Menge von Data-Lake-Konzepten assoziiert, wie zum Beispiel Konzepte für die Datenmodellierung. Einzelne Data-Lake-Konzepte können einen Aspekt teilweise, komplett oder sogar mehrere Aspekte gleichzeitig abdecken. Ein einzelner Aspekt reicht jedoch nicht aus, um einen Data Lake zu definieren, vielmehr umfasst ein Data Lake immer mehrere Aspekte.

Existierende Arbeiten zu Data Lakes bieten nur wenige Ansatzpunkte für die Definition einer solchen vollständigen Data-Lake-Architektur. Zwar gibt es Arbeiten, welche als Data-Lake-Architekturen bezeichnet werden, etwa Sharmas „Data Lake Reference Architecture“ [Sha18] oder Inmons „Data Lake Architecture“ [Inm16]. Diese Architekturen sind allerdings nicht vollständig, da relevante Aspekte von Data Lakes ausgelassen werden, z.B. Datenmodellierung oder Metadatenmanagement. Sawadogo und Darmont zeigen in ihrer Arbeit [SD21] eine dreiteilige Klassifikation von Data-Lake-Architekturen auf, nämlich in funktionale Architekturen, datenreifebasierte Architekturen und hybride Architekturen. Allerdings werden auch hier Aspekte wie Datenmodellierung und Metadatenmanagement vernachlässigt. Somit existieren in der Literatur bislang keine vollständigen Data-Lake-Architekturen.

Wollen Organisationen daher selbst eine vollständige Data-Lake-Architektur definieren, stoßen sie auf zahlreiche Herausforderungen. Zwar existieren Ansätze, um einzelne oder sogar mehrere Aspekte umzusetzen, allerdings fehlt es an Einheitlichkeit und Diskussion. So gibt es beispielsweise verschiedenste Zonenmodelle für den Aspekt der internen Datenorganisation in einem Data Lake (siehe z. B. [Gor16; Mad15; PWD17; RZ19; Sha18; ZDB+15]), die sich teilweise stark unterscheiden, aber in den beschreibenden Arbeiten nicht zueinander in Bezug gesetzt oder voneinander abgegrenzt werden. Es entsteht dadurch eine Vielzahl unterschiedlichster Konzepte, deren Eignung und Unterschiede unklar sind. Zudem existiert zum aktuellen Zeitpunkt weder eine vollständige Data-Lake-Architektur, die konkrete Konzepte umfasst, noch eine Leitlinie zur

Erstellung einer solchen Architektur, die relevante Aspekte herausarbeitet, ihre Zusammenhänge beschreibt und geeignete Konzepte vorschlägt.

1.2 Forschungsfrage, -Ziele und -Beiträge

Basierend auf den oben genannten Herausforderungen stellt sich für Unternehmen und Organisationen, die einen Data Lake umsetzen wollen, folgende Frage:

> *Wie muss ein Data Lake aufgebaut sein, um die Verwaltung und Verarbeitung von Big Data in der Praxis zu unterstützen?*

Um diese Frage zu beantworten, schafft diese Arbeit die methodischen Grundlagen für die Konzeption und Realisierung eines Data Lake. Dieses Gesamtziel lässt sich in drei Teilziele unterteilen, welche jeweils direkt mit Forschungsbeiträgen dieser Arbeit assoziiert sind. Die folgenden Absätze stellen diese Ziele (Z) und die jeweiligen Beiträge (B) vor.

Z1 - Identifikation der angestrebten Eigenschaften eines Data Lake, wie sie in der praktischen Anwendung benötigt werden

Existierende Literatur umfasst zahlreiche Data-Lake-Definitionen, die sich in ihren enthaltenen Eigenschaften teils widersprüchlich gegenüberstehen. Es gilt darum zunächst, ein grundlegendes Verständnis für das Konzept des Data Lake zu schaffen. Dabei müssen vor allem die Besonderheiten praktischer Anwendungen berücksichtigt werden. Solche Besonderheiten sind beispielsweise die Vielfalt der Nutzergruppen in einer praktischen Anwendung oder rechtliche Regularien, die es zu beachten gilt. Um dieses Ziel zu erreichen, leistet diese Arbeit folgende Beiträge:

B1.1 Konsolidierung verschiedener Data-Lake-Definitionen zu einer einheitlichen Definition.

B1.2 Identifikation der angestrebten Eigenschaften eines Data Lake.

B1.3 Identifikation von Lücken durch Abgleich zwischen angestrebten Eigenschaften aus der Praxis und Stand der Literatur.

Z2 - Erstellung eines Leitfadens zur Definition einer vollständigen Data-Lake-Architektur

Um einen Data Lake zu realisieren, braucht es eine geeignete Data-Lake-Architektur. Eine solche Architektur beschreibt, welche Konzepte und Umsetzungsmöglichkeiten für die verschiedenen Aspekte des Data Lake verwendet werden. Dabei wird jeder Aspekt ausdefiniert und mit passenden Konzepten realisiert. Insbesondere ist hier auf Zusammenhänge und Einflüsse der Aspekte untereinander zu achten. Eine bestimmte Realisierung eines Aspekts kann sich darauf auswirken, welche Konzepte sich für andere Aspekte eignen, da sich manche Konzepte über Aspekte hinweg ergänzen oder ausschließen. Da solche Architekturen stark von dem Umfeld abhängen, in dem der Data Lake eingesetzt werden soll, ist das Ziel dieser Arbeit nicht eine einzelne Architektur, sondern vielmehr ein Leitfaden zur Erstellung einer passenden Architektur. Dazu liefert diese Arbeit folgende Beiträge:

B2.1 Identifikation der Aspekte für die Umsetzung eines Data Lake, sowie deren Abhängigkeiten.

B2.2 Erstellung eines Leitfadens in Form eines Rahmenwerks, des Data Lake Architecture Framework (DLAF), das die Definition einer vollständigen Data-Lake-Architektur unterstützt und Entscheidungsunterstützung bei der Auswahl geeigneter Konzepte liefert.

Z3 - Erarbeitung einer internen Datenorganisation für Data Lakes, die die Verwaltung, Nutzung und Analyse von Big Data praxisgerecht unterstützt

Organisationen und Unternehmen setzen sich typischerweise nicht nur mit heterogenen Daten auseinander, sondern auch mit einer Vielzahl an unterschiedlichen Nutzer*innen und Anwendungsfällen. So reicht beispielsweise das Spektrum der Nutzer*innen von Data Scientists, die viel Erfahrung mit der Aufbereitung und Analyse von Daten haben, bis hin zu Nutzer*innen, die umfangreiche Unterstützung bei der Datenanalsyse benötigen. Im Bereich der Anwendungsfälle herrscht eine ähnliche Vielfalt, von explorativer Analyse mit fortgeschrittenen Methoden, wie beispielsweise Machine Learning, bis hin zu traditionellem Reporting und Online Analytical Processing (OLAP). Die Daten

für all diese Anwendungsfälle sollen dabei gemeinsam im Data Lake verwaltet werden. Beim Entwurf eines Data Lake ist es darum notwendig sicherzustellen, dass sowohl die Bedarfe der verschiedenen Nutzergruppen, als auch der unterschiedlichen Anwendungsfälle bestmöglich abgedeckt werden. Hierbei haben sich insbesondere Zonenmodelle als passendes Konzept bewährt, etwa [RZ19; Sha18]. Existierende Ansätze sind allerdings meist vage gehalten und bieten keine Hinweise auf ihre Herleitung oder verwandte Arbeiten. Um eine Datenorganisation zu erstellen, welche die Bedarfe der Praxis abdeckt, leistet diese Arbeit die folgenden Beiträge:

B3.1 Identifikation von Anforderungen an die Datenorganisation innerhalb des Data Lake.

B3.2 Erstellung eines Metamodells für Zonen für den systematischen Vergleich existierender Zonenmodelle.

B3.3 Erarbeitung des Zonenreferenzmodells zur internen Datenorganisation eines Data Lake für die Unterstützung unterschiedlicher Nutzer*innen und Anwendungsfälle.

1.3 Aufbau der Arbeit

Diese Arbeit ist wie folgt aufgebaut:

Kapitel 2 - Grundlagen: In diesem Kapitel werden die notwendigen Grundlagen für das Verständnis dieser Arbeit gelegt. Insbesondere geht es dabei auf die Definitionen von Data Lakes (vgl. Forschungsbeitrag B1.1), sowie auf Technolgien aus dem Big-Data-Umfeld ein. Auch benötigte Begriffe werden näher behandelt und voneinander abgegrenzt.

Kapitel 3 - Anwendungsszenarien und Fallbeispiel: Die Anwendungsszenarien für Data Lakes sind sehr vielfältig. Jedes dieser Szenarien bringt andere Rahmenbedingungen mit sich, auf die beim Design des Data Lake eingegangen werden muss. Dieses Kapitel umfasst eine Übersicht über mögliche Anwendungsszenarien für Data Lakes sowie deren Besonderheiten. Zudem beschreibt es das Fallbeispiel, das im Rest dieser Arbeit als Basis für eine praktische Betrachtung des Data Lake und der entwickelten Konzepte dient. Dadurch liefert es die Grundlage zur Erfüllung von Forschungsbeitrag B1.3.

Kapitel 4 - Praktische Anforderungen und Forschungslücken: In diesem Kapitel gilt es, die praktischen Anforderungen an Data Lakes herzuleiten (vgl. Forschungsbeitrag B1.2) und diese mit existierenden Arbeiten abzugleichen. So werden Forschungslücken identifiziert, die es im Bereich Data Lake noch zu adressieren gilt (vgl. Forschungsbeitrag B1.3). Zusammen mit Kapitel 2 und Kapitel 3 dient dieses Kapitel der Erfüllung von Forschungsziel Z1.

Kapitel 5 - Das Data Lake Architecture Framework: Forschungsziel Z2 bildet den Fokus dieses Kapitels. Mit dem DLAF wird hier ein Rahmenwerk vorgestellt, welches sowohl alle benötigten Aspekte für Data Lakes zusammenstellt, als auch ihre Abhängigkeiten herausarbeitet (vgl. Forschungsbeitrag B2.1). Zudem werden jedem Aspekt eine Menge möglicher Konzepte zugeordnet, die bei der Definition einer konkreten Data-Lake-Architektur verwendet werden können. Die enthaltene Methodik unterstützt Entwickler*innen bei der Erstellung einer vollständigen Data-Lake-Architektur für ein spezifisches Anwendungsszenario (vgl. Forschungsbeitrag B2.2).

Kapitel 6 - Metamodell für Zonen: Das DLAF umfasst verschiedene Aspekte, die für die Funktionsfähigkeit des Data Lake von hoher Wichtigkeit sind. Für die Erfüllung von Forschungsziel Z3 ist insbesondere der Aspekt der Datenorganisation zentral. Wie bereits erwähnt eignen sich insbesondere Zonenmodelle für dessen Umsetzung. Dieses Kapitel beschriebt das erarbeitete Metamodell für Zonen, welches die Natur einer Zone beschreibt (vgl. Forschungsbeitrag B3.2). Anhand dieses Metamodells werden existierende Zonenmodelle aus der Literatur verglichen und Gemeinsamkeiten sowie Unterschiede festgestellt.

Kapitel 7 - Das Zonenreferenzmodell: Um Forschungsziel Z3 vollständig zu erreichen, befasst sich dieses Kapitel mit den Ansprüchen aus der Praxis, die an eine Datenorganisation gestellt werden. Dafür werden zunächst die Anforderungen erarbeitet, die eine praktische Anwendung an eine Datenorganisation stellt (vgl. Forschungsbeitrag B3.1) und evaluieren existierende Konzepte hinsichtlich dieser Anforderungen. Da sich die bereits bestehenden Konzepte als unzureichend herausstellen, wird aus dem Fallbeispiel und der Literatur das Zonenreferenzmodell abgeleitet, welches unterschiedliche Anwendungsfälle und Nutzer*innen für sowohl Stapel- als auch Datenstromverarbeitung unterstützt (vgl. Forschungsbeitrag B3.3).

Kapitel 8 - Prototypische Umsetzung: Nachdem die in dieser Arbeit erstellten Konzepte in vorangegangenen Kapiteln vorgestellt wurden, beschreibt dieses Kapitel ihre konzeptionelle und physische Umsetzung als Prototyp. Als Grundlage dient hierfür ein konkretes Anwendungsszenario aus dem in Kapitel 3 vorgestellten Fallbeispiel.

Kapitel 9 - Gesamtevaluation: Der Fokus dieses Kapitels ist die Evaluation der in Kapitel 5, Kapitel 7 und Kapitel 8 entwickelten Konzepte. Hierzu wird die Vollständigkeit des DLAF gezeigt, sowie die Vor- und Nachteile des Zonenreferenzmodells anhand des Prototypen herausgearbeitet.

Kapitel 10 - Zusammenfassung und Ausblick: Dieses Kapitel schließt die vorliegende Arbeit ab und bietet einen Ausblick auf zukünftige Arbeiten.

Kapitel 2

Grundlagen

In dieser Arbeit werden verschiedene Konzepte vorgestellt, für deren Verständnis einige grundlegenden Kenntnisse benötigt werden. Diese sind Bestandteil dieses Kapitels, welches auf vorangegangenen Veröffentlichungen der Autorin basiert [GGH+19a; GGH+20a]. Abschnitt 2.1 beschreibt das Konzept des Data Lake in tieferem Detail und formuliert eine Definition auf Grundlage einer ausführlichen vorangegangenen Literaturrecherche. In Abschnitt 2.2 werden verschiedene Big-Data-Technologien vorgestellt, die oft in Data Lakes zur Anwendung kommen und deren Verständnis darum für die weitere Arbeit notwendig ist. Abschnitt 2.3 stellt die Definitionen der Begriffe *Architekturrahmenwerk* und *Referenzarchitektur* vor, wie sie in dieser Arbeit verwendet werden. Abschließend enthält Abschnitt 2.4 eine kurze Zusammenfassung des Kapitels.

2.1 Data Lakes

Um ein grundlegendes Verständnis für das Konzept des Data Lake zu gewinnen, wurde in vorangegangenen Arbeiten eine ausführliche Literaturübersicht erstellt [GGH+19a; GGH+20a]. Im Rahmen dieser wurde deutlich, dass es kein einheitliches Verständnis für die Natur des Data Lake gibt. Stattdessen

existieren zahlreiche Definitionen, die sich nicht nur unterscheiden, sondern teilweise sogar widersprechen. Dieser Abschnitt gibt einen Überblick über die Definitionen und Eigenschaften, die in der Literaturübersicht herausgearbeitet wurden, und leitet aus diesen eine Definition für Data Lakes ab, wie sie im Rest der Arbeit verwendet wird.

Zunächst soll jedoch kurz auf das Konzept der Dataspaces eingegangen werden. Dataspaces fanden erstmalig in einer Veröffentlichung von Franklin et al. im Jahre 2005 Erwähnung [FHM05]. Es handelt sich dabei um ein System, welches mehrere Datenquellen miteinander in Verbindung setzt, ungeachtet der Struktur der in ihnen gespeicherten Daten (z. B. relationale Daten, Texte oder Datenströme). Ein Dataspace modelliert dabei die Beziehungen zwischen den Datenquellen, etwa ob die Daten einer Quelle aus anderen Quellen abgeleitet wurden oder ob sich zwei Quellen auf das selbe physische Objekt beziehen. Damit ermöglicht ein Dataspace das Abfragen, Durchsuchen und Explorieren über Quellsystemgrenzen hinweg, ähnlich zu einem Data Lake. Im Gegensatz zum Data Lake werden die Daten allerdings nicht in ein zentrales System übertragen, sondern verbleiben in den Quellsystemen [JQ17]. Datenintegration hierdurch nur in ausgewählten Fällen statt [HFM06] und die Datenhoheit verbleibt bei den Eigentümern der Daten. Im Jahre 2016 wurde die International Data Spaces Association (IDSA)[1] gegründet, welche einen globalen Standard für Dataspaces über Unternehmens-- und Ländergrenzen hinaus erstellt. Im Folgenden wird auf eine nähere Betrachtung von Dataspaces verzichtet, da in dieser Arbeit die Zentralisierung von Daten innerhalb einer Organisation im Fokus steht. Insbesondere sollen Daten dabei unter eine einheitliche Hoheit gebracht werden, um Datenzugriff und Datengovernance zentral zu verwalten. Hierfür ist der Data Lake geeignet.

Das Konzept des Data Lake fand in einem Blogpost von James Dixon im Jahr 2010 zum ersten Mal Erwähnung [Dix10]. Diese erste Definition beschreibt den Data Lake als einen Datenspeicher für heterogene Daten aus einer einzelnen Quelle, die in ihrem Rohformat abgelegt werden. Anschließend können diese Daten durch eine Vielzahl von Nutzer*innen für verschiedenste analytische Anwendungsfälle verwendet werden. Dabei unterscheidet sich der Data Lake von einem traditionellen Data Warehouse mit anwendungsspezifischen Data Marts

[1] https://internationaldataspaces.org

insofern, als dass keinerlei Informationen durch vorgelagerte Transformationen, wie Extract-Transform-Load (ETL)-Prozesse, verloren gehen (z. B. durch Aggregationen oder das Weglassen von Attributen). Während diese Definition weitestgehend übernommen wurde, werden Data Lakes in späteren Arbeiten typischerweise aus mehreren Quellen befüllt, statt wie von Dixon definiert aus einer einzelnen (siehe z. B. [ML16; Fan15; OLe14]).

Für die Defintion des Data Lake werden Vergleiche zum bekannten Data-Warehouse-Ansatz in zahlreichen Veröffentlichungen gezogen (etwa in [OLe14; Mad15; TD16]). Tabelle 2.1 stellt die zentralen Unterschiede der beiden Konzepte dar, wie sie in der Literatur beschrieben werden. Dabei setzt sich der Data Lake insbesondere durch drei Eigenschaften vom Data Warehouse ab:

1. *Beliebig strukturierte Daten:* Während im Data Warehouse (vorrangig) strukturierte Daten verwaltet werden, enthält der Data Lake Daten beliebiger Struktur (strukturiert, semi-strukturiert, unstrukturiert).
2. *Schemalosigkeit:* Wie aus dem Datenverarbeitungsgrad, dem Schema und der Extraktion ersichtlich, werden beim Data Lake die Daten nicht vorverarbeitet oder in eine bestimmte Form gebracht. Stattdessen werden Daten roh abgespeichert und erst dann verarbeitet, wenn sie verwendet werden sollen (Schema-on-read). Dies unterscheidet sich grundlegend vom Ansatz des Data Warehouse, wo Daten bei der Aufnahme bereits für eine bestimmte Nutzung aufbereitet werden (Schema-on-write).
3. *Flexible Nutzung:* Eng verbunden mit der Schemalosigkeit ist auch die dritte Eigenschaft des Data Lake, welche auf den Kriterien Datennutzung und Analysen beruht. Da Daten in ihrem Rohformat im Data Lake gespeichert werden, gehen keinerlei Informationen verloren. So können die Daten flexibel für Anwendungsfälle verwendet werden, die zum Zeitpunkt ihrer Aufnahme in den Data Lake noch nicht feststanden. Auch werden im Data Lake Daten gespeichert, deren späterer Nutzen noch vollkommen unbekannt ist. Im Data Warehouse dagegen werden nur solche Daten aufgenommen, welche auch für einen vordefinierten Anwendungsfall benötigt werden. Durch anwendungsfallspezifische Aufbereitungen, wie beispielsweise Datenbereinigung, gehen hier potentiell Informationen

Kriterium	Data Warehouse	Data Lake
Struktur der Daten	**Strukturiert**	**Beliebig**
Datenverarbeitungsgrad	Verarbeitet	Roh
Schema	**Schema-On-Write**	**Schema-On-Read**
Extraktion	ETL	EL(T)
Datennutzung	Bekannt	Unbekannt
Analysen	**Vordefiniert**	**Flexibel**
Speicherkosten	Teuer	Billig
Änderbarkeit	Niedrig	Hoch
Datenaufbereitung	IT	Self-Service

Tabelle 2.1: Gegenüberstellung von Data Warehouse und Data Lake (basierend auf [OLe14; Mad15; TD16])

verloren. Die Daten können dadurch nicht so flexibel verwendet werden, wie das im Data Lake der Fall ist.

Obwohl das Konzept des Data Lake oft dem des Data Warehouse abgrenzend gegenübergestellt wird, schließen sich die beiden Ansätze keineswegs gegenseitig aus. Dixon selbst hält in einem zweiten Blogpost zu Data Lakes fest, dass diese weder Data Warehouses noch Data Marts ersetzen [Dix14]. Zahlreiche weitere Arbeiten unterstützen diese Ansicht, indem sie Data Lake und Data Warehouse miteinander kombinieren. So dient der Data Lake beispielsweise als Quelle für das Data Warehouse (z. B. in [SM14; MB17; Fan15]) oder der Data Lake selbst umfasst Data-Warehouse-Funktionalitäten (z. B. in [ZDB+15; Wel20]). Im ersten Fall dient der Data Lake lediglich für explorative und fortgeschrittene Analysen, während Reporting und Online Analytical Processing (OLAP) im Data Warehouse ausgeführt werden [SM14]. Im zweiten Fall sind Reporting und OLAP ebenfalls Teil der auf dem Data Lake ausgeführten Analysen.

Die oben genannten drei Eigenschaften sind zentral für jede Data-Lake-Definition. Darüber hinaus fügen verschiedene Definitionen weitere Eigenschaften hinzu. So sehen manche Data-Lake-Konzepte nicht nur die Speicherung von Rohdaten, sondern auch vorverarbeiteten Daten vor, um die Effizienz von

Anfragen zu erhöhen [TSRC15; Mad15]. Viele Definitionen fügen darüber hinaus die Art des Speichers hinzu, auf der ein Data Lake aufgebaut werden sollte [TD16; Mat17]. Dieser Speicher soll kostengünstig sein, um große Mengen von Daten vorhalten zu können [TD16]. Zudem schlagen verschiedenste Arbeiten vor, den Data Lake auf mehreren unterschiedlichen Speichertechnologien aufzubauen, um die Daten dort zu verwalten, wo ihre Eigenschaften und Nutzung bestmöglich unterstützt werden [Dix14; GH19]. In einigen Fällen wird der Data Lake direkt mit Apache Hadoop[2] und dem Hadoop Distributed File System (HDFS) assoziiert [Mat17; ML16]. Bei Hadoop handelt es sich allerdings um nur eine von vielen Möglichkeiten, einen Data Lake zu realisieren [Mat17; ML16]. Darum werden konkrete Speicherinstanzen aus der Definitionen für Data Lakes ausgeschlossen. Data Lakes können on-premise, in der Cloud oder als Hybride aufgebaut werden [Loc16].

Neben diesen Eigenschaften, die durch die meisten Data-Lake-Definitionen unterstützt werden, gibt es einige Punkte, in denen sich existierende Arbeiten uneinig sind oder sich sogar widersprechen. Diese Punkte betreffen die Rolle des Data Lake, die Nutzergruppen des Data Lake, sowie das Metadatenmanagement und Governance im Data Lake. Die folgenden Absätze gehen kurz auf die verschiedenen existierenden Ausprägungen dieser Punkte ein.

Rolle des Data Lake. Aus der ersten Data-Lake-Nennung von Dixon [Dix10] wird nicht ersichtlich, welche Rolle dem Data Lake ursprünglich zugedacht war. Es finden sich zwei unterschiedliche Ansätze in der Literatur, die sich gegenüberstehen: 1) der Data Lake als reiner Datenspeicher (siehe z. B. [Mat17]) und 2) der Data Lake als Datenmanagementplattform (siehe z. B. [Mad15]). In Fall 1), Data Lake als reiner Datenspeicher, besteht die Hauptaufgabe des Data Lake darin, Rohdaten aufzunehmen und direkt für Analysen zugreifbar zu machen. Im Gegensatz dazu übernimmt der Data Lake in Fall 2), Data Lake als Datenmanagementplattform, Aufgaben zur Datenaufbereitung. Die Daten stehen hier nicht nur roh, sondern auch integriert, qualitätsgesichert und vorverarbeitet zur Verfügung. So werden vordefinierte analytische Anwendungsfälle durch den Data Lake gezielt unterstützt, während dennoch explorative Analysen auf den Rohdaten möglich sind. Da insbesondere neuere

[2] https://hadoop.apache.org/

Definitionen zu Fall 2) tendieren (siehe z. B. [Sha18; RZ19]), wird der Data Lake auch in dieser Arbeit als Datenmanagementplattform aufgefasst.

Nutzergruppen des Data Lake. Auch bei den Nutzergruppen, die auf den Data Lake zugreifen, sind sich existierende Definitionen uneinig. Das Spektrum der genannten Nutzergruppen reicht hier von 1) Data Scientists als Datenanalyseexpert*innen (siehe z. B. [ML16]) bis zu 2) einer weiten Vielfalt an Nutzer*innen mit unterschiedlichstem Hintergrund (siehe etwa [Fan15]). Data Scientists verfügen über tiefes Wissen im Bereich der Statistik, der fortgeschrittenen Analysen, und der Datenaufbereitung. Daher brauchen sie wenig Unterstützung, um erfolgreich mit Daten zu arbeiten. Allerdings liegt ihre Expertise in der Datenverarbeitung und nicht im Domänenwissen. In Fall 2) dagegen sind nicht nur Data Scientists an der Nutzung des Data Lake beteiligt, sondern auch Nutzer*innen mit weniger Erfahrung in der Datenanalyse. Hier greifen auch Domänenexpert*innen auf Daten zu, die ein sehr tiefes Wissen über die Daten und Anwendungsfälle selbst haben, aber eher selten mit Statistik und fortgeschrittenen Analysen arbeiten. Zudem nutzen in diesem Fall auch Geschäftsnutzer*innen die Daten des Data Lake, um darauf Berichte zu erstellen und OLAP auszuführen. Diese Nutzer*innen haben nur geringe Erfahrung in der Verarbeitung von Daten, da ihre Expertise in der Planung und Überwachung von Geschäftsprozessen liegt. Für unsere Data-Lake-Definition werden zunächst beide Ausprägungen zugelassen, da die Nutzergruppen stark von der Umgebung abhängen, in der ein Data Lake eingesetzt wird.

Metadatenmanagement und Governance im Data Lake. Der letzte Punkt, in dem die Literatur zwiegespalten ist, ist das Vorhandensein von Metadaten und Governance im Data Lake. Der Begriff „Governance“ bezeichnet hierbei die Menge interner Prozesse und Regularien, welche für die Wahrung von etwa Datenqualität, Datensicherheit oder rechtlicher und wirtschaftlicher Interessen eingesetzt werden, sowie die Kontrolle der Einhaltung dieser [AC15]. Insbesondere frühe Arbeiten sprechen dem Data Lake jegliche Art von Metadatenmanagement und Governance ab (siehe z. B. [Gar14; Fan15]). In diesem Fall droht der Data Lake, sich in einen so genannten *Data Swamp* zu verwandeln [CSN+14]. In einem solchen Data Swamp gibt es keinerlei Übersicht oder Regulation der Daten, sodass die vorliegenden Daten nicht vertrauenswürdig oder schlichtweg nicht auffindbar sind. Dadurch werden sie für Analysen

unbrauchbar [Mat17]. Neuere Data-Lake-Definitionen beschreiben Metadatenmanagement und Governance daher als zentrale Bestandteile eines Data Lake, ohne die eine gewinnbringende Nutzung unmöglich ist (siehe z. B. [RZ19]). Manche Arbeiten bezeichnen einen Data Lake, der Governance umfasst, auch als *Data Reservoir* (siehe z. B. [CSN+14; Top16]). Da allerdings auch Data-Lake-Definitionen Governance umfassen und die Unterschiede zwischen den beiden Konzepten nicht diskutiert werden bleibt unklar, inwiefern sich ein solches Data Reservoir von einem Data Lake unterscheidet. Im Rest dieser Arbeit wird daher auf diese Unterscheidung verzichtet. Da neuere Definitionen Metadatenmanagement und Governance als Teil des Data Lake sehen, sind diese auch Teil der Definition.

Abbildung 2.1 fasst die in der Literatur genannten Eigenschaften eines Data Lake und ihre Ausprägungen zusammen. Aus diesen formuliert dieser Abschnitt folgende Data-Lake-Definition, wie sie im Rest der Arbeit verwendet wird:

*Bei einem Data Lake handelt es sich um eine Datenmanagementplattform, die auf skalierbarem und kostengünstigen Speicher aufgebaut ist. In einem Data Lake werden Daten heterogener Struktur in ihrem Rohformat abgespeichert, sodass sie auch ohne vordefinierte Anwendungsfälle verwendet werden können. Zudem können auch vorverarbeitete Daten gespeichert werden, um die Effizienz bestimmter Anwendungsfälle zu erhöhen. Die Daten können dabei aus beliebig vielen Quellen stammen. Ein Data Lake umfasst typischerweise mehrere Speichersysteme, sodass Daten dort verwaltet werden können, wo ihre Eigenschaften bestmöglich unterstützt werden. Dabei kann ein Data Lake sowohl on-premise, in der Cloud oder als Hybrid aufgebaut sein. Es gibt sowohl Data Lakes für Datenanalyseexpert*innen, als auch für ein breites Spektrum an Nutzer*innen mit unterschiedlicher Erfahrung in der Datenanalyse. Metadatenmanagement und Governance sind von zentraler Bedeutung für die Funktion eines Data Lake und darum wichtige Bestandteile dessen.*

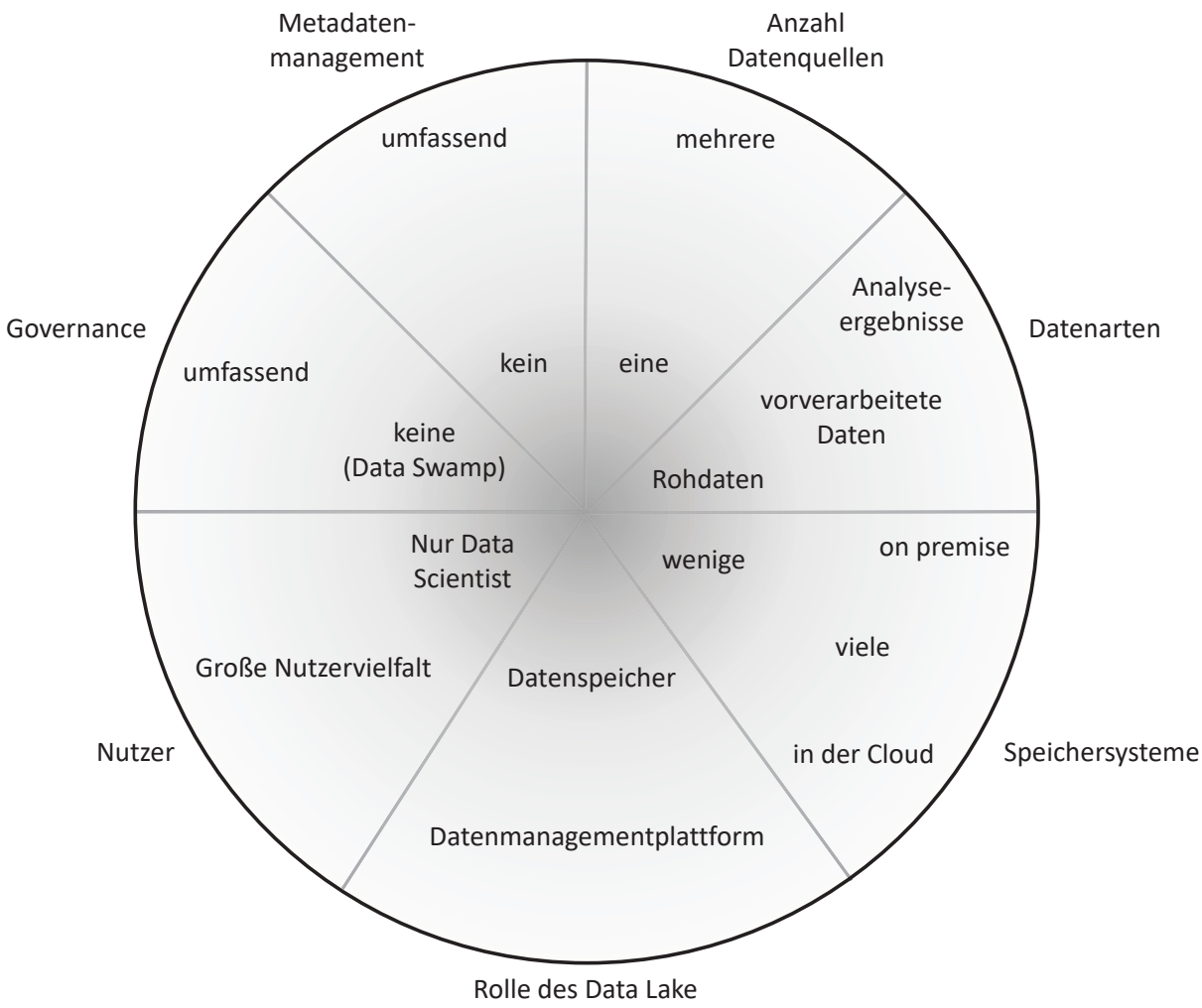


Abbildung 2.1: Übersicht über Data-Lake-Charakteristika in Data-Lake-Definitionen [GGH+20a]

2.2 Big-Data-Technologien

Das Konzept des Data Lake ist vorrangig auf die Speicherung und Verwaltung von *Big Data* ausgelegt [ML16]. Zwar können auch andere Daten im Data Lake abgelegt werden, der Fokus liegt allerdings auf voluminösen und heterogenen Daten. Daher eignen sich Technologien aus dem Bereich Big Data zur Realisierung des Data Lake. In diesem Abschnitt wird zunächst auf existierende Speichertechnologien für Big Data eingegangen (Abschnitt 2.2.1) und anschließend auf Möglichkeiten zur Verarbeitung von Big Data (Abschnitt 2.2.2). Dabei wird insbesondere auf Stapel-, Datenstrom- und hybride Verarbeitung eingegangen.

2.2.1 Speichertechnologien für Big Data

Betrachtet man traditionelle Data Warehouses, so sind diese typischerweise auf *relationalem Speicher* aufgebaut. Relationale Datenbanken bieten viele Eigenschaften, die bei der Verwaltung von Daten vorteilhaft sind, wie beispielsweise geringe Redundanz und ACID-Eigenschaften (Atomicity, Consistency, Isolation, Durability) für Transaktionen [VBH+15; SF13]. Allerdings können relationale Datenbankmanagementsysteme (RDBMS) allein nicht die Herausforderungen von Big Data meistern. Die hohe Heterogenität der Daten, insbesondere ihrer Struktur (strukturiert, semi-strukturiert, unstrukturiert) fordert eine Flexibilität, die RDBMS oftmals nicht leisten können [VBH+15]. Auch das Volumen der Daten schafft die Notwendigkeit für neue Ansätze [AJO+13].

Um diese Problematiken zu adressieren, wurden die so genannten *NoSQL-Datenbanken* entwickelt. Die Bedeutung von NoSQL wandelte sich über die Zeit hinweg: Während diese Art der Datenbanken bei ihrer Entstehung keinerlei SQL umfassten, steht NoSQL heute für *Not Only SQL* [SF13]. Der Fokus von NoSQL-Datenbanken unterscheidet sich von dem relationaler Datenbanken. Statt auf Konsistenz und geringer Redundanz liegt er bei NoSQL-Datenbanken auf Performanz und Skalierbarkeit [HAR16; VBH+15]. Dafür nehmen NoSQL-Datenbanken auch Redundanz in Kauf, wenn dadurch zusammengehörende Daten effizienter abgefragt werden können [KR13].

Über ihren Fokus hinaus haben alle NoSQL-Datenbanken weitere Eigenschaften gemein [SF13]. Dabei heben sie sich insbesondere durch ihre Schemalosigkeit von relationalen Datenbanken ab. Schemalos bedeutet dabei nicht die komplette Abwesenheit jeglichen Schemas [Sti14]. Stattdessen bezieht es sich eher auf ein festes vordefiniertes Schemas, wie es in relationalen Datenbanken vorkommt. Schemata in NoSQL-Datenbanken sind dagegen flexibel. So kann jedes Tupel, das in einer NoSQL-Datenbank gespeichert wird, eine andere Menge an Attributen besitzen als alle anderen Tupel in derselben Datenbank. Dadurch ist es möglich, neben strukturierten Daten auch semi-strukturierte Daten effizient zu verwalten.

Bei NoSQL-Datenbanken wird typischerweise zwischen drei Typen unterschieden [SF13; Cat11; BC13]:

1. *Key-Value-Datenbanken (engl. Key-Value Stores)*: Jedes Datenobjekt (value) wird mit einem eindeutigen Schlüssel (key) versehen, durch den es zugegriffen werden kann. Das Datum kann dabei einfach (z. B. eine Zahl oder ein String) oder komplex sein (z. B. ein zusammengesetztes Objekt).
2. *Dokumentenorientierte Datenbanken (engl. Document Stores)*: Mehrere Key-Value-Paare werden zu einem Dokument verknüpft, welches durch eine ID identifiziert wird. Dabei können Key-Value-Paare verschachtelt werden. Typisch sind hier Formate wie JavaScript Object Notation (JSON) oder Extensible Markup Language (XML).
3. *Spaltenorientierte Datenbanken (engl. Wide-Column Stores, Column-Family Stores)*: Ähnlich zu relationalen Datenbanken werden auch in spaltenorientierten Datenbanken die Daten in Tabellen abgelegt. Anstatt die Daten allerdings zeilenweise zuzugreifen, findet der Zugriff über Spalten statt. Dies erhöht die Effizienz von Leseoperationen, insbesondere bei Spalten, die nur gering befüllt sind. Unter Spaltenfamilien (engl. Column Families) versteht man eine Gruppierung von Spalten, die oft zusammen abgerufen werden.

Zusätzlich werden auch Graphdatenbanken manchmal als NoSQL-Datenbank gezählt (z. B. in [SF13]):

4. *Graphdatenbanken (engl. Graph Databases)*: Graphdatenbanken speichern Daten als Knoten, während Beziehungen zwischen Daten als Kanten dargestellt werden. Bestimmte Graphformate, wie der Labeled Property Graph, erlauben zudem die Speicherung von Key-Value-Paaren an Knoten und Kanten, um Attribute zu hinterlegen [RWE15].

Diese verschiedenen NoSQL-Datenbanktypen sind jeweils auf bestimmte Anwendungsfälle und Daten ausgelegt, bei denen sie ihre Vorteile besonders gut ausspielen können [SHJ17], z. B. ermöglichen Graphdatenbanken die Nutzung von Graphalgorithmen auf stark vernetzten Daten.

NoSQL-Datenbanken bieten allerdings nicht nur Vorteile gegenüber relationaler Datenbanken. Um die Performanz zu erhöhen, verzichten sie auf einige Eigenschaften, die in relationalen Datenbanken für höhere Datenqualität und bessere Governance sorgen. So fehlt z. B. in vielen Fällen die Unterstützung

von Fremdschlüsseln (z. B. bei der spaltenorientierten Datenbank Cassandra[3] oder bei der dokumentenorientierten Datenbank MongoDB[4]). Auch die Unterstützung von Transaktionen ist nicht immer gegeben (z. B. bei Cassandra).

Neben SQL- und NoSQL-Datenbanken gibt es auch *Hadoop*[5] zu Speicherung und Verwaltung von Daten. Darüber hinaus bietet Hadoop mit MapReduce [DG08] auch Möglichkeiten zur Verarbeitung von Daten an. Zur Datenspeicherung umfasst Hadoop das HDFS. Dabei handelt es sich, wie der Name schon sagt, um ein verteiltes und dadurch skalierbares Dateisystem, in dem Daten aller Strukturen gespeichert werden können [CY15]. Die im HDFS abgelegten Daten können anschließend mit verschiedensten Tools zugegriffen und verarbeitet werden. Zudem bietet das HDFS Datenreplikation, wodurch es Datenverlust vorbeugt.

Da Data Lakes große Mengen heterogener Daten verwalten, eignen sich sowohl NoSQL-Datenbanken als auch das HDFS sehr gut für ihre Umsetzung. Allerdings können auch die Stärken von SQL-Datenbanken, wie Fremdschlüsselbedingungen, im Data Lake von Vorteil sein. Wie in 2.1 beschrieben, kann ein Data Lake auch auf mehreren Speichersystemen aufgesetzt werden. Somit ist auch eine Mischung von SQL-, NoSQL-Datenbanken und HDFS denkbar.

2.2.2 Möglichkeiten zur Verarbeitung von Big Data

Um den potentiellen Wert aus den voluminösen und Daten zu extrahieren, müssen diese verarbeitet werden. Dabei gibt es drei Varianten, Big Data zu verarbeiten [CY15]:

1. *Stapelverarbeitung (engl. Batch Processing)*: Bei der Stapelverarbeitung liegen alle zu verarbeitenden Daten beim Start der Verarbeitung vor. Das bedeutet, dass alle Daten zusammen verarbeitet werden und nach dem Start der Verarbeitung keine neuen Daten mehr dazukommen (ruhende Daten). Auch die Verarbeitungslogik ändert sich nicht mehr, sobald sie einmal angestoßen wurde. Sie kann nur für eine komplett neue Verarbeitung angepasst werden. Da alle Daten auf einmal verarbeitet werden, hängt die Verarbeitungsdauer stark von der Menge der zu verarbeitenden

[3] https://cassandra.apache.org/
[4] https://www.mongodb.com/
[5] https://hadoop.apache.org/

Daten ab. Dafür sind die Ergebnisse der Stapelverarbeitung typischerweise sehr verlässlich, da eine große Menge Daten in ihre Erzeugung einfließt, anstatt nur einzelne Datenpunkte wie bei der Datenstromverarbeitung [CY15].

2. *Datenstromverarbeitung (engl. Stream Processing)*: Im Gegensatz zur Stapelverarbeitung werden die zu verarbeitenden Daten kontinuierlich über einen Datenstrom angeliefert. Das bedeutet, dass die Daten zum Start der Verarbeitung nicht vollständig vorliegen. Je nach verwendetem Verarbeitungssystem werden die Daten direkt verarbeitet, wenn sie in der Verarbeitung eintreffen, oder sie werden mehrere Millisekunden lang gesammelt, bevor sie in Mikrostapeln (engl. Micro Batches) verarbeitet werden. Dadurch ist die Verarbeitungszeit unabhängig von der Gesamtmenge der Daten. Die Verarbeitungslogik kann jederzeit angepasst werden, auch, wenn die ersten Daten bereits verarbeitet wurden. Alle folgenden Daten werden in diesem Fall mit der neuen Logik verarbeitet. Da die Verarbeitung allerdings immer nur auf einer sehr begrenzten Menge an Daten ausgeführt wird, manchmal sogar auf einzelnen Datenpunkten, sind Aussagen über die Gesamtheit der Daten deutlich ungenauer als bei der Stapelverarbeitung.

3. *Hybride Verarbeitung (engl. Hybrid Processing)*: Nicht immer reicht eine reine Stapel- oder Datenstromverarbeitung aus, um allen Anforderungen an die Verarbeitung gerecht zu werden. So gibt es beispielsweise Anwendungen, die eine echtzeitnahe Verarbeitung benötigen, in denen aber gleichzeitig auch große Mengen historischer Daten in Betracht gezogen werden müssen [MW15]. Ein Beispiel hierfür ist ein aktueller Zähler für die Anzahl der eindeutigen Nutzer*innen einer Webseite, d. h. kein Nutzer*in wird doppelt gezählt. Hierfür müssen die Aufrufe der Webseite in Echtzeit verarbeitet werden und gleichzeitig die Nutzer-ID mit den bisher gezählten Nutzer*innen verglichen werden. In diesen Fällen wird auf die hybride Verarbeitung zurückgegriffen, welche Stapel- und Datenstromverarbeitung kombiniert, um die gewünschten Ergebnisse zu erzielen.

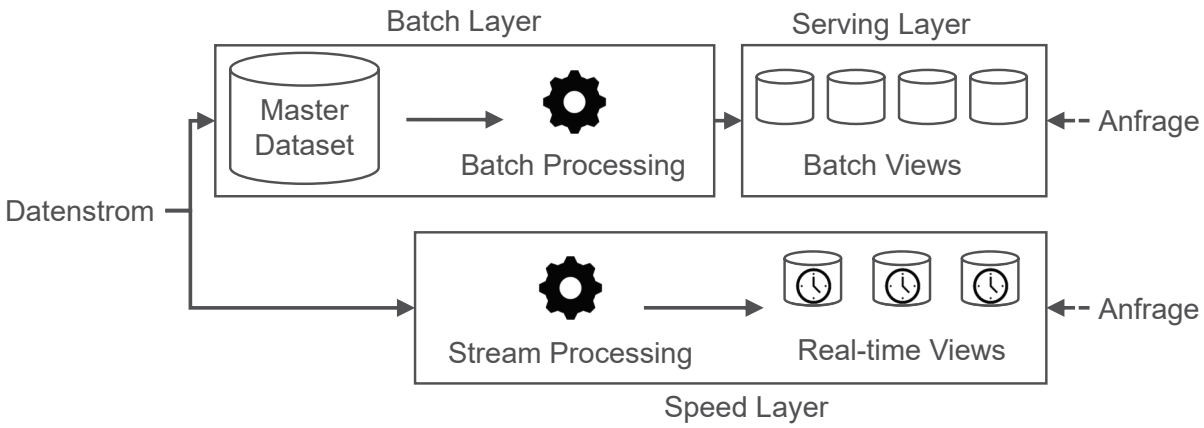


Abbildung 2.2: Die Lambda-Architektur (basierend auf [MW15])

Um Big Data zu verarbeiten, gibt es verschiedene Architekturen, die sich der unterschiedlichen Verarbeitungsmodi bedienen. Die älteste dieser Architekturen ist die *Lambda-Architektur* [MW15], eine hybride Verarbeitungsarchitektur, welche sowohl Stapel- als auch Datenstromverarbeitung umfasst. Ihr Aufbau ist in Abbildung 2.2 dargestellt. Daten werden als Datenstrom aufgenommen und sowohl an den Batch Layer als auch an den Speed Layer weitergeleitet. Im Batch Layer werden die Daten permanent in einem Master Dataset gespeichert, an das Daten nur angehängt werden können, und in regelmäßigen Intervallen per Stapelverarbeitung verarbeitet. Die Ergebnisse dieser Verarbeitung werden im Serving Layer als Batch Views für Abfragen zur Verfügung gestellt. Parallel dazu werden die Daten per Datenstromverarbeitung im Speed Layer in Echtzeit verarbeitet und über die Real-time Views zur Verfügung gestellt. Anfragen werden auf der Kombination aus Batch Views und Real-time Views ausgeführt. Die Lambda-Architektur umgeht somit das Problem, dass durch die zeitintensive Stapelverarbeitung die aktuellsten Daten nicht in den Ergebnismengen vorhanden sind, da sich diese in den Real-time Views finden lassen.

Die Verwendung der getrennten Batch Layer und Speed Layer führt bei der Lambda-Architektur allerdings dazu, dass zwei separate Systeme entwickelt und gewartet werden müssen. Dieses Problem wird durch die *Kappa-Architektur* adressiert, welche eine rein datenstrombasierte Alternative zur Lambda-Architektur darstellt [Kre14]. Abbildung 2.3 stellt den Aufbau der Kappa-Architektur dar. Dabei werden Daten, die als Datenstrom ankommen, zunächst in einen Puffer geschrieben, der als Master Dataset fungiert. Die

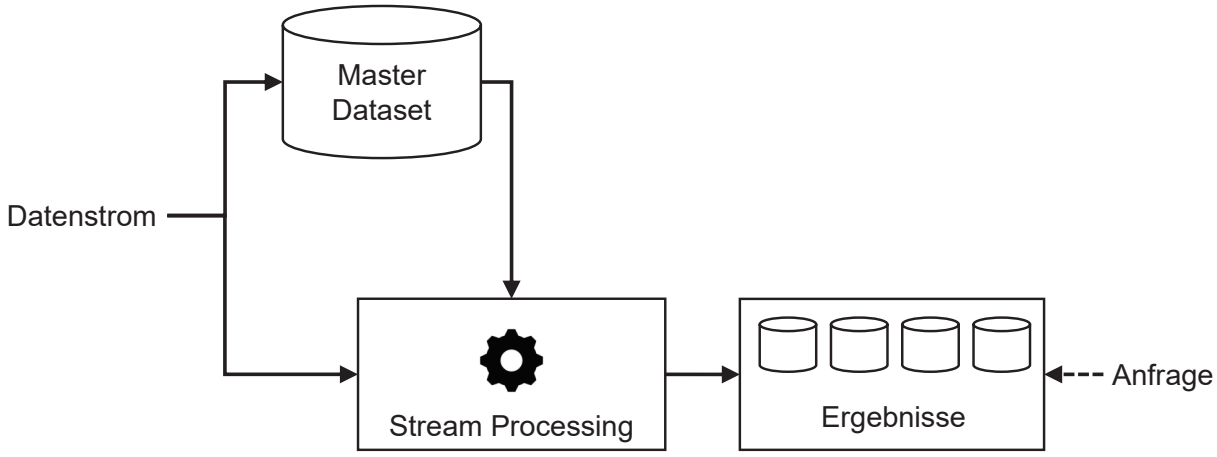


Abbildung 2.3: Die Kappa-Architektur(basierend auf [Kre14])

Daten werden rein per Datenstromverarbeitung verarbeitet. Dabei können mehrere Verarbeitungsprozesse in der Stream-Processing-Komponente parallel laufen. Sollte es nötig sein, die historischen Daten erneut zu verarbeiten, so können diese aus dem Master Dataset in einen Datenstrom umgewandelt und erneut in die Datenstromverarbeitung geschickt werden.

Zwar besteht die Kappa-Architektur nur aus einem einzelnen Datenstromverarbeitungssystem, was die Implementierung und Wartung vereinfacht, es fehlt aber die Genauigkeit der Stapelverarbeitung. Zudem sieht weder die Lambda- noch die Kappa-Architektur die Möglichkeit vor, Ergebnisse aus der Stapelverarbeitung in der Datenstromverarbeitung anzuwenden, z. B. ein Machine-Learning-Modell, welches eintreffende Daten klassifiziert. Darum wurde in vorangegangenen Arbeiten eine hybride Datenverarbeitungsarchitektur entwickelt, welche sowohl die Lambda- als auch die Kappa-Architektur emulieren kann und zudem weitere Anwendungsfälle unterstützt. Diese Datenverarbeitungsarchitektur heißt (Hybrid Processing Architecture for Big Data (BRAID) [GSSM18]. Der Aufbau von BRAID ist in Abbildung 2.4 dargestellt. Auch hier werden die Daten zunächst in einem Master Dataset in einem geteilten Speicher gespeichert, welches sowohl per Stapel- als auch per Datenstromverarbeitung verarbeitet werden kann. Für die Echtzeitverarbeitung werden die Daten direkt in einen Puffer geschrieben, welcher dann echtzeitnah in der Datenstromverarbeitung verarbeitet wird. Zudem können die Ergebnisse der Verarbeitungen ebenfalls gespeichert und für weitere Verarbeitungsprozesse erneut geladen werden. So kann z. B. ein in der Stapelverarbeitung

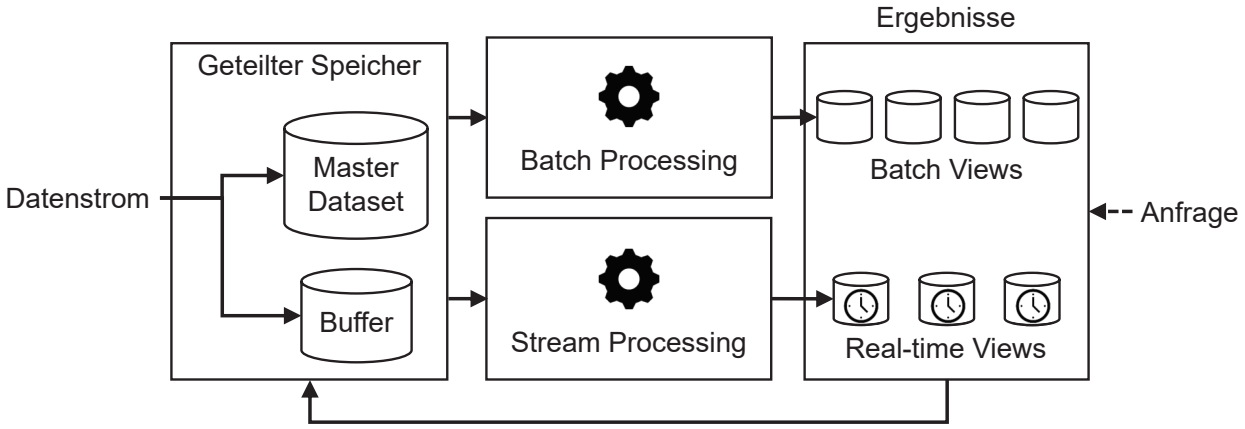


Abbildung 2.4: BRAID [GSSM18]

erlerntes Machine-Learning-Modell auf Daten im Datenstrom angewandt werden [SGS19]. Die Daten können in einem gemeinsamen Ergebnisspeicher abgefragt werden.

Da in einem Data Lake verschiedenste Arten von Daten verwaltet werden müssen, ist die Wahl einer passenden Verarbeitungsmöglichkeit von hoher Wichtigkeit. Insbesondere wenn sowohl ruhende Daten als auch Datenströme verwendet werden, kann der Data Lake stark von der Nutzung einer hybriden Verarbeitungsarchitektur profitieren (siehe z. B. [MB17]).

2.3 Definition Architekturrahmenwerk und Referenzarchitektur

Die im Rahmen dieser Arbeit entwickelten Konzepte umfassen sowohl ein Architekturrahmenwerk (engl. Architecture Framework) als auch eine Referenzarchitektur (engl. Reference Architecture) für Data Lakes. Dieser Abschnitt stellt die im Rest der Arbeit verwendeten Definitionen beider Begriffe vor und grenzt diese voneinander ab.

Architekturrahmenwerk. Der Fokus eines Architekturrahmenwerks ist die Gesamtheit eines Produkts, z. B. eines physischen Produkts oder auch eines Softwaresystems [Zac87]. Dabei gilt es, verschiedene Beschreibungen und Perspektiven desselben Produkts miteinander zu verbinden und in Kontext zu setzen [Zac87]. Jede dieser Perspektiven wird wiederum durch eine eigenständige Architektur umgesetzt. Die Aufgabe des Rahmenwerks ist es, die

verschiedenen Perspektiven auf das Produkt zu definieren, ihre Abhängigkeiten herauszuarbeiten und jeder Perspektive eine Menge an möglichen Architekturen zuzuweisen [SZ92]. Wird jede Perspektive im Architekturrahmenwerk durch eine konkrete Architektur instantiiert, ergibt sich eine Gesamtarchitektur für das gewünschte Produkt. Beispiele für Architekturrahmenwerke sind The Open Group Architecture Framework (TOGAF)[6] oder Zachmans Rahmenwerk für Informationssystemarchitekturen [Zac87; SZ92].

Referenzarchitektur. Eine Referenzarchitektur fasst Erfolgsmethoden in einer Architektur zusammen. Ihr Fokus liegt dabei auf einzelnen Teilen eines Systems und deren Beziehungen untereinander [SSPW14]. Dabei ist es wichtig, dass eine Referenzarchitektur nicht zu jedem beliebigen Anwendungsfall passt. Vielmehr können aus einer Referenzarchitektur Architekturen definiert werden, die dann die Anforderungen eines konkreten Anwendungsfalls erfüllen können. Die Referenzarchitektur ist hierbei eine idealtypische Architektur, die als Grundlage für spezifischere Architekturen dient.

2.4 Zusammenfassung

In diesem Kapitel wurden die Grundlagen erklärt, die für das weitere Verständnis der Arbeit benötigt werden. Dabei wurde insbesondere auf Data Lakes, Big-Data-Technologien, sowie relevante Begriffsdefinitionen näher eingegangen. In der Literaturrecherche zu Data Lakes zeigte sich, dass sich existierende Definitionen teils stark unterscheiden oder sogar widersprechen. Für den Rest dieser Arbeit wurde darum eine Arbeitsdefinition für Data Lakes aufgestellt. Die Grundlagen zu Big-Data-Technologien betrachteten sowohl Speichertechnologien als auch Verarbeitungsmöglichkeiten für Big Data. Besonderes Augenmerk lag hierbei auf Verarbeitungsarchitekturen, wie Lambda, Kappa, oder BRAID. Abschließend wurden die Begriffe „Architekturrahmenwerk“ und „Referenzarchitektur“ definiert und voneinander abgegrenzt.

[6] https://www.opengroup.org/togaf

Kapitel 3

Anwendungsszenarien und Fallbeispiel

Genau wie die im Data Lake verwalteten Daten sind auch die möglichen Einsatzgebiete breit gefächert und heterogen. So können Data Lakes in verschiedensten Domänen Anwendung finden, beispielsweise in der Industrie (vgl. [GGH+20a]), der Gesundheitsfürsorge (vgl. [SGW+20]) oder auch der Forschung (vgl. [TSRC15]). Als *Anwendungsszenario* wird in dieser Arbeit hierbei eine Kombination aus Anwendungsdomäne (z. B. Industrie), verwalteten Daten (z. B. unstrukturierter Datenstrom von Überwachungskameras) und beabsichtigten Analysen (deskriptiv bis präskriptiv [Hag16]) bezeichnet. Dieses Kapitel gibt einen Überblick über verschiedene mögliche Anwendungsszenarien und deren Einflüsse auf die benötigte Data-Lake-Architektur. Dazu stellt Abschnitt 3.1 Anwendungsszenarien für Data Lakes aus der Literatur und Praxis vor. Anschließend beschreibt Abschnitt 3.2 ein praktisches Fallbeispiel aus der Industrie, welches in dieser Arbeit für die weitere Betrachtung von Data Lakes als fortlaufendes Beispiel fungiert. Das Kapitel wird in Abschnitt 3.3 zusammengefasst.

3.1 Anwendungsszenarien

Der *AIRPORTS Data Lake (AIRPORTS DL)* von Martínez-Prieto et al. [MBG+17] fokussiert sich auf die Speicherung und Auswertung von Flugdaten. Insbesondere werden hier Daten zu Fluglinien gesammelt und ausgewertet. Diese Daten werden direkt in den Flugzeugen erfasst und in den Data Lake übertragen. Sie umfassen beispielsweise die Position, Geschwindigkeit und Höhe des Flugzeugs, aber auch Informationen zum Transpondercode (Code zur eindeutigen Identifikation eines Flugzeugs) oder ob ein Notrufsignal gesendet wurde. Dazu werden weitere Datenquellen an den Data Lake angebunden, wie Wetterdaten, Flugpläne oder Daten zu Flughäfen. Ein Teil der Daten wird als Datenstrom in Echtzeit mithilfe von Apache Flume[1] in den Data Lake aufgenommen. Für die Aufnahme der verbleibenden Daten verwenden die Autoren unter anderem Apache Sqoop[2] und Hadoop-Befehle für eine stapelweise Aufnahme. Die Daten werden anschließend im Hadoop Distributed File System (HDFS)[3] gespeichert und mit MapReduce [DG08], Apache Pig[4] und Apache Hive[5] verarbeitet. Dabei werden auch die Daten, die als Datenstrom aufgenommen wurden, als Stapel weiterverarbeitet. Anschließend können diese Daten exploriert werden. Die Werkzeuge dafür reichen von Hadoop-Befehlen bis hin zur Verwendung von R Programmen[6]. Die Autoren gehen nicht näher auf die Art der geplanten Analysen auf diesen Daten ein. Die gegebenen Beispiele lassen allerdings auf vorwiegend deskriptive Analysen schließen.

Auch Google verwaltet eine Vielzahl von Datensätzen in einem Data Lake [HKN+16a; HKN+16b]. Zu den Daten selbst, deren Nutzung und dem Aufbau des Data Lake gibt es in den Veröffentlichungen allerdings keine Informationen. Dafür beschreiben die Autoren ein graphbasiertes Metadatenmanagement für diesen Data Lake. Da das Management von Metadaten jedoch nicht im Fokus dieser Arbeit liegt, wird an dieser Stelle nicht weiter darauf eingegangen.

[1] https://flume.apache.org/
[2] https://sqoop.apache.org/
[3] hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
[4] https://pig.apache.org/
[5] https://hive.apache.org/
[6] https://www.r-project.org/

Mehr Details zu den Daten, die in dem betrachteten Data Lake verwaltet werden, bietet die Beschreibung des IBM Research Data Lake [TSRC15]. Im Gegensatz zu AIRPORTS DL fokussiert sich dieser nicht auf eine spezifische Domäne, sondern verwaltet Daten für verschiedenste analytische Anwendungsfälle. So wird beispielsweise die Krebsforschung durch Textsuche in medizinischen Veröffentlichungen und Patenten unterstützt [TSRC15]. Die Daten sowie ihre Anwendung sind damit hochgradig heterogen. Zum technischen Aufbau liefert die Beschreibung jedoch wenig Information. Lediglich auf die Nutzung einer Cloud-Infrastruktur wird verwiesen. In einer anderen Veröffentlichung zu Data Lakes von IBM sprechen die Autoren von einer Vielzahl von Systemen, unter anderem Hadoop und DB2[7] [ZDB+15]. Es bleibt hierbei allerdings unklar, ob es sich bei dem dort beschriebenen Data Lake ebenfalls um den IBM Research Data Lake handelt oder um einen generischen Data-Lake-Entwurf.

Im Smart Grid Big Data Eco System [MM18] werden Daten zur intelligenten Stromerzeugung und -Verteilung in einem Data Lake verwaltet. Auf diesen Daten werden Data-Mining-Verfahren angewandt, z. B. Klassifikation oder auch Modelle zu Vorhersagen, um so die vorhandenen Stromnetze zu verbessern. Da die Daten in diesem Anwendungsszenario sowohl in großer Menge als auch schnell verarbeitet werden müssen, greifen die Autoren auf einen Data Lake auf Grundlage der Lambda-Architektur [MW15] zurück. Dabei findet die Stapelverarbeitung in der Lambda-Architektur in einem Data Lake basierend auf Hadoop[8] statt. Die Datenstromverarbeitung ist mit Apache Spark[9] umgesetzt. Die Abfrage der Daten erfolgt über Spark SQL, Hive[4] oder Impala[10]. All dies ist in einer Cloud-Umgebung realisiert. Besonders wichtig in diesem Anwendungsszenario ist zudem eine Feedback-Schleife, durch die Erkenntnisse der Analysen wieder in die Daten selbst und die Netzstruktur einfließen.

Die Betrachtung dieser sehr unterschiedlichen Anwendungsszenarien zeigt die Varianz der Anwendungsgebiete für Data Lakes auf. Insbesondere zeigt sich, dass es je nach Anwendungsszenario unterschiedliche Arten gibt, den Data Lake aufzubauen (z. B. mehrere Systeme oder ein einzelnes System, on-premise oder in der Cloud). Um daher eine geeignete Umsetzung für ein konkretes

[7] https://www.ibm.com/de-de/products/db2-database
[8] http://hadoop.apache.org/
[9] https://spark.apache.org/
[10] https://impala.apache.org/

Anwendungsszenario zu konzipieren, müssen zunächst die Anforderungen und Voraussetzungen dieses Szenarios bekannt sein.

3.2 Fallbeispiel

Für die Erreichung der Forschungsziele Z1, Z2 und Z3 dient im Folgenden das Fallbeispiel eines großen, international agierenden Herstellers als Basis. Die Betrachtung und Diskussion des Fallbeispiels sind Teil einer vorangegangenen Veröffentlichung der Autorin [GGH+20a].

Das Geschäft des betrachteten Herstellers ist sehr vielfältig und umfasst, unter anderem, Zulieferteile, Werkzeuge, sowie Haushaltsgeräte. Diese Produkte werden sowohl in Massen- als auch Einzelteilproduktion hergestellt. Die digitalen Systeme, die dabei zum Einsatz kommen, sind ähnlich vielfältig. Sie reichen von Enterprise Resource Planning (ERP)-Systemen und Manufacturing-Execution-Systeme (MESs) über Customer Relationship Management (CRM)-Systeme bis hin zu Computer Aided Design (CAD)- und Product Lifecycle Management (PLM)-Systemen verschiedenster Entwickler. Dieser Hersteller eignet sich daher für eine repräsentative Betrachtung, da eine solche heterogene System- und Datenlandschaft typisch für große Industrieunternehmen ist [OLe14].

Ein großes Ziel des Herstellers ist es dabei, die Transformation hin zur Industrie 4.0 [Grö18] zu vollziehen. Durch die Umsetzung dieser Initiative sollen Prozesse optimiert und die Wettbewerbsfähigkeit erhöht werden. Dazu setzt der Hersteller insbesondere auf den Einsatz von Sensoren in Produktion und Endprodukten, die Erfassung von kunden- und produktbezogenen Daten aus sozialen Netzwerken, aber auch auf die Integration der Daten aus den unterschiedlichen Systemen. Darüber hinaus werden im gesamten Unternehmen Datenanalyseprojekte ins Leben gerufen, welche sowohl traditionelles Reporting als auch fortgeschrittene Analysen umfassen. Die Daten für diese Analysen werden in einem unternehmensweiten Data Lake verwaltet.

Dieser Abschnitt detailliert die verschiedenen Facetten des Fallbeispiels. Dazu beschreibt Abschnitt 3.2.1 die zugrundeliegenden Daten des Fallbeispiels. Abschnitt 3.2.2 gibt einen Überblick über die Analysen, die auf diesen Daten ausgeführt werden, und die Nutzergruppen, die auf die Daten zugreifen.

3.2.1 Daten

Nicht nur die Geschäftsbereiche und Systemlandschaft des Herstellers sind heterogen, auch die verwalteten Daten weisen eine Vielzahl an Eigenschaften auf. Diese Vielzahl beschränkt sich nicht nur auf die Struktur der Daten (z. B. strukturiert, semi-strukturiert, unstrukturiert). Stattdessen unterscheiden sich die Daten auch in ihrer Qualität, ihrem Volumen, wie schützenswert sie sind oder weiteren Eigenschaften. So benötigen beispielsweise personenbezogene Daten eine deutlich strengere Qualitätssicherung und Zugriffsregelung als andere Daten.

Bei der Betrachtung verschiedener Datenanalyseprojekte des Herstellers wurden im Rahmen dieser Arbeit drei Kategorien von Daten identifiziert, die in analytischen Anwendungen verwendet werden:

- *Stammdaten* bezeichnen Daten zu Geschäftsobjekten, die für Geschäftsprozesse von zentraler Bedeutung sind, beispielsweise Kunden, Mitarbeiter oder auch Produkte [SME14; Los09]. Diese Daten sind typischerweise, so auch im Fall des betrachteten Unternehmens, strukturiert und eher statischer Natur, d. h., die Daten ändern sich nur selten. Durch das Verwalten dieser Daten in einem unternehmensweiten Data Lake können sie mit anderen Datenarten verknüpft werden. So können z. B. Produktdaten direkt mit Fotos und Videos zu Produktdefekten verbunden werden. Stammdaten haben für Unternehmen einen hohen Wert und ihre Korrektheit ist von großer Bedeutung. Sie sind darum als schützenswert einzustufen und müssen entsprechend verwaltet werden. Auch muss eine hohe Datenqualität hergestellt und erhalten werden. Darum ist eine weitreichende Governance notwendig, um die Daten gegen unautorisierten Zugriff, Änderungen und andere Betrugsversuche abzusichern.
- *Transaktionale Daten*, häufig auch als Bewegungsdaten bezeichnet, werden mit Geschäftstransaktionen und -prozessen innerhalb des Unternehmens assoziiert. Sie sind an einen bestimmten Ereigniszeitpunkt gebunden, wie beispielsweise den Eingang einer Kundenbeschwerde [SME14]. Nach ihrer Erfassung werden sie typischerweise nicht mehr verändert [SME14]. Transaktionale Daten kommen in unterschiedlichsten Strukturen vor, von strukturiert bis unstrukturiert. Strukturierte trans-

aktionale Daten spielen eine wichtige Rolle in traditioneller Business Intelligence (BI) [LC16]. Werden diese im Data Lake verwaltet, müssen folglich weiterhin die traditionellen Analysen, wie z. B. Reporting und Online Analytical Processing (OLAP) [CCS12], auf diesen ermöglicht werden, wie sie auch in einem Data Warehouse möglich sind. Semi-strukturierte und unstrukturierte Daten, die mit Transaktionen assoziiert werden, sind ebenfalls transaktionale Daten, z. B. ein PDF-Dokument mit einer Rechnung oder ein Foto bei einer Produktreklamation. Unabhängig von ihrer Struktur müssen sie verwaltet und mit anderen Daten integriert werden, um ganzheitliche Analysen zu ermöglichen. Sie werden auch mit Stammdaten verknüpft, z. B. sind Kunden- und Produktdaten zentrale Bestandteile einer Reklamation. Transaktionale Daten werden von Geschäftsprozessen in großen Mengen generiert und sind darum voluminös.

- *Internet of Things (IoT)-Daten* sind all die Daten, die im Zusammenhang mit dem IoT erfasst werden, beispielsweise Sensorwerte oder Daten aus Fertigungsanlagen. Typischerweise werden sie als Datenstrom erfasst. Im Gegensatz zu transaktionalen Daten haben sie keinen direkten Bezug zu Geschäftstransaktionen, -prozessen oder -objekten. Solch ein Bezug muss erst über Integrationsschritte hergestellt werden, beispielsweise indem Sensorwerte über die Sensor-ID der Maschine zugeordnet werden, in der sie erfasst wurden. IoT-Daten sind meist sehr kurzlebig, da sie den aktuellen Kontext beschreiben, der sich sofort nach der Erfassung wieder ändern kann. Sie können sowohl semi-strukturiert als auch unstrukturiert sein. Semi-strukturierte IoT-Daten sind Sensorwerte, die als Schlüssel-Wert-Paare repräsentiert werden, etwa als JavaScript Object Notation (JSON)-Objekt. Unstrukturierte IoT-Daten dagegen sind, unter anderem, Bilder oder Videos, die durch das IoT erfasst wurden. IoT-Daten werden in periodischen Zeitintervallen erfasst, was zu großen Mengen heterogener Daten führt. In den meisten Fällen ist die Qualität dieser Daten unbekannt. So können Bilder verschwommen sein oder Sensorwerte können beeinflusst werden. IoT-Daten werden typischerweise mit Stammdaten oder transaktionalen Daten assoziiert; beispielsweise werden Sensormesswerte einem bestimmten Produkt zugeordnet. Darum

müssen, unabhängig von ihrer Struktur, all diese Daten verknüpft werden. Zusätzlich muss die Historie von IoT-Daten festgehalten werden, um z. B. den Verlauf von Sensordaten analysieren zu können.

Diese Aufzählung erhebt allerdings keinen Anspruch auf Vollständigkeit. Je nach Domäne und Anwendungsfall können weitere Datenkategorien von hoher Wichtigkeit sein. Allerdings reichen diese drei Kategorien aus, um daraus die Managementherausforderungen an den unternehmensweiten Data Lake abzuleiten.

3.2.2 Analysen und Nutzergruppen

In unterschiedlichen Geschäftsbereichen über das gesamte Unternehmen verteilt wurden mehrere Datenanalyseprojekte ins Leben gerufen, um den potentiellen Wert der genannten Daten auszuschöpfen (vgl. [TSRC15; SKW17]). Die Projektvielfalt ist groß, da die Projekte unterschiedliche Bereiche abdecken, wie beispielsweise *Beschwerdemanagement* und *Produktlebenszyklusmanagement*. Im Projekt zu Beschwerdemanagement werden beispielsweise Daten zu Kundenbestellungen aus ERP- und CRM-Systemen mit Bildern und Videos verbunden, die Kunden ihren Beschwerden beigefügt haben. Hierdurch können Produktreklamationen einfacher kategorisiert und Fehler identifiziert werden. Im Projekt zu Produktlebenszyklusmanagement werden kontinuierlich die Felddaten verkaufter Produkte beim Kunden gesammelt und mit Simulationsdaten aus dem Produktentwicklungsprozess abgeglichen. So kann das Verhalten des Produkts im Feld besser untersucht und die Produktentwicklung in zukünftigen Iterationen entsprechend angepasst werden. Die im Data Lake ausgeführten Analysen reichen dabei von beschreibend (descriptive) über diagnostisch (diagnostic) und vorhersagend (predictive) bis hin zu präskriptiv (prescriptive) [Hag16].

Auch die an den Analysen beteiligten Nutzergruppen sind stark heterogen. Nutzer*innen aus unterschiedlichen Domänen und mit verschiedenem Erfahrungsstand zu Datenaufbereitung und -Analyse greifen auf die Daten des Data Lake zu. Beispiele für solche Nutzergruppen sind: *Data Scientists* mit hoher Kenntnis zu Datenanalyse aber geringem Domänenwissen, *Domänenexpert*innen* mit guter Kenntnis zu ihren Daten und der Domäne aber eher

geringer Datenanalyseerfahrung und *Geschäftsnutzer*innen* mit hohem Domänenwissen aber wenig Erfahrung in der Datenanalyse. Die Analysen, die diese verschiedenen Nutzer*innen ausführen, reichen von OLAP und Reporting (Geschäftsnutzer*innen) über visuelle Analysen (Domänenexpert*innen) bis hin zu fortgeschrittenen Analysen (Data Scientists).

3.3 Zusammenfassung

Insgesamt ergibt sich in den vorangegangenen Abschnitten ein sehr heterogenes und dadurch herausforderndes Anwendungsszenario für den Data Lake. Es müssen nicht nur unterschiedlichste Daten verwaltet, sondern auch eine Vielzahl grundsätzlich verschiedener Analysen und Nutzer*innen unterstützt werden, etwa Data Scientists, Domänenexpert*innen, oder Geschäftsnutzer*innen. Das folgende Kapitel untersucht, welche konkreten Anforderungen sich aus diesem Anwendungsszenario ergeben und inwiefern diese durch existierende Konzepte zu Data Lakes abgedeckt werden.

Kapitel 4

Praktische Anforderungen und Forschungslücken

Hergeleitet aus der Literatur (siehe Abschnitt 2.1) sowie der praktischen Betrachtung realer Fallbeispiele (siehe Kapitel 3) ergibt sich eine Reihe an Anforderungen, die ein Data Lake in der Praxis erfüllen muss (Forschungsbeitrag B1.2). Mithilfe dieser Anforderungen können die Forschungslücken in der existierenden Data-Lake-Literatur identifiziert werden (Forschungsbeitrag B1.3). Dieses Kapitel stellt zunächst in Abschnitt 4.1 die praktischen Anforderungen dar, die an einen unternehmensweiten Data Lake gestellt werden, bevor in Abschnitt 4.2 die verbleibenden Forschungslücken detailliert werden. Abschließend fasst Abschnitt 4.3 die Ergebnisse des Kapitels zusammen.

Dieses Kapitel ist eine abgeänderte Version einer vorausgegangenen Veröffentlichung der Autorin [GGH+20a].

4.1 Praktische Anforderungen

Basierend auf den Erkenntnissen aus sowohl der Literatur als auch den diskutierten Fallbeispielen lassen sich vier zentrale Anforderungen (A) an einen unternehmensweiten Data Lake ableiten (siehe Forschungsbeitrag B1.2):

A1 *Verwaltung und Integration vielfältiger Daten.* Alle drei in Abschnitt 3.2.1 beschriebenen Datenarten sollen in einem unternehmensweiten Data Lake verwaltet werden. Dies führt dazu, dass Daten verschiedenster Struktur (strukturiert, semi-strukturiert, unstrukturiert) in einem Data Lake zusammenkommen. Hierbei gilt es, die Daten je nach Zusammenhang zu integrieren, unabhängig vom Grad ihrer Strukturiertheit. Im Falle einer Produktreklamation beispielsweise können unstrukturierte Fotos und Videos der fehlerhaften Funktionsweise oder der Schäden analysiert werden, um den auftretenden Fehler zu klassifizieren, z. B. einen Materialfehler. Durch die Verknüpfung mit den strukturierten Produktdaten kann das fehlerhafte Material identifiziert werden. Die so gewonnenen Erkenntnisse können dazu genutzt werden, ähnlichen Fehlern vorzubeugen. Es gilt allerdings auch, weitere Eigenschaften der Daten bei der Verwaltung zu berücksichtigen, wie beispielsweise deren Volumen.

A2 *Unterstützung verschiedener analytischer Anwendungsfälle.* Die analytischen Anwendungsfälle, die mit den Daten im Data Lake umgesetzt werden, sind so vielfältig wie die Daten selbst. Wie bereits in Abschnitt 3.2.1 und Abschnitt 3.2.2 beschrieben, werden einige der Daten für Reporting und Online Analytical Processing (OLAP) verwendet, z. B. für die Berechnung wichtiger Key-Performance-Indikatoren (KPIs), wie Verkaufszahlen. Hinzu kommen aber auch fortgeschrittene Analysen, insbesondere maschinelles Lernen, mithilfe derer neue Erkenntnisse über Prozesse, Kunden und andere Aspekte gewonnen werden können. Auf den Daten eines Data Lake werden somit verschiedene Arten von Anwendungsfällen ausgeführt [Rus17].

A3 *Unterstützung verschiedener Verarbeitungsvarianten.* In einem Data Lake müssen große Mengen an Daten effizient verarbeitet werden. Eine Stapelverarbeitung kann diese Effizienz bieten. Die Einbindung von Internet of Things (IoT)-Daten erfordert zudem die Unterstützung der Datenstromverarbeitung im Data Lake. Ein unternehmensweiter Data Lake muss all diese Varianten der Verarbeitung unterstützen können.

A4 *Governance für verschiedenartige Daten.* Unter dem Begriff Data-Lake-Governance werden Strategien und Regeln zusammengefasst, die bei-

spielsweise zu Datensicherheit, Datenqualität oder auch zum Einhalten von Rechtsgrundlagen im Data Lake beitragen [CJL+15]. Wie in Abschnitt 3.2.1 dargestellt, benötigen manche der im Data Lake verwalteten Daten, insbesondere Stammdaten, eine strenge Governance, um sie vor unerlaubter Manipulation zu schützen und eine hohe Datenqualität zu gewährleisten. Für andere, weniger kritische Daten dagegen stellt eine solch umfangreiche Governance einen unverhältnismäßigen Mehraufwand dar, der flexible und explorative Analysen hemmen oder gar verhindern kann. Demnach ist es notwendig, die verschiedenen Datenarten mit den entsprechenden Richtlinien zu verwalten.

Diese Anforderungen muss ein Data Lake erfüllen, um den Bedarfen aus der Industriepraxis gerecht zu werden.

4.2 Forschungslücken

Beim Abgleich der in Abschnitt 4.1 hergeleiteten praktischen Anforderungen an Data Lakes mit existierenden Arbeiten aus dem Data-Lake-Kontext zeigt sich, dass Letztere zur Abdeckung der Anforderungen nicht ausreichend sind. So ergeben sich verschiedene Forschungslücken, die es zu adressieren gilt (siehe Forschungsbeitrag B1.3). Dieser Abschnitt diskutiert diese in mehr Detail. Die Forschungslücken sind auch in Abbildung 4.1 dargestellt.

F1 *Unklare Datenmodellierungsmethoden.* Zwar existieren verschiedene Datenmodellierungsmethoden sowohl aus dem Data-Lake-Kontext als auch aus anderen Bereichen, allerdings gibt es kaum Evaluationen oder Bewertungen dieser Ansätze. Es bleibt unklar, welche Datenmodellierungsmethoden für Data Lakes geeignet sind. Besonders in Datenmodellierungsmethoden aus anderen Bereichen fehlt auch die Möglichkeit, heterogene Daten zu integrieren (siehe A1). Entsprechend müssen diese erweitert werden, um in Data Lakes anwendbar zu sein.

F2 *Fehlende Data-Lake-Organisation.* Bezüglich der geeigneten Organisation und des Managements von Daten im Data Lake stellt die Heterogenität der verfügbaren Ansätze eine Herausforderung dar (siehe Abschnitt 2.1). Hier gibt es weder ein allgemein akzeptiertes Konzept, noch werden

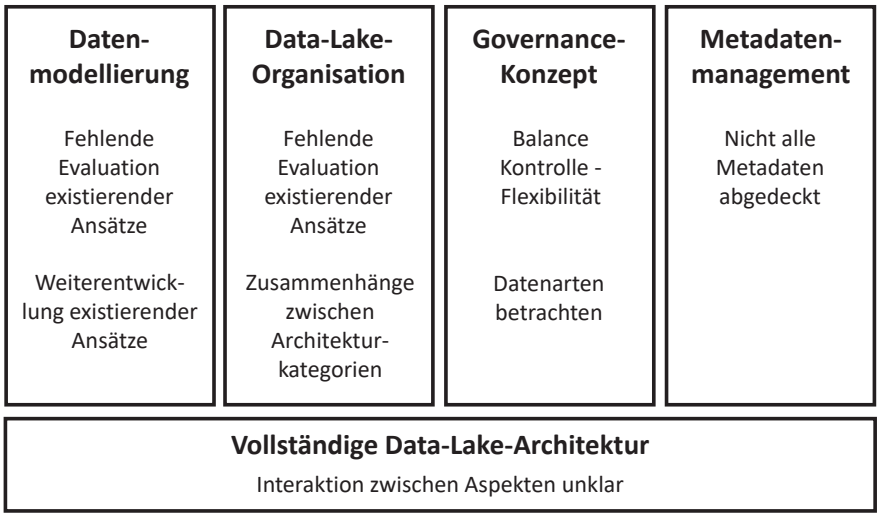


Abbildung 4.1: Übersicht über die identifizierten Forschungslücken [GGH+20a]

vorgeschlagene Architekturen evaluiert. Es bleibt somit unklar wie verschiedenste Analysen für unterschiedliche Nutzergruppen unterstützt werden können (siehe A2). Zudem fehlen oft Details zur Herleitung oder Umsetzung der Konzepte, wodurch sie schwer zu realisieren sind. Hier ist es notwendig, die existierenden Arbeiten vergleichend zu evaluieren und einheitliche Architekturkonzepte zu entwickeln, die auch die Zusammenhänge zwischen konzeptioneller Architektur und physischer Umsetzung betrachten.

F3 *Unvollständiges Governance-Konzept.* Für Data Lakes ist ein Governance-Konzept notwendig, das die benötigte hohe Flexibilität in den Verarbeitungsvarianten (siehe A3) unterstützen kann. So muss es z. B. möglich sein, Daten für explorative Analysen zuzugreifen und zu verwenden, ohne dass strenge Regularien dies behindern. Zudem muss eine Balance zwischen Kontrolle und Flexibilität erreicht werden. Insbesondere müssen dabei die verschiedenen Datenarten und deren Anforderungen an Data-Lake-Governance, beispielsweise unterschiedliche Anforderungen an Datenqualität, berücksichtigt werden (siehe A4). Zur Adressierung dieser Forschungslücke ist es notwendig, ein ganzheitliches Konzept zur Governance in Data Lakes zu entwickeln, welches ein entsprechendes organisatorisches Regelwerk für die Rollen und Prozesse im Data Lake umfasst.

F4 *Unvollständiges Metadatenmanagementkonzept.* Weder kommerzielle Software noch offene Ansätze aus der Forschung zu Metadatenmanagement decken alle im Data Lake benötigten Metadaten ab. So können insbesondere verschiedene Analysen und Governance nicht optimal unterstützt werden (siehe A2 und A4). Es ist daher notwendig, existierende Ansätze anzupassen und zu erweitern, um umfassendes Metadatenmanagement zu ermöglichen.

F5 *Fehlende vollständige Data-Lake-Architektur.* Zusätzlich zu den Problematiken, die sich hinsichtlich der betrachteten Data-Lake-Aspekte ergeben, fehlt es an einer allgemeinen und vollständigen Architektur zur Umsetzung eines Data Lake. Solch eine Architektur befasst sich nicht nur mit einzelnen Data-Lake-Aspekten, sondern betrachtet zudem die Interaktionen zwischen den verschiedenen Teilen des Data Lake. Beispielsweise werden Zusammenhänge zwischen der Datenmodellierung und der Data-Lake-Organisation berücksichtigt. So entsteht eine zusammenhängende, systematische und übergreifende Data-Lake-Architektur, die eine Anleitung für eine praktische Umsetzung liefert. Dies ist notwendig für die Erfüllung aller vier beschriebenen Anforderungen.

Diese Forschungslücken gilt es zu schließen, um das volle Potential von Data Lakes in der Praxis ausnutzen zu können.

4.3 Zusammenfassung

In diesem Kapitel wurden die praktischen Anforderungen an einen Data Lake aus der Literatur sowie dem in Abschnitt 3.2 vorgestellten Fallbeispiel identifiziert. So ergaben sich vier Anforderungen, namentlich A1: Verwaltung und Integration vielfältiger Daten, A2: Unterstützung verschiedener analytischer Anwendungsfälle, A3: Unterstützung verschiedener Verarbeitungsvarianten und A4: Governance für verschiedenartige Daten. Beim Abgleich dieser Anforderungen mit existierenden Arbeiten aus dem Data-Lake-Kontext zeigte sich, dass es fünf Forschungslücken zu schließen gilt, um diese Anforderungen zu erfüllen: F1: Unklare Datenmodellierungsmethoden, F2: Fehlende Data-Lake-Organisation, F3: Unvollständiges Governance-Konzept, F4: Unvollstän-

diges Metadatenmanagementkonzept, F5: Fehlende vollständige Data-Lake-Architektur. Da die Betrachtung all dieser Forschungslücken den Rahmen dieser Arbeit sprengt, soll im Folgenden ein Fokus auf F2 sowie F5 gelegt werden. Diese Forschungslücken spiegeln sich auch in den definierten Forschungszielen Z2 und Z3 in Abschnitt 1.2 wieder.

Kapitel 5

Das Data Lake Architecture Framework

Trotz der Vielzahl der verfügbaren Konzepte für Data Lakes, wie in Abschnitt 2.1 aufgeführt, gibt es noch einige Forschungslücken, die es zu schließen gilt, um Data Lakes effizient und vorteilhaft einzusetzen (siehe Abschnitt 4.2). Eine dieser Forschungslücken betrifft das Fehlen einer vollständigen Data-Lake-Architektur (siehe F5), die die Konzepte spezifiziert, die im Data Lake Anwendung finden sollen. Um diese Lücke und damit Forschungsziel 2 zu adressieren, stellt dieses Kapitel das Data Lake Architecture Framework (DLAF) vor, welches die Basis für die Definition einer vollständigen Data-Lake-Architektur bildet. Dazu bietet das DLAF eine Entscheidungsunterstützung für Entwickler bei der Wahl geeigneter Konzepte für den Data Lake. Zudem kann das DLAF verwendet werden, um bereits bestehende Data-Lake-Architekturen hinsichtlich ihrer Vollständigkeit zu evaluieren.

Bei diesem Kapitel handelt es sich um eine überarbeitete Version einer vorausgegangenen Veröffentlichung der Autorin [GGH+21].

Diese Arbeit definiert eine *vollständige Data-Lake-Architektur* wie folgt:

> Eine Data-Lake-Architektur ist dann vollständig, wenn alle für die Funktion des Data Lake notwendigen Aspekte und deren Abhängigkeiten durch die Architektur abgedeckt werden.

Ein *Aspekt* ist eine Perspektive auf den Data Lake, wie beispielsweise die verwendete Datenmodellierung oder das genutzte Metadatenmanagement. Eine Data-Lake-Architektur besteht aus dem Zusammenschluss mehrerer Konzepte für die einzelnen Aspekte. Dabei gilt es, diese Konzepte so zu wählen, dass eine effiziente Architektur entsteht und das gewünschte Anwendungsszenario möglichst optimal unterstützt werden kann. Beispielsweise sollten der Datenspeicher so gewählt werden, dass die gewählte Datenmodellierung unterstützt wird.

In der Literatur existieren einige Konzepte, die von ihren Autoren als „Data-Lake-(Referenz-)Architektur“ bezeichnet werden, etwa in [Sha18]. Allerdings fokussieren sich diese lediglich auf eine Untermenge der benötigten Aspekte, wie Datenorganisation [Sha18]. Zudem sind die möglichen Anwendungsgebiete für Data Lakes hochgradig divers, wie in Kapitel 3 dargestellt. Beispielsweise kann ein Data Lake lediglich Stapelverarbeitung [MBG+17] oder sowohl Stapel- als auch Datenstromverarbeitung umfassen [MM18]. Abhängig von der Art der Daten und des Szenarios, für die der Data Lake verwendet wird, ergeben sich unterschiedliche Anforderungen an seine Architektur. Daher erweist sich die Erstellung einer allgemeingültigen und universell anwendbaren Data-Lake-Architektur als schwierig. Stattdessen wird an dieser Stelle das DLAF vorgeschlagen, welches ein Architekturrahmenwerk für die Entwicklung einer vollständigen Data-Lake-Architektur stellt. Eine Definition des Begriffes „Architekturrahmenwerk“ findet sich in Abschnitt 2.3. Zwar gibt es bereits Architekturrahmenwerke in verschiedenen Domänen, wie beispielsweise das Zachman-Framework [Zac87] für die Architektur von Informationssystemen. Allerdings existieren im Kontext von Data Lakes keine solchen Ansätze. Der Zusammenhang zwischen dem DLAF und einer daraus abgeleiteten Architektur ist in Abbildung 5.1 dargestellt. Dabei handelt es sich bei der Architektur um eine Instanziierung des Architekturrahmenwerks. Die Anleitung, die das DLAF bereitstellt, gliedert sich in drei Bereiche: 1) Das DLAF definiert eine Menge von Aspekten, die für einen Data Lake notwendig sind, z. B. Datenmodellierung (siehe Forschungsbeitrag B2.1). 2) Es assoziiert jeden dieser Aspekte mit

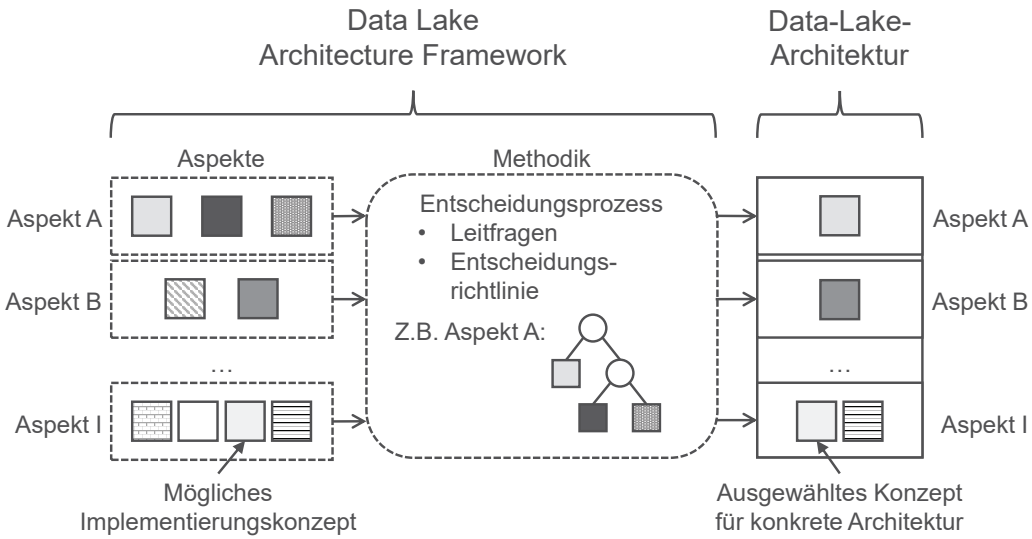


Abbildung 5.1: Das DLAF enthält mögliche Implementierungskonzepte. Um eine Data-Lake-Architektur daraus zu konfigurieren, werden konkrete Konzepte aus dem Rahmenwerk anhand der enthaltenen Methodik ausgewählt [GGH+21].

einer Menge möglicher Implementierungskonzepte, z. B. Data Vault [Lin12]. 3) Es stellt eine Methodik zur Verfügung, welche bei der Auswahl geeigneter Konzepte unterstützt unter Berücksichtigung von Abhängigkeiten zwischen den Aspekten, z. B. zwischen Datenmodellierung und Datenspeicher (siehe Forschungsbeitrag B2.2).

Der Rest dieses Kapitels ist wie folgt unterteilt: Abschnitt 5.1 stellt die verwandten Arbeiten zum DLAF vor. In Abschnitt 5.2 werden die Aspekte beschrieben, welche im DLAF enthalten sind. Auch werden hier die assoziierten Konzepte kurz diskutiert. Abschnitt 5.3 behandelt die Abhängigkeiten zwischen diesen Aspekten. Abschnitt 5.4 stellt die Methodik vor, die ebenfalls Teil des DLAF ist und die Definition einer vollständigen Data-Lake-Architektur ermöglicht. Abschließend fasst Abschnitt 5.5 das Kapitel zusammen. Die entwickelten Konzepte werden in Kapitel 9 evaluiert.

5.1 Verwandte Arbeiten

In der Literatur findet sich eine Vielzahl so genannter „Data-Lake-Architekturen“ (z. B. in [Inm16; JQ17; RZ19; Sha18]). Das Ziel dieser Architek-

turen ist es, möglichst generisch zu sein. Sawadogo und Darmont differenzieren drei Arten von Data-Lake-Architekturen voneinander [SD21]:

1. *Funktionale Architekturen (functional architectures)*, welche sich mit Datenaufnahme und -speicherung befassen, z. B. [JQ17]
2. *Datenreifebasierte Architekturen (data maturity-based architectures)*, in denen Daten anhand ihres Verarbeitungsgrades organisiert werden, z. B. [Sha18]
3. *Hybride Architekturen (hybrid architectures)*, welche beide Sichtweisen kombinieren

Da sich sowohl funktionale als auch datenreifebasierte Architekturen nur auf einzelne Aspekte des Data Lake fokussieren [SD21], qualifizieren sich diese nicht als vollständige Data-Lake-Architekturen. Auch hybride Architekturen bieten nicht den notwendigen Umfang, da sie sich lediglich auf Datenaufnahme, -speicherung und -organisation fokussieren. Weitere wichtige Aspekte sind Datenmodellierung und Metadatenmanagement (siehe Abbildung 4.1). In ihrer Arbeit nennen Sawadogo und Darmont zwei hybride Architekturen: Inmons Data-Pond-Architektur [Inm16] und Ravats funktionale Data-Lake-Architektur [RZ19]. Keine dieser beiden Architekturen umfasst die genannten zusätzlichen Aspekte (Datenmodellierung, Metadatenmanagement). Nach unserem besten Wissen sind keine weiteren allgemeingültigen hybriden Architekturen verfügbar. Zwar gibt es verschiedenste Arbeiten zu sowohl Datenmodellierung als auch Metadatenmanagement in Data Lakes (z. B. [EGG+20; HGQ16; Hou17; NRD18], allerdings fokussieren sich diese nicht auf eine Gesamtarchitektur des Data Lake. Insgesamt zeigt sich, dass keine dieser Architekturen aus der Literatur vollständig ist.

Zusätzlich zu den allgemeingültigen Architekturen gibt es auch Data-Lake-Architekturen in spezifischen Implementierungen, z. B. in [MBG+17; MM18]. Diese Architekturen sind jedoch auf spezifische Anwendungsfälle und -szenarien zugeschnitten und darum nicht als allgemeingültige Data-Lake-Architekturen anwendbar. Daher gibt es unseres Wissens nach keine Leitlinien um eine vollständige Data-Lake-Architektur zu erstellen.

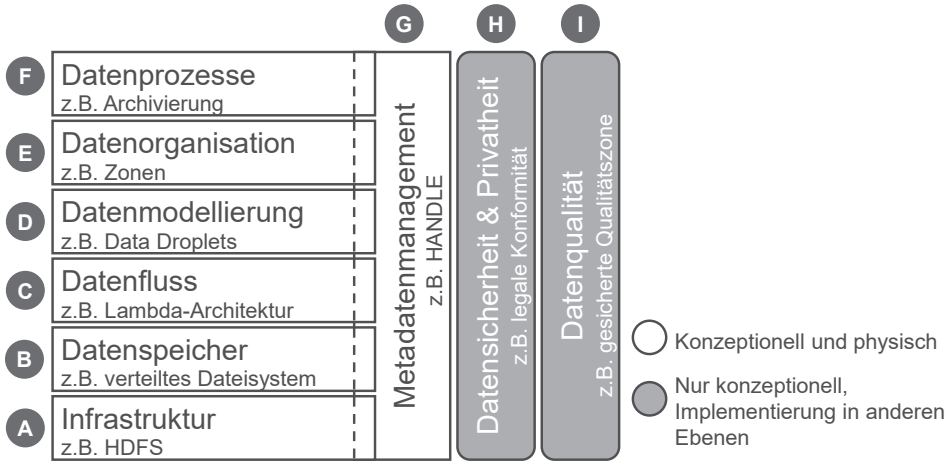


Abbildung 5.2: Das DLAF besteht aus neun Aspekten, die es bei der Definition einer vollständigen Data-Lake-Architektur zu Berücksichtigen gilt [GGH+21].

5.2 Aspekte des DLAF

Um die fehlende Unterstützung bei der Konfiguration einer vollständigen Data-Lake-Architektur zu adressieren, präsentiert dieses Kapitel das DLAF. Dieses Framework besteht aus zwei Teilen: I) Es beschreibt notwendige Aspekte eines Data Lake. Dadurch definiert es den Rahmen für eine vollständige Data-Lake-Architektur. In diesem Abschnitt werden die Aspekte beschrieben, die Bestandteil des DLAF sind. Abschnitt 5.3 detailliert anschließend die Abhängigkeiten zwischen diesen Aspekten. II) Zudem enthält das DLAF eine Methodik, um eine vollständige Data-Lake-Architektur zu definieren. Diese Methodik leitet die Auswahl geeigneter Konzepte für jeden Aspekt. Sie ist in Abschnitt 5.4 vorgestellt.

Abbildung 5.2 zeigt das DLAF. Jeder Teil des Framework repräsentiert einen Aspekt von Data-Lakes, der mit einer Menge von Konzepten für seine Implementierung assoziiert wird (vgl. Abbildung 5.1). Die Aspekte, die im DLAF enthalten sind, wurden im Rahmen einer sorgfältigen Literaturrecherche zu Data-Lake-Konzepten identifiziert (siehe Abschnitt 2.1). Die Ergebnisse dieser Literaturrecherche wurden nach ihren Eigenschaften, ihrem Fokus und ihrem

Abstraktionslevel geclustert. Aus den neun entstandenen Clustern wurden die disjunkten Aspekte formuliert, die in Abbildung 5.2 dargestellt sind:

A) Infrastruktur

B) Datenspeicher

C) Datenfluss

D) Datenmodellierung

E) Datenorganisation

F) Datenprozesse

G) Metadatenmanagement

H) Datensicherheit & Privatheit

I) Datenqualität

Diese Aspekte können weder weiter zusammengefasst werden, da sie unterschiedliche Perspektiven auf den Data Lake beschreiben, noch konnten weitere zu betrachtende Aspekte gefunden werden. Die Aspekte A-F sind nach ihrem Abstraktionsgrad sortiert, mit Infrastruktur als der am wenigsten abstrakte Aspekt unten und Datenprozesse als der abstrakteste Aspekt oben. Die Aspekte G-I übergreifen alle diese anderen Aspekte. Es wird zwischen Aspekten, die aus einem Konzept und dessen physischer Implementierung bestehen (weiß), und Aspekten, die nur eine konzeptionelle Sicht umfassen, aber durch andere Aspekte implementiert werden (grau) unterschieden. Falls beispielsweise der Datensicherheit & Privatheit-Aspekt Datenverschlüsselung verlangt, muss die Infrastruktur diese Funktionalität bieten. Die folgenden Abschnitte beschreiben die Aspekte in mehr Detail.

A) Infrastruktur. Der Infrastrukturaspekt enthält Konzepte für die Implementierung des Data Lake. Der Fokus liegt auf konkreten Speichersystemen und Werkzeugen, z. B. Hadoop Distributed File System (HDFS) als verteiltes Dateisystem oder MySQL[1] als relationale Datenbank. Auch zur Infrastruktur gehört der Einsatz von Cloud-Technologien, also ob Speichersysteme und Werkzeuge On-Premise oder in der Cloud verwendet werden. Ein Beispielkonzept ist durch

[1] www.mysql.com/de/

Zikopoulos et al. gegeben, welche Hadoop[2] und DB2[3] für ihren Data Lake verwenden [ZDB+15]. Als ein Beispielkonzept für die Nutzung von Cloud-Technologien existieren hybride Data Lakes, welche sowohl On-Premise als auch in der Cloud realisiert sind [Loc16].

B) Datenspeicher. Der Aspekt des Datenspeichers fokussiert sich auf die Arten von Systemen und Werkzeugen, die für die Speicherung und Verarbeitung von Daten verwendet werden (z. B. Dateisysteme und NoSQL Datenbanken, oder Werkzeuge für die Stapel- und Datenstromverarbeitung). Im Gegensatz zum Infrastrukturaspekt werden hier keine spezifischen Systeme oder Werkzeuge ausgewählt, sondern lediglich eine Gruppe solcher. Beispiele sind Data Lakes, die auf einem einzelnen verteilten Dateisystem (z. B. [MBG+17]), oder auf mehreren Speichersystemen aufgebaut sind (z. B. [ZDB+15]).

C) Datenfluss. Der Datenflussaspekt befasst sich mit der Architektur und Interaktion für die zwei Arten von Datenbewegung im Data Lake: ruhende Daten und Datenströme. Ruhende Daten (engl. Batch Data) sind in einem Speichersystem persistent gespeichert. Sie werden typischerweise alle auf einmal in großen Mengen verarbeitet [CY15]. Datenströme dagegen, auch Echtzeitdaten genannt, werden kontinuierlich in den Data Lake eingeliefert und müssen meist sofort verarbeitet werden [CY15]. Datenströme können auch in ruhende Daten umgewandelt werden, wenn sie für die spätere Nutzung persistiert werden. Beispiele für Datenflusskonzepte sind hybride Verarbeitungsarchitekturen, wie beispielsweise die Lambda-Architektur [MW15] oder (Hybrid Processing Architecture for Big Data (BRAID) [GSSM18]. In beiden Architekturen werden sowohl ruhende Daten als auch Datenströme parallel verarbeitet.

D) Datenmodellierung. Der Datenmodellierungsaspekt beschreibt ob und wie Daten im Data Lake modelliert werden. Typischerweise unterscheidet sich die genutzte Modellierungstechnik je nach den Eigenschaften und der geplanten Verwendung der Daten. Beispiele für Datenmodellierungstechniken, die in Data Lakes Anwendung finden, sind Data Droplets, wo Daten als Graph gespeichert werden [Hou17], oder Data Vault, wo eine dreiteilige Tabellenstruktur verwendet wird [Lin12].

[2] hadoop.apache.org/
[3] www.ibm.com/analytics/db2

E) Datenorganisation. Der Aspekt der Datenorganisation definiert den konzeptionellen Aufbau innerhalb des Data Lake. Dazu beschreiben assoziierte Konzepte, welche Daten wo im Data Lake abgelegt sind und in welchem Zustand sie sich befinden (z. B. roh oder vorverarbeitet). Beispiele sind die Data-Pond-Architektur [Inm16], wo Daten aufgrund ihrer Struktur und Herkunft organisiert werden, die Zonenmodelle (z. B. [GGH+20b; Sha18]), die Daten nach Verarbeitungsgrad separieren, und Data Meshes für eine semantische Datenorganisation [Deh19].

F) Datenprozesse. Datenprozesse können unterteilt werden in *Prozesse für Datenlebenszyklusmanagement* und *Prozesse zur Datenpipeline*. Datenlebenszyklusmanagementprozesse verwalten Daten von ihrer Erstellung und Erfassung bis zur Löschung [DAM17]. Diese Prozesse müssen innerhalb eines Data Lake detailliert definiert und standardisiert werden, um Self-Service zu ermöglichen, Datennutzbarkeit sicherzustellen und legale Konformität zu gewährleisten. So muss etwa definiert werden, wie Nutzer*innen auf Daten und deren Metadaten zugreifen können oder wie Zugriffsbeschränkungen umgesetzt werden. Im Gegensatz dazu befassen sich Datenpipelineprozesse mit der technischen Aufnahme, Bewegung und Verarbeitung von Daten. Hierzu zählen etwa Extract-Transform-Load (ETL)-Prozesse. Datenpipelineprozesse werden verwendet, um beispielsweise zu beschreiben, wie sich Daten zwischen Zonen in einem Zonenmodell bewegen. Im Gegensatz zum Datenflussaspekt befasst sich der Datenprozessaspekt damit, was mit Daten getan wird, anstatt sich auf die Eigenschaften der Daten selbst zu fokussieren. Ein beispielhafter Datenlebenszyklusmanagementprozess ist der Archivierungsprozess gegeben in [CJL+15]. Beispiele für Datenpipelineprozesse können in den meisten Zonenmodelle gefunden werden (z. B. in [GGH+20b]) oder in Jarkes und Quix' konzeptioneller Data-Lake-Architektur [JQ17].

G) Metadatenmanagement. Der Metadatenmanagementaspekt umfasst zwei Unteraspekte. Der erste, *Metadaten als Unterstützer*, überlappt die horizontalen Aspekte in Abbildung 5.2. In diesem Unteraspekt werden Metadaten dazu verwendet, andere Aspekte zu unterstützen. So beschreiben Metadaten etwa, zu welcher Zone ein Datum in einem Zonenmodell gehört oder wann Daten erstellt wurden für das Lebenszyklusmanagement. Der andere Unteraspekt ist *Metadaten als Funktion*, dargestellt als die rechte Seite des Metadatenmanagement in

Abbildung 5.2. Dieser Unteraspekt enthält die Funktionalitäten, die Metadaten über ihre Unterstützungsfunktion hinaus bieten können, wie beispielsweise Business Glossaries [BCJ+14] oder semantische Beschreibungen. Für alle Metadaten müssen auch die Aspekte A-F definiert werden, da Metadaten ebenfalls gespeichert, modelliert oder organisiert werden müssen. Beispielkonzepte sind Metadatenmanagementsysteme, wie Constance [HGQ16] oder Metadatenmodelle wie HANDLE [EGG+20].

H) Datensicherheit & Privatheit. Dieser Aspekt ist ein rein konzeptioneller Aspekt. Er ist von großer Wichtigkeit in einem Data Lake, da er legale Konformität, Ausrichtung mit Geschäftszielen und mehr sicherstellt [CJL+15]. Beispielkonzepte für diesen Aspekt sind das Prüfen von legaler Konformität im Data-Wrangling-Prozess [TSRC15] oder AMNESIA für DSGVO-konformes Machine Learning [SGW+20], wo Datensicherheit & Privatheit durch Zonen implementiert werden.

I) Datenqualität. Der Datenqualitätaspekt ist ebenfalls rein konzeptionell. Die Datenqualität aufrecht zu erhalten ist wichtig um die Datennutzbarkeit sicherzustellen und die Wandlung des Data Lake in einen Data Swamp zu verhindern [CSN+14]. Während es Werkzeuge für Datenqualitätsmanagement gibt, z. B. Informatica Data Quality[4], verlassen sich diese dennoch auf Implementierung durch andere Aspekte, wie beispielsweise Metadatenmanagement. Datenqualität kann auch in Konzepten zu z. B. Datenorganisation gefunden werden, wo manche Zonen vertrauenswürdige und qualitätsgesicherte Daten beinhalten (z. B. in [GGH+20b]).

5.3 Abhängigkeiten innerhalb des DLAF

Die Aspekte des DLAF sind nicht unabhängig voneinander, da Entscheidungen für einen Aspekt andere Aspekte beeinflussen. Zum Beispiel beeinflusst der Aspekt der Datenspeicherung die Datenmodellierung, da verschiedene Arten von Speichersystemen verschiedene Arten der Modellierung unterstützen (z. B. relationale Modellierung für relationale Datenbanken, Graphen für Graphdatenbanken). Dieser Abschnitt erforscht die Abhängigkeiten zwischen den Aspekten und deren Auswirkungen.

[4] https://www.informatica.com/de/products/data-quality.html

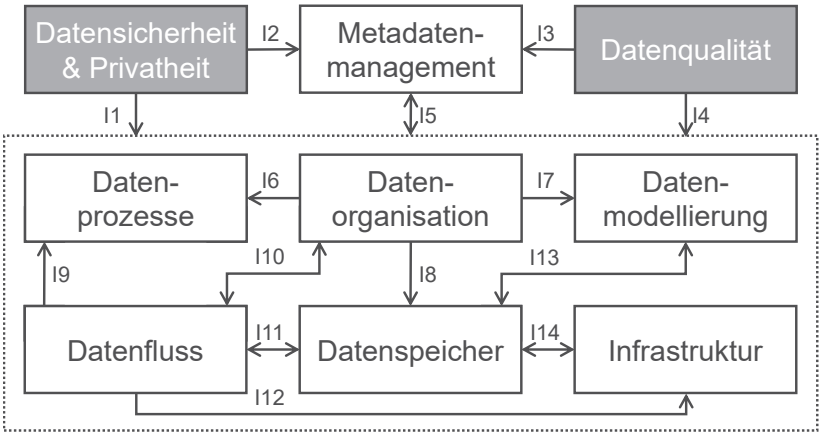


Abbildung 5.3: Der Abhängigkeitsgraph der Aspekte des DLAF. Gerichtete Pfeile zeigen vom beeinflussenden Aspekt zum beeinflussten Aspekt. [GGH+21].

Abbildung 5.3 stellt diese Abhängigkeiten als Graph dar. Sie stehen im Einklang mit den in der vorhandenen Literatur beschriebenen Abhängigkeiten. In diesem Graph sind sechs der insgesamt neun Aspekte zur einfacheren Visualisierung gruppiert. Die dargestellten Einflüsse zwischen den Aspekten bilden die Grundlage für die Methodik, die in Abschnitt 5.4 vorgestellt ist. Die Grafik zeigt, dass die Aspekte Datensicherheit & Privatheit und Datenqualität alle anderen Aspekte des DLAF beeinflussen (I1-I4). Dies liegt daran, dass diese beiden Aspekte nicht direkt, sondern durch andere Aspekte implementiert werden. Daher müssen Entscheidungen, die für Datensicherheit& Privatheit und Datenqualität getroffen wurden, bei der Auswahl von Konzepten für alle anderen Aspekte berücksichtigt werden. Für den Aspekt des Metadatenmanagements gilt, dass alle Aspekte Metadaten erzeugen und Metadaten nutzen, z. B. Metadaten, die beschreiben, wie Daten verarbeitet werden sollen (I5). Alle Aspekte beeinflussen also, welche Metadaten gesammelt werden und wie sie organisiert werden sollten. Zugleich sind Metadaten auch Daten und müssen daher bei der Definition aller anderen Aspekte berücksichtigt werden (I5). Der Aspekt der Datenorganisation beeinflusst die Datenprozesse (I6), da die Datenpipelineprozesse an die gewählte Datenorganisation, z. B. Zonen oder Data Ponds, angepasst werden müssen. Die Datenorganisation beeinflusst auch die Datenmodellierung (I7), da z. B. Zonenmodelle typischerweise eine standardisierte Datenmodellierung in mindestens einer Zone vorschreiben. Dies bedeutet, dass alle Daten in dieser Zone nach festgelegten Regeln model-

liert werden, z. B. nach den Regeln des Data Vault. Außerdem beeinflusst die Datenorganisation den Datenspeicher (I8), da z. B. die Data-Pond-Architektur, bei der die Daten nach ihrer Struktur unterteilt sind, ein anderes Speicherkonzept erfordert als z. B. Zonenmodelle. Der Aspekt des Datenflusses beeinflusst die Datenprozesse (I9), die Datenorganisation (I10), den Datenspeicher (I11) und die Infrastruktur (I12). Dies liegt daran, dass der Datenflussaspekt die verschiedenen Modi von Datenbewegung (ruhend und Strom) umfasst. Abhängig von den Entscheidungen, die für diesen Aspekt getroffen werden, müssen auch andere Aspekte die jeweiligen Modi unterstützen. Zum Beispiel, wenn das Konzept, das für den Datenflussaspekt gewählt wurde, die Verarbeitung von Datenströmen beinhaltet, muss das Konzept für die Infrastruktur Systeme zur Datenstromverarbeitung enthalten. Für die Datenorganisation und den Datenspeicher (I10, I11) kann der Einfluss auch umgekehrt sein, da z. B. die Verwendung einer reinen stapelbasierten Datenorganisation oder die Wahl von Stapel- und Datenstromverarbeitung im Datenspeicher einen bestimmten Datenfluss vorschreiben. Der Aspekt des Datenspeichers beeinflusst und wird beeinflusst sowohl von der Datenmodellierung (I13) als auch von der Infrastruktur (I14). Je nach Art der gewählten Speichersysteme können bestimmte Modellierungstechniken verwendet werden oder nicht. Wird beispielsweise eine relationale Speicherung gewählt, sollten die Datenmodelle auf relationale Schemata anwendbar sein. Gleichzeitig macht die Wahl eines bestimmten Datenmodells ein anderes Datenspeicherkonzept erforderlich. Datenspeicher und Infrastruktur sind eng miteinander verbunden, da das eine die Art der Systeme wählt und das andere die konkreten Systeme und Werkzeuge definiert.

5.4 Methodik zur Definition einer Data-Lake-Architektur mit dem DLAF

Um eine vollständige Data-Lake-Architektur aus dem Framework zu konfigurieren, müssen aus der Menge der Konzepte, die mit jedem Aspekt verbunden sind, bestimmte Konzepte ausgewählt werden (siehe Abbildung 5.1). Dieser Abschnitt bietet eine Methodik hierfür, die aus neun Schritten besteht. Mehrere dieser Schritte entsprechen direkt den Aspekten des DLAF (siehe Ab-

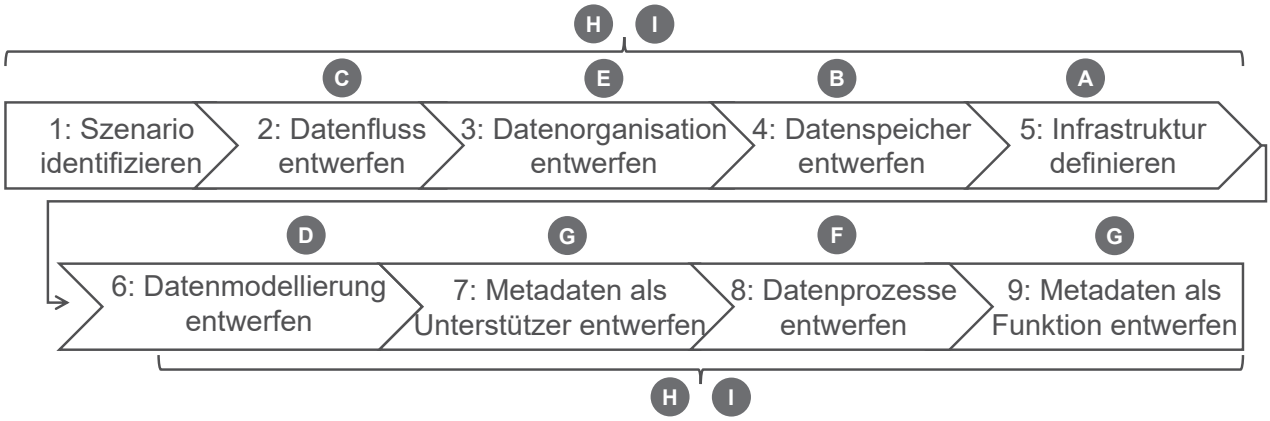


Abbildung 5.4: Um eine vollständige Data-Lake-Architektur mit dem DLAF zu konfigurieren, sind neun Schritte notwendig. Jeder Schritt ist mit verschiedenen DLAF-Aspekten verbunden, die in den Kreisen dargestellt sind. [GGH+21].

bildung 5.4), und die Reihenfolge der Schritte wurde aus den Abhängigkeiten der Aspekte untereinander bestimmt (siehe Abbildung 5.3).

Wie in Abbildung 5.4 dargestellt, sind die Aspekte Datensicherheit & Privatheit und Datenqualität den Schritten 1 bis 9 zugeordnet. Das bedeutet, dass bei allen Schritten diese beiden Aspekte berücksichtigt und entsprechend einbezogen werden müssen. Unsere Methodik bietet für jeden Schritt Leitfragen, die die Auswahl geeigneter Konzepte für die Aspekte unterstützen. Nach unserem besten Wissen gibt es keine weiteren Quellen zu solchen Fragen. Darüber hinaus enthält die Methodik typische Konzepte und Entscheidungsrichtlinien für jeden Aspekt. Diese erheben jedoch keinen Anspruch auf Vollständigkeit aufgrund der Vielzahl der verfügbaren Konzepte für die Umsetzung.

Schritt 1: Szenario identifizieren. Das Verständnis der zentralen Anforderungen des Anwendungsszenarios des Data Lake ist eine wichtige Voraussetzung für alle folgenden Architekturentscheidungen. Darum sollten zunächst die folgenden Fragen beantwortet werden: 1) Welche Art von Daten werden im Data Lake verwaltet? Was sind ihre Eigenschaften? Diese Fragen fokussieren auch Datensicherheits- & Privatheitsanforderungen. 2) Welche zeitlichen Anforderungen sind mit den Daten und ihrer Nutzung assoziiert (z. B. Echtzeit oder stündliche Updates)? 3) Wie werden Daten verwendet (z. B. rein fortgeschrittene Analysen oder auch Reporting)? Während diese drei Fragen ausreichend für die weitere Definition der Data-Lake-Architektur sind, kann

es sein, dass darüber hinaus andere Punkte in der Datenverwaltung beachtet werden müssen, etwa dass Daten nur im europäischen Raum gespeichert werden dürfen. Solche weiterführenden Anforderungserhebungen sind ebenfalls in diesem Schritt durchzuführen.

Schritt 2: Datenfluss entwerfen. Um ein passendes Datenflusskonzept zu definieren ist insbesondere Frage 2 aus Schritt 1 (welche zeitlichen Anforderungen sind mit den Daten und ihrer Nutzung assoziiert) relevant. Sollen Daten sowohl in Echtzeit als auch in größeren Zeitintervallen verarbeitet werden, sollten sowohl Stapel- als auch Datenstromverarbeitung verwendet werden. Hybride Verarbeitungsarchitekturen umfassen beides, wie beispielsweise die Lambda-Architektur [MW15] oder BRAID [GSSM18]. Eine Leitfrage zu dieser Entscheidung ist: Sind Stapel- und Datenstromverarbeitung unabhängig voneinander? Falls sie unabhängig sind, ist die Lambda-Architektur eine passende Lösung, da hier die Stapelverarbeitung und die Datenstromverarbeitung komplett getrennt in unterschiedlichen Systemen ausgeführt werden. Falls aber Daten und auch Ergebnisse zwischen Stapel- und Datenstromverarbeitung ausgetauscht werden sollen, bietet BRAID die benötigte Funktionalität, da hier über einen gemeinsamen Speicher ein Austausch zwischen beiden Verarbeitungsarten stattfinden kann.

Schritt 3: Datenorganisation entwerfen. Die Datenorganisation fokussiert sich auf das effiziente Management von Daten für unterschiedliche Nutzung. So werden Daten etwa für verschiedene Anwendungsfälle aufbereitet oder bereinigt, sodass diese schneller ausgeführt werden können. Dies stellt auch eine Unterstützung für Nutzergruppen dar, welche wenig Erfahrung in der Datenaufbereitung haben. Diese aufbereiteten Daten werden in unterschiedlichen Segmenten des Data Lake abgelegt. Ein passend segmentierter und strukturierter Data Lake bietet effizienteren Zugriff und Nutzung als ein unsegmentierter. Allerdings erhöht eine solche Segmentierung die Komplexität des Data Lake. Um ein passendes Konzept für diesen Aspekt zu wählen, ist Frage 3 aus Schritt 1 (wie werden Daten verwendet) von hoher Wichtigkeit. Abbildung 5.5 stellt einen möglichen Entscheidungsprozess für den Datenorganisationsaspekt dar, inklusive einiger Leitfragen. Die uns bekannten Optionen, aus denen ausgewählt wird, sind: keine Segmentierung, semantische Data Meshes [Deh19], Inmons Data-Pond-Architektur [Inm16] oder Zonenmodelle (z. B. [GGH+20b]).

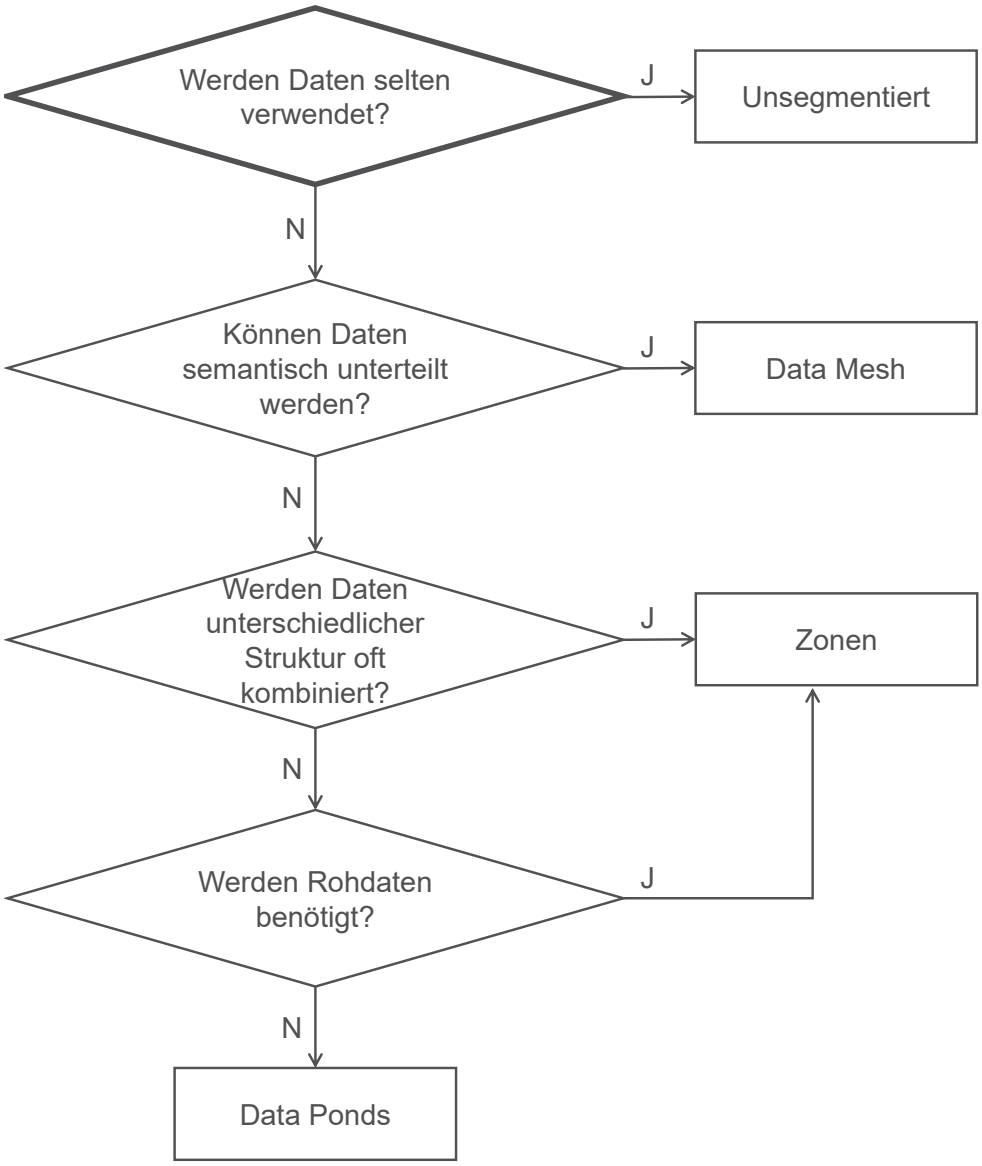


Abbildung 5.5: Der Entscheidungsprozess zur Auswahl eines passenden Konzepts für Datenorganisation. Kombinationen aus Konzepten sind aus Platzgründen nicht enthalten [GGH+21].

Werden Daten nur selten zugegriffen, stellt eine unsegmentierte Datenorganisation die einfachste Lösung dar, da die Vorteile einer Segmentierung ihren Aufwand in diesem Fall nicht aufwiegen. Können Daten semantisch unterteilt werden, so eignet sich ein Data Mesh [Deh19]. In Inmons Data Ponds werden Daten unterschiedlicher Struktur voneinander isoliert, indem sie in separierten Data Ponds gespeichert werden [Inm16]. Auch werden Daten aus dem Raw Data Pond gelöscht, sobald sie für eine Nutzung aufbereitet wurden. Die Rohdaten sind somit nicht mehr verfügbar. Diese Organisation sollte daher

nur gewählt werden, wenn Daten unterschiedlicher Struktur selten kombiniert werden und wenn Rohdaten nicht benötigt werden. Es ist anzumerken, dass es unter Umständen andere passende Konzepte für Datenorganisation gibt, sowie Kombinationen dieser Konzepte. Zudem benötigen einige der genannten Konzepte weitere Spezifizierung, da es z. B. mehrere Varianten von Zonenmodellen gibt [GGH+20b]. Konzepte für Datenqualität sind in den meisten Zonenmodellen und in der Data-Pond-Architektur enthalten. Datensicherheit & Privatheit dagegen sind nur Teil von wenigen Zonenmodellen (z. B. [GGH+20b; Gor16]) und nicht in der Data-Pond-Architektur. Andere Datenorganisationskonzepte betrachten weder Datenqualität noch Datensicherheit & Privatheit.

Schritt 4: Datenspeicher entwerfen. Die Konfiguration eines Datenspeicherkonzepts ist abhängig von der Art der Daten, die verwaltet werden sollen (Frage 1 von Schritt 1) und wie sie verwendet werden (Frage 3 von Schritt 1). Beispielhafte Leitfragen für diesen Schritt sind: Sind mehrere Arten von Speichersystemen nötig? Welche Speichersysteme können die Eigenschaften der Daten unterstützen? Zum Beispiel bieten relationale Datenbanken die passende Unterstützung für strukturierte Daten oder Graphdatenbanken für stark verknüpfte Daten. Weitere Entscheidungsunterstützung kann z. B. in [GWFR17] gefunden werden. Wenn die verwalteten Arten der Daten weit gefächert sind, ist möglicherweise eine Kombination von Speichersystemen passend. Wenn Data Ponds zur Datenorganisation gewählt wurden kann jeder Pond mit einem anderen System realisiert werden, z. B. ein relationales System für den Analog Data Pond und ein Dateisystem für den Textual Data Pond. Es ist notwendig, Datensicherheits- & Privatheits- sowie Datenqualitätsanforderungen bei der Wahl des Datenspeichers zu berücksichtigen, da verschiedene Arten von Datenspeicher unterschiedliche Grade von Konsistenz, Randbediungen, etc. unterstützen.

Schritt 5: Infrastruktur definieren. In diesem Schritt werden die definierten Datenspeicher- und Datenflusskonzepte verwendet, um eine passende Infrastruktur für den Data Lake auszuwählen. Speichersysteme und Verarbeitungswerkzeuge entwickeln sich konstant weiter und die Anforderungen an Infrastruktur sind vielfältig. Daher werden keine Details zur Entscheidungsunterstützung für Infrastruktur in dieser Arbeit geboten. Leitfragen sind allerdings: Welche Aufnahmeraten werden benötigt? Werden Indizes oder Fremd-

schlüssel benötigt? Welche Lese-/Schreibperformanz sollte verfügbar sein? Die Infrastruktur sollte im Einklang mit den Antworten zu diesen Fragen gewählt werden.

Schritt 6: Datenmodellierung entwerfen. Die Antworten zu Fragen 1 und 3 von Schritt 1 (welche Daten werden verwaltet und wie werden sie genutzt) sind von hoher Wichtigkeit um ein Konzept für Datenmodellierung auszuwählen, da sie festlegen, welche Datenmodelle passend sind. Beispielleitfragen für diesen Schritt sind: Wie sollten strukturierte und semi-strukturierte Daten modelliert werden? Zum Beispiel kann Data Vault [Lin12] verwendet werden, um diese Daten in Data Lakes zu modellieren [GGH+19b]. Hierbei werden Daten in einer dreiteiligen Struktur organisiert, welche aus Hubs (repräsentieren Geschäftsobjekte), Links (repräsentieren Beziehungen) und Satellites (enthalten zusätzliche Daten zu Geschäftsobjekten oder Beziehungen) bestehen. Data Vault bietet für Data Lakes Flexibilität, Skalierbarkeit und unterstützt anwendungsfallunabhängige Datenmodellierung und agile Softwareentwicklung [GGH+19b]. Da Data Vault allerdings nicht nativ für Data Lakes entwickelt wurde, stellen sich auch einige Herausforderungen in der Anwendung. So bleibt beispielsweise unklar, wie genau Internet of Things (IoT)-Daten oder die Ergebnisse von Analysen in Data Vault gespeichert werden können. Hierfür müssen innerhalb des Data Lake weitere Modellierungsvorschriften definiert und ihre Einhaltung überprüft werden [GGH+19b]. Weitere Fragen sind: Wie können Daten über Systeme hinweg verbunden werden? Wie können unstrukturierte Daten mit strukturierten Daten integriert werden? Mögliche Antworten zu diesen Fragen sind Data Droplets [Hou17] oder Link-Based Integration [GSM14]. Werden Data Droplets verwendet, so wird jedes Objekt im Data Lake, etwa ein Dokument, als ein Graph aufgefasst. Dieser Graph enthält alle relevanten Daten zu diesem Objekt. Durch Beziehungen zwischen Objekten werden diese Einzelgraphen zu immer größeren Graphen zusammengeschlossen [Hou17]. Diese Modellierungstechnik kann für Daten beliebiger Struktur angewandt werden. In der Link-Based-Integration dagegen werden strukturierte und semi-strukturierte Daten aus Tabellen mithilfe von Links mit unstrukturierten Daten in einem anderen Speichersystem verbunden. Die Links werden dabei in einem eigenständigen Link-Speicher verwaltet und als Graphen realisiert [GSM14]. Darüber hinaus können Integrations-Frameworks

wie etwa SoFIA [CMB+16] verwendet werden. Falls Zonen für die Datenorganisation verwendet werden, muss die Datenmodellierung für jede Zone separat definiert werden. Typischerweise enthält eine Zone Rohdaten, die aus den Datenquellen repliziert werden, während eine andere Zone Daten in einem standardisierten oder sogar Use-Case-abhängigen Format (z. B. dimensional) enthält. Anforderungen an Datensicherheit & Privatheit sowie Datenqualität müssen angesprochen werden, z. B. durch separate Tabellen für sensitive Daten oder Datenmodelle, die Daten konsolidieren.

Schritt 7: Metadaten als Unterstützer entwerfen. Die Leitfrage, um Metadaten als Unterstützer zu konfigurieren, ist: Welche Information wird benötigt, um Daten sinnvoll zu verwalten? Dies beinhaltet 1) Metadaten, die benötigt werden um die Konzepte aus anderen Aspekten widerzuspiegeln (z. B. Metadaten, die beschreiben in welcher Zone Daten liegen) und 2) Metadaten, die für die allgemeine Funktion des Data Lake benötigt werden (z. B. Information zur Datenherkunft, Zugriffsoperationen oder das Datum des letzten Zugriffs). Manche Metadaten werden für die Ausführung von Datenprozessen aus Schritt 8 benötigt. Darum ist es unter Umständen notwendig, diesen Schritt während der Definition von Datenprozessen erneut zu besuchen. Es ist dennoch sinnvoll, das Design von Metadaten als Unterstützer zuerst durchzuführen, da die Datenprozesse so definiert werden müssen, dass diese Metadaten erfasst und gespeichert werden. Schritt 7 beinhaltet auch Metadaten für Datensicherheit & Privatheit und Datenqualität, wie Sicherheitsklassifikationen, ein geplantes Löschdatum oder bekannte Qualitätsprobleme. Da die Metadaten als Unterstützer, die in diesem Schritt identifiziert werden, sich unter Umständen je nach Anwendungsszenario stark unterscheiden, ist es unmöglich eine Entscheidungsleitlinie zu bieten. Die Wahl eines flexiblen Metadatenmanagementmodells wie etwa HANDLE [EGG+20] ist vorteilhaft, da es auch noch später angepasst und erweitert werden kann. Falls Metadaten in den Schritten 1-6 noch nicht als Daten betrachtet wurden, werden diese Lücken in Schritt 7 gefüllt. Metadaten, genau wie andere Daten, benötigen Infrastruktur-, Datenspeicher-, Datenfluss-, Datenmodellierungs- und Datenorganisationskonzepte.

Schritt 8: Datenprozesse entwerfen. Aus Platzgründen können keine detaillierten Leitlinien für die Datenprozesskonfiguration geboten werden. Einige Leitfragen für diesen Schritt sind: Wie werden Daten innerhalb des Data Lake

bewegt und verarbeitet? Was ist der Datenlebenszyklus? Die meisten Konzepte zur Datenorganisation beinhalten passende Datenprozesskonzepte, z. B. wie sich Daten zwischen Zonen oder Ponds (siehe Schritt 3) bewegen und verarbeitet werden [GGH+20b; Inm16]. Diese Prozesse müssen angepasst und erweitert werden, um zu den Konzepten aus den anderen Aspekten zu passen. Falls sich herausstellt, dass Datenprozesse weitere Metadaten benötigen, wird an dieser Stelle zu Schritt 7 zurückgegangen. Auch die Definition von Prozessen für die Erfassung und Extraktion von Metadaten findet in diesem Schritt statt. Datenprozesse für Datensicherheit & Privatheit, wie z. B. Prozesse zum Zugriff auf sensitive Daten, und Datenqualität müssen sinnvoll ausgewählt werden, um zum Anwendungsszenario zu passen. Der Data-Wrangling-Prozess [TSRC15] und existierende Lebenszyklusmanagementprozesse können als Grundlage für die Konfiguration der Datenprozesse verwendet werden. Zur Modellierung

Schritt 9: Metadaten als Funktion entwerfen. Im finalen Schritt sind all die Funktionalitäten enthalten, die über die reine Beschreibung von Daten für die Funktionalität des Data Lake hinausgehen. Metadatenmanagementsysteme wie Data Catalogs [CJL+15] oder Datenmarktplätze [MSLV13] bieten Funktionalität, die über den Rahmen von Metadaten als Unterstützer hinausgehen, wie beispielsweise semantischen Datenzugriff oder Kaufoptionen. Da diese zusätzlichen Funktionalitäten nur mit detailliertem Wissen zur restlichen Data-Lake-Architektur implementiert werden können, wird dieser Schritt als letztes ausgeführt. Dieser Teil der Architekturkonfiguration kann sehr frei ausgeführt werden. Eine zugehörige Leitfrage ist: Welchen zusätzlichen Vorteil können Metadaten bieten?

5.5 Zusammenfassung

In diesem Kapitel wurde das DLAF vorgestellt. Durch die Sammlung notwendiger Data-Lake-Aspekte sowie die Definition der oben beschriebenen Methodik bietet das DLAF einen systematischen Leitfaden zur Definition einer Data-Lake-Architektur. So kann für ein beliebiges Anwendungsszenario von Data Lakes eine vollständige Data-Lake-Architektur definiert werden. Eine detaillierte Evaluation des DLAF und der enthaltenen Methodik erfolgt in Abschnitt 9.1 auf Grundlage einer prototypischen Implementierung.

Im Rahmen dieser Arbeit können aus Zeit- und Platzgründen nicht alle Aspekte des DLAF näher betrachtet werden. Die Betrachtung wird daher auf die für die Forschungsziele notwendigen Aspekte beschränkt. Für die Erreichung von Forschungsziel Z3 ist insbesondere der Aspekt der Datenorganisation von großer Wichtigkeit. Dieser wird darum im Folgenden detailliert diskutiert.

Kapitel 6

Metamodell für Zonen

Insbesondere der Aspekt der Data-Lake-Organisation (siehe Abschnitt 5.2) bietet Unternehmen weitreichendes Potential zur Steigerung der Effizienz des Data Lake. Konzepte zur Data-Lake-Organisation beschreiben, wo Daten gefunden werden und in welchem Zustand sie vorliegen. Dieser Aspekt ist von zentraler Wichtigkeit zur Unterstützung verschiedener Nutzergruppen und Analysen, da durch eine passende Organisation des Data Lake benötigte Daten einfacher aufzufinden sind. Eine nähere Betrachtung der Data-Lake-Organisation ist daher insbesondere für Forschungsziel Z3 notwendig. Die Verwaltung bereits vorverarbeiteter Daten bietet hier die größten Potentiale zur Erhöhung der Effizienz und Kosteneinsparung, da das Aufbereiten der Daten die zeitintensivste Teilaufgabe bei der Datenanalyse darstellt [Pre16]. Zudem können durch die Bereitstellung vorverarbeiteter Daten Nutzer*innen ohne Erfahrung bei der Datenaufbereitung unterstützt werden und Analysen schneller ausgeführt werden. Daher liegt der Fokus für die Erfüllung von Forschungsziel Z3 auf Datenorganisationskonzepten, die eben diese Verwaltung ermöglichen. Inmons Data Ponds [Inm16] sowie Zonenmodelle [Mad15; Sha18; ZDB+15; Gor16; RZ19] sind genau solche Konzepte. Es zeigt sich allerdings, dass das Data-Pond-Konzept für Data Lakes ungeeignet ist, da hier Rohdaten verloren gehen (siehe Abschnitt 6.1). Der Fokus liegt im Folgenden darum auf Zonenmodellen.

In diesen Modellen werden Daten in unterschiedlichen Verarbeitungsgraden in so genannten Zonen verwaltet und verfügbar gemacht.

Bei der Betrachtung existierender Zonenmodelle stellte sich heraus, dass sich trotz zahlreicher Unterschiede der grundlegende Aufbau einer Zone über alle Architekturen hinweg gleicht. Dieses Kapitel erstellt darum ein Metamodell für Zonen aus Arbeiten zu Zonen, um eine strukturierte Diskussion der existierenden Zonenmodelle zu ermöglichen (siehe Forschungsbeitrag B3.2). Dazu stellt Abschnitt 6.2 die Attribute des Metamodells vor, während Abschnitt 6.3 die Beziehungen zwischen Zonen vorstellt. Abschließend bewertet Abschnitt 6.4 das entstandene Metamodell, indem existierende Zonenmodelle in das Metamodell eingeordnet werden.

Dieses Kapitel ist eine abgewandelte Fassung einer vorangegangenen Veröffentlichung der Autorin [GGH+20b].

6.1 Data Ponds und Zonen

Wie bereits erwähnt existieren zwei Ansätze, um vorverarbeitete Daten in Data Lakes zu verwalten: Die Data-Pond-Architektur und Zonenmodelle. Die folgenden Absätze geben eine kurze Übersicht über die Grundlagen beider Ansätze.

Die *Data-Pond-Architektur* [Inm16] von Inmon unterteilt den Data Lake in fünf disjunkte Ponds: Raw Data Pond, Analog Data Pond, Application Data Pond, Textual Data Pond und Archival Data Pond. Daten sind zu jedem Zeitpunkt nur in einem der genannten Ponds verfügbar und werden verarbeitet wenn sie sich durch die Ponds bewegen. So gehen die originalen Rohdaten verloren. Dies widerspricht dem Konzept von Data Lakes (siehe Abschnitt 2.1) und verhindert flexible Analysen, da die im Data Lake vorhandenen Daten nicht mehr für beliebige Anwendungen verwendet werden können. Zudem werden Daten aufgrund ihrer Charakteristika in isolierte Ponds unterteilt. Dies verhindert eine Integration der Daten, welche für die praktische Anwendung unabdingbar ist. Aufgrund dieser Einschränkungen ist die Data-Pond-Architektur zur Erfüllung der Anforderungen aus Abschnitt 4.1 ungeeignet.

Zonenmodelle stellen Daten simultan in mehreren Verarbeitungsgraden zur Verfügung. Zur Verwaltung dieser Daten werden Zonen verwendet, die jeweils

beschreiben, wie Daten in ihnen aufbereitet sind. Durch die Verwaltung von Daten in verschiedenen Graden der Verarbeitung können unterschiedlichen Nutzer*innen und Anwendungsfällen Daten in passendem Format (z. B. roh, bereinigt oder integriert) angeboten werden. Darüber hinaus können Daten in unterschiedlichen Zonen auch unterschiedlich streng verwaltet werden, z. B. mit verschiedenen Zugriffsberechtigungen, Anforderungen an Datenqualität und Verantwortlichkeiten. Je nach Nutzer*innen und Anwendungsfall kann dann auf die Daten im passendsten Verarbeitungsgrad zugegriffen werden. So erlauben Zonenmodelle das Teilen und Wiederverwenden von vorverarbeiteten Daten zwischen Anwendungsfällen. Zonen ähneln dabei den Ebenen, wie sie in Data Warehouses verwendet werden (z. B. in [SJWW16]), allerdings bewegen sich Daten nicht unbedingt durch alle Zonen und können manchmal sogar wieder zurück wandern.

Es existiert eine Vielzahl unterschiedlicher Varianten von Zonenmodellen [Mad15; Sha18; ZDB+15; Gor16; RZ19]. Diese Varianten unterscheiden sich nicht nur in den Zonen, die sie umfassen, sondern auch in der Anzahl der Zonen, den unterstützten Nutzergruppen (nur Data Scientists vs. mehrere Nutzergruppe) und ihrem Fokus (Verarbeitung [Mad15] vs. Governance [Gor16]). Die grundlegende Idee bleibt allerdings gleich: Unterschiedliche Zonen beinhalten Daten in verschiedenen Verarbeitungsgraden, z. B. roh oder vorverarbeitet. Dazu beschreibt jede Zone eine Menge an Eigenschaften, die Daten in ihr haben (z. B. Daten sind bereinigt und in einem standardisierten Format). Im Gegensatz zur Data-Pond-Architektur sind Zonen nicht disjunkt. Daten können von Zone zu Zone kopiert werden oder eine Zone kann Sichten auf Daten einer anderen Zone enthalten. So sind Rohdaten immer verfügbar in was meist als „Raw Zone“ bezeichnet wird. Während die Merkmale der Raw Zone im allgemeinen zwischen Zonenmodellen gleich sind (Daten werden persistent in ihrem Rohformat gespeichert), gilt dies nicht für die anderen Zonen der unterschiedlichen Modelle.

6.2 Attribute des Metamodells

Das Metamodell für Zonen wird in zwei Teilen vorgestellt: 1) die Attribute der Zonen (dieser Abschnitt) und 2) die Beziehungen einer Zone innerhalb und

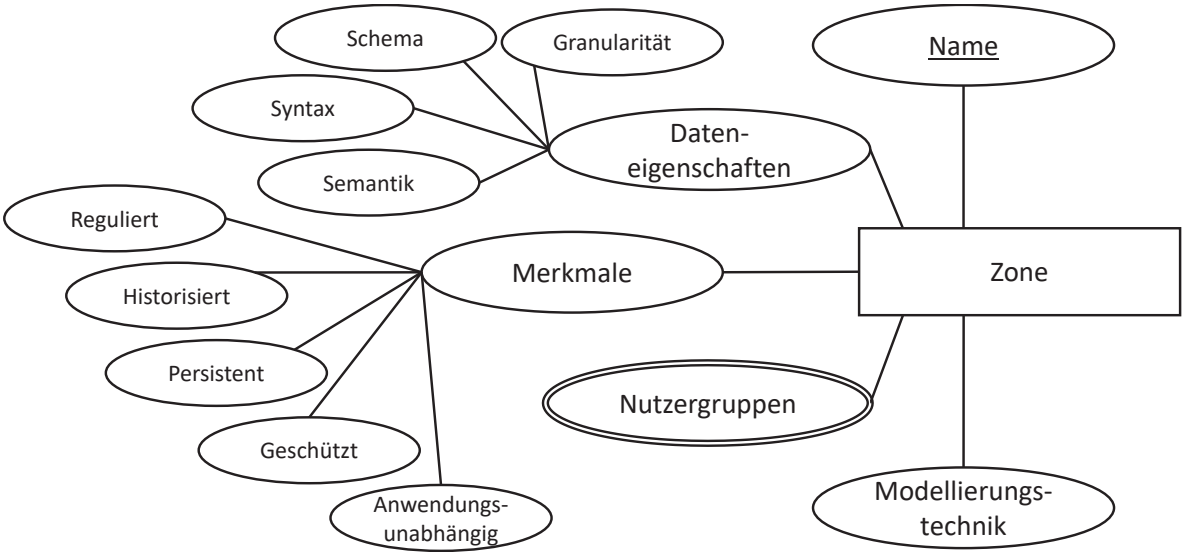


Abbildung 6.1: Die Attribute des Metamodells für Zonen [GGH+20b]. „Nutzergruppe“ ist ein mehrwertiges Attribut.

außerhalb des Zonenmodells (siehe Abschnitt 6.3). Abbildung 6.1 zeigt die erste Hälfte des Metamodells für Zonen als Entity-Relationship (ER)-Diagramm. Dabei sind Zonen als Entitäten modelliert. Innerhalb des Zonenmodells wird jede Zone mit einem eindeutigen Namen identifiziert.

Eine Zone definiert mehrere Dateneigenschaften, die Daten in dieser Zone haben. Jede Eigenschaft bezieht sich auf eine von vier Dimensionen: Granularität, Schema, Syntax und Semantik. Granularität beschreibt, ob Daten roh oder aggregiert vorliegen, z. B. durch Key-Performance-Indikator (KPI)-Berechnung. Schema bezieht sich auf die Struktur der Daten, welche sich durch das Hinzufügen weiterer Felder oder Beziehungen ändern kann. Syntax bezieht sich darauf, ob Daten syntaktisch verändert werden, z. B. durch die Umwandlung von Datentypen. Semantik letztendlich bezieht sich auf die Bedeutung der Daten, welche sich durch z. B. das Entfernen semantischer Fehler, wie Ausreißer, verändern kann.

Zusätzlich zu diesen Dateneigenschaften hat jede Zone Merkmale, die das Wesen der Zone beschreiben. Aus der Literatur (z. B. [Sha18; Gor16]) wurden *reguliert*, *historisiert*, *persistent*, *geschützt* und *anwendungsunabhänig* als mögliche Merkmale für Zonen identifiziert. Reguliert beschreibt, ob eine Zone durch die zentrale IT verwaltet wird und darum der unternehmensweiten Governance unterworfen ist. Historisiert bedeutet, dass Änderungen in den Daten in der

entsprechenden Zone nachvollziehbar sind. Persistent beschreibt, ob Daten in dieser Zone für eine lange bis potentiell unendliche Zeitspanne gespeichert werden im Gegensatz zu temporärem Speicher. Geschützt beschreibt, ob Daten in einer Zone über den Standard hinaus geschützt werden, z. B. durch Verschlüsselung oder besonders strenge Zugriffskontrollen. Anwendungsunabhängig bezieht sich darauf, ob Daten in einer Zone für bestimmte Anwendungsfälle oder Gruppen von Anwendungsfällen verarbeitet wurden, oder ob sie flexibel nutzbar sind. Madsens Zonenmodell [Mad15] nennt zudem *unveränderbar* als ein Zonenmerkmal. Allerdings zeigt sich in der Praxis, dass die Speicherung aller Daten ohne jegliche Änderungen und ohne Archivierung oder Löschung zu Problemen bei der Speicherung und Verwaltung führt. Zudem schreiben einige rechtliche Regularien, wie beispielsweise die DGSVO, explizit vor, dass Daten veränderbar und löschbar sein müssen. Darum wird dieses Merkmal nicht miteinbezogen.

Eine oder mehrere Nutzergruppen interagieren mit jeder Zone. Möglich sind menschliche Nutzer*innen (Data Scientists, Domänenexpert*innen, Geschäftsnutzer*innen) und nichtmenschliche Nutzer*innen (Systeme, Prozesse). Jede Zone hat mindestens eine Nutzergruppe, nämlich die Prozesse, die Daten in die Zone einfügen.

Als letztes hat jede Zone eine assoziierte Modellierungstechnik, d. h. eine Beschreibung davon, wie ein bestimmtes Schema erreicht werden kann, wie beispielsweise dimensionale Tabellen. Hierbei zählt auch „kein vordefiniertes Schema“ als ein mögliches Schema. Innerhalb von Data Lakes sollte Datenmodellierung nicht vernachlässigt werden, da dies zu Problemen mit der Datenqualität, -Verständlichkeit oder -Integration führen kann [Sti14]. Beispiele für mögliche Modellierungstechniken ist das Kopieren von Quellsystemformaten, das Organisieren von Daten in flachen Dateien oder Data Vault, welcher sich für Data Lakes eignet [GGH+19b].

6.3 Beziehungen im Metamodell

Der zweite Teil des Metamodells für Zonen sind die Beziehungen innerhalb und außerhalb des Zonenmodells. Abbildung 6.2 zeigt die Beziehungen als ER-Diagramm. Zonen haben sowohl Beziehungen mit anderen Zonen als auch

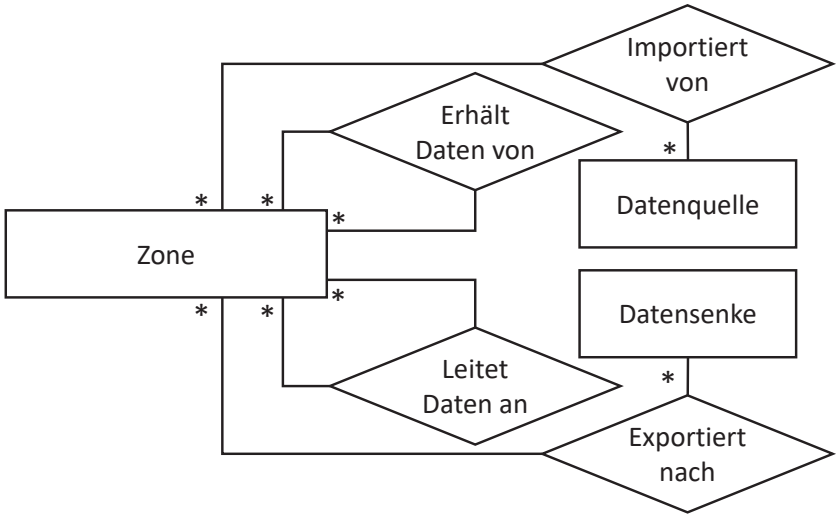


Abbildung 6.2: Die Beziehungen des Metamodells für Zonen [GGH+20b]

Systemen außerhalb des Data Lake. Eine Zone erhält Daten aus keiner oder mehreren anderen Zonen und kann Daten an null oder mehrere Zonen weiterleiten. Dies erlaubt Datentransfer zwischen Zonen. Zudem können Zonen Daten von externen Datenquellen importieren und Daten zu externen Datensenken exportieren, z. B. operative Systeme, Datenströme oder Dateisysteme.

6.4 Bewertung des Metamodells

In diesem Abschnitt wird das entstandene Metamodell für Zonen bewertet. Hierfür wird es auf existierende Zonenmodelle angewandt, um deren Eigenschaften und Aufbau systematisch darzustellen. Bei näherer Betrachtung vorhandener Zonenmodelle wird deutlich, dass kein einheitlich anerkanntes Zonenmodell existiert. Zwar gibt es besonders in den ersten Zonen (typischerweise Raw Zone genannt) starke Ähnlichkeiten, danach allerdings unterscheiden sich die Modelle insbesondere, was die Merkmale, Nutzergruppen und Modellierungstechniken betrifft. Das erarbeitete Metamodell für Zonen ermöglicht den Vergleich der unterschiedlichen Ansätze aus der Literatur. Durch eine Einteilung der vorhandenen Zonenmodelle anhand des Metamodells für Zonen können deren Gemeinsamkeiten und Unterschiede herausgearbeitet werden. Im Folgenden liegt der Fokus dabei auf den Attributen, da die Beziehungen innerhalb des Zonenmodells und außerhalb für den Vergleich von geringer Wichtigkeit sind.

Im Rahmen einer Literaturrecherche wurden fünf Zonenmodelle gefunden, die die Verwaltung vorverarbeiteter Daten unterstützen: Goreliks Zonenmodell [Gor16] (siehe Tabelle 6.1), Madsens Data Architecture [Mad15] (siehe Tabelle 6.2), Ravats Data Lake Functional Architecture [RZ19] (siehe Tabelle 6.3), Sharmas Data Lake Reference Architecture [Sha18] (siehe Tabelle 6.4) und Zikopoulos Data Zones Model [ZDB+15]. Letzteres ist von dem Vergleich ausgenommen, da die Beschreibung nicht genügend Details zur Einordnung im Metamodell für Zonen bietet.

Wie der Vergleich von Tabelle 6.1 bis Tabelle 6.5 zeigt, erlaubt die Einteilung existierender Zonenmodelle in das Metamodells für Zonen eine einheitliche und systematische Beschreibung ihrer Eigenschaften.

6.5 Zusammenfassung

Dieses Kapitel beschreibt das entwickelte Metamodell für Zonen. Es umfasst mögliche Attribute für Zonen, sowie die möglichen Interaktionen zwischen Zonen und externen Datenquellen und -senken. Das Metamodell für Zonen bietet den Vorteil, dass zwischen Dateneigenschaften und Merkmalen der Zonen unterschieden wird. Auch ist durch das Metamodell vorgegeben, welche Dimensionen für die Dateneigenschaften von Interesse sind. Ebenso steht eine begrenzte Menge von Merkmalen für die Zonenbeschreibung zur Verfügung. So können Zonenmodelle untereinander verglichen und Gemeinsamkeiten und Unterschiede identifiziert werden. Auch kann das Metamodell für Zonen für die systematische Definition weiterer Zonenmodelle verwendet werden.

Name	Dateneigenschaften	Merkmale	Nutzergruppen	Modellierungstechnik
Raw Zone	Granularität, Schema, Syntax und Semantik gleich zum Quellsystem	Historisiert, Persistent, Anwendungsunabhängig	Technische Entwickler	Unklar
Gold Zone	Daten sind aufbereitet, Granularität, Schema, Syntax und Semantik können sich ändern	Historisiert, Persistent, Anwendungsabhängig	Datenanalysten	Durch SQL abfragbar
Work Zone	Siehe Gold Zone	Siehe Gold Zone	Data Scientists, Entwickler	In Projekten organisiert
Sensitive Zone	Beliebige Granularität, Schema, Syntax und Semantik	Reguliert, Historisiert, Persistent, Geschützt	Streng limitiert	Unklar

Tabelle 6.1: Das Metamodell für Goreliks Zonenmodell [Gor16]

Name	Dateneigenschaften	Merkmale	Nutzergruppen	Modellierungstechnik
Collection Zone	Granularität, Schema, Syntax und Semantik gleich zum Quellsystem	Persistent, Anwendungsunabhängig	Unklar	Aus Quellsystemen kopiert
Management Zone	Daten sind aufbereitet, Granularität, Schema, Syntax und Semantik können sich ändern	Persistent, Anwendungsunabhängig	Unklar	Standardisiert
Delivery Zone	Siehe Collection Zone	Persistent, Anwendungsabhängig	Unklar	Anwendungsabhängig
Transient Zone	Beliebig, da hier mit den Daten gearbeitet wird	Anwendungsabhängig	Unklar	Anwendungsabhängig

Tabelle 6.2: Das Metamodell für Madsens Data Architecture [Mad15]

Name	Dateneigenschaften	Merkmale	Nutzergruppen	Modellierungstechnik
Raw Data Zone	Granularität, Schema, Syntax und Semantik gleich zum Quellsystem	Persistent, Historisiert, Anwendungsunabhängig	Unklar	Aus Quellsystemen kopiert
Process Zone	Daten sind aufbereitet, Granularität, Schema, Syntax und Semantik können sich ändern	Anwendungsabhängig	Unklar	Unklar
Access Zone	Siehe Process Zone	Persistent, Anwendungsabhängig	Beliebige Nutzer*innen	Anwendungsabhängig
Governance Zone	Unklar	Reguliert, Geschützt	Unklar	Unklar

Tabelle 6.3: Das Metamodell für Ravats Data Lake Functional Architecture [RZ19]

Name	Dateneigenschaften	Merkmale	Nutzergruppen	Modellierungstechnik
Transient Landing Zone	Granularität, Schema, Syntax und Semantik gleich zum Quellsystem, außer für legale Konformität	Reguliert, Anwendungsunabhängig	Unklar	Aus Quellsystemen kopiert
Raw Zone	Siehe Transient Landing Zone	Reguliert, Historisiert, Persistent, Anwendungsunabhängig	Data Scientists, Business Analysten	Aus Quellsystemen kopiert
Trusted Zone	Daten werden aufbereitet, sodass sie Geschäftsregelungen befolgen. Granularität und Semantik bleiben größtenteils erhalten, Schema und Syntax ändern sich.	Reguliert, Historisiert, Persistent, größtenteils Anwendungsunabhängig	Beliebige Nutzer*innen	Standardisiert

Tabelle 6.4: Das Metamodell für Sharmas Data Lake Reference Architecture [Sha18]

Name	Dateneigenschaften	Merkmale	Nutzergruppen	Modellierungstechnik
Refined Zone	Daten sind aufbereitet, Granularität, Schema, Syntax und Semantik können sich ändern	Reguliert, Historisiert, Persistent, Anwendungsabhängig	Beliebige Nutzer*innen	Anwendungsabhängig
Sandbox	Beliebig	Anwendungsabhängig	Data Scientists	Anwendungsabhängig

Tabelle 6.5: Das Metamodell für Sharmas Data Lake Reference Architecture [Sha18] (cont.)

Kapitel 7

Das Zonenreferenzmodell

Nachdem in Kapitel 6 ausführlich auf die Natur von Zonen und Zonenmodelle eingegangen wurde, gilt es nun, Forschungsziel Z3 zu adressieren. Wie bereits in Kapitel 6 erwähnt, eignen sich Zonenmodelle [Mad15; Sha18; ZDB+15; Gor16; RZ19] hierfür besonders.

Bei näherer Betrachtung existierender Ansätze für Zonenmodelle stellt sich allerdings heraus, dass ihre Beschreibungen vielfältig, vage und inkonsistent sind (siehe Abschnitt 6.4). Es gibt weder ein einheitliches Konzept für zonenbasierte Data Lakes, noch irgendeine Form einer systematischen Bewertung der Ansätze. Die Gegenüberstellung der Zonenmodelle in Abschnitt 6.4 bietet hier einen ersten Ansatz, reicht jedoch noch nicht aus, um das für ein bestimmtes Anwendungsszenario passende Zonenmodell auszuwählen. In der Praxis stellt diese Diversität eine Herausforderung dar, da unklar ist, welche Zonenmodelle sich eignen und wie sie zu implementieren sind.

Dieses Kapitel befasst sich mit diesem Problem. Dazu identifiziert Abschnitt 7.1 zunächst Anforderungen an einen zonenbasierten Data Lake aus unserem Industriebeispiel (vgl. Forschungsbeitrag B3.1). Anhand dieser Anforderungen werden existierende Zonenmodelle in Abschnitt 7.2 evaluiert. Da keines der existierenden Modelle die Anforderungen vollständig unterstützt, wird auf dieser Basis das Zonenreferenzmodell in Abschnitt 7.3 entwickelt, welches

die unternehmenstaugliche Verwaltung vorverarbeiteter Daten erlaubt. „Unternehmenstauglich“ beschreibt dabei, dass das erarbeitete Zonenreferenzmodell Nutzer*innen und Anwendungsfälle unterstützen kann, die für Unternehmen typisch sind (vgl. Forschungsbeitrag B3.3). Das Zonenreferenzmodell nutzt dabei das Metamodell für Zonen aus Kapitel 6 als Basis für eine systematische Definition der einzelnen Zonen. Anschließend stellt Abschnitt 7.4 Implementierungsmuster für Zonenmodelle vor, bevor Abschnitt 7.5 die Methodik für die Umsetzung einer zonenbasierten Datenorganisation beschreibt. Abschnitt 7.6 schließlich schließt dieses Kapitel ab.

Bei diesem Kapitel handelt es sich größtenteils um eine überarbeitete Version einer vorausgegangenen Veröffentlichung der Autorin [GGH+20b].

7.1 Anforderungen an ein Zonenmodell

Für die Herleitung der Anforderungen an ein Zonenmodell werden mehrere reale Anwendungsfälle aus dem Unternehmen des in Abschnitt 3.2 vorgestellten Herstellers genutzt. Nach unseren Erfahrungen spiegeln die Beobachtungen, die in diesem Unternehmen gemacht wurden, die Realität in anderen großen Unternehmen wider. Wie bereits erwähnt handelt es sich bei dem Geschäft des Herstellers um ein diverses, mit verschiedenen Datenanalyseprojekten in zahlreichen unterschiedlichen Domänen. Die Daten für alle diese Projekte werden in einem unternehmensweiten Data Lake verwaltet.

Die Anwendungsfälle, die für die Herleitung der Anforderungen verwendet werden, stammen aus vier unterschiedlichen, jedoch häufig repräsentierten Domänen, nämlich *Finanzen*, *Qualitätsmanagement*, *Produktion* und *Endkundenservice*. Die verwendeten Analysen sind ebenfalls vielfältig und reichen von Reporting bis hin zu fortgeschrittenen Analysen mithilfe von Machine Learning. Die verwendeten Daten bewegen sich auf dem gesamten Spektrum von strukturiert bis unstrukturiert. Daher können diese Anwendungsfälle als repräsentativ für Data-Lake-Anwendungen angenommen werden. Im Folgenden werden diese Anwendungsfälle genauer betrachtet und aus ihnen die konkreten Anforderungen an ein Zonenmodell für einen unternehmensweiten Data Lake abgeleitet.

Das Datenanalyseprojekt der Finanzdomäne fokussiert sich auf die Realisierung von *Reporting und Online Analytical Processing (OLAP)* auf den Daten des Data Lake. Strukturierte ruhende Daten aus mehreren Quellen werden miteinander *integriert*, z. B. aus mehreren Enterprise Resource Planning (ERP)-Systemen. Da die Ergebnisse der Analysen von hoher Wichtigkeit für das Unternehmen sind, müssen die verwendeten Daten vorsichtig *bereinigt* und *reguliert* werden. Bestimmte Analysen, wie die Berechnung von Key-Performance-Indikatoren (KPIs) (z. B. der operative Geldfluss), werden wiederholt ausgeführt und profitieren daher von *vorverarbeiteten* Daten.

Im Fall des Qualitätsmanagements werden Daten aus einer Vielzahl von Quellen verwendet, um Qualitätsdefekte in hergestellten Produkten zu untersuchen. Hierzu werden Fehler-Ursachen-Analysen mithilfe von Schadensberichten ausgeführt, zusammen mit anderen *fortgeschrittenen Analysen*. Darüber hinaus werden ruhende Daten für traditionelles *Reporting und OLAP* verwendet, z. B. um die Anzahl der Defekte einer bestimmten Produktlinie zu ermitteln. Da Daten aus verschiedenen Quellsystemen stammen, müssen sie *integriert* werden. Die Qualität der Daten wird dabei im Quellsystem sichergestellt, weshalb keine separate Bereinigung im Data Lake notwendig ist. Einige der verwendeten Daten sind personenbezogen, z. B. Kundendaten aus Fehlerberichten. Diese müssen daher passend *reguliert* werden, da für personenbezogene Daten besondere Richtlinien gelten, z. B. die DSGVO. Auch dieser Anwendungsfall umfasst Analysen, die wiederholt ausgeführt werden und profitiert darum von *vorverarbeiteten* Daten. Eine zusätzliche Anforderung ist es, dass Data Scientists die Möglichkeit haben sollen, die Ergebnisse ihrer Analysen mit anderen Nutzer*innen zu teilen, indem sie in den Data Lake *zurückgeschrieben* werden. Diese Ergebnisse, wie Transformationen oder Data-Mining-Modelle, können dann für andere Anwendungsfälle wiederverwendet werden.

Das Ziel des Datenanalyseprojekts aus der Produktionsdomäne ist es, Einsichten in den Produktionsprozess von Produkten zu erhalten, bei denen mehrere Teile verschiedener Zulieferer verbaut werden. Ruhende Daten zu den zugelieferten Teilen und Echtzeitdaten von Sensoren aus Maschinen und Messstationen werden für verschiedene Analysen verwendet, von *Reporting und OLAP* auf Produktionsdaten (siehe z. B. [KBLK10]) zu fortgeschrittenen Analysen, wie Machine Learning. Eine hohe Anzahl von Quellsystemen (über 600) sind

Teil dieses Anwendungsfall. Daher müssen die verfügbaren Daten miteinander *integriert* werden. Einige der Daten werden manuell von Arbeitern erfasst, z. B. Fehlerbeschreibungen, während andere sensibel sind, z. B. Daten zu den Arbeitern selbst. Daher ist die *Bereinigung* und *Governance* dieser Daten von hoher Wichtigkeit. Dies gilt auch für die *Vorverarbeitung* einer Teilmenge der Daten, da bestimmte Anwendungsfälle periodisch durchgeführt werden (z. B. die Berechnung von KPIs). Data Scientists sollten zudem die Möglichkeit haben, Daten in den Data Lake *zurückzuschreiben* für die zukünftige Nutzung.

Das Datenanalyseprojekt der Endkundenservicedomäne befasst sich vorrangig mit Echtzeitfelddaten, wie GPS-Daten, die während der Nutzung des Produkts gesammelt werden. Diese werden verwendet, um dem Kunden weitere Services per App anzubieten (z. B. Ortungsservices oder Dashboards). Zudem werden Datenanalysen auf gesammelten, ruhenden Daten ausgeführt, um das Produkt zu verbessern. Da die Daten zu bestimmten Kunden gesammelt werden, können tiefgehende Analysen die Privatheit des Kunden gefährden. Daher müssen Daten geschützt und *reguliert* werden. Für die Verbesserung der Produkte analysieren Data Scientists die gesammelten Daten mit *fortgeschrittenen Methoden*, wie Machine Learning. *Reporting und OLAP* werden ebenfalls ausgeführt. Ergebnisse, die durch die Data Scientists erarbeitet wurden, sollen *zurückgeschrieben* werden, um für weitere Analysen zur Verfügung zu stehen. Auch dieses Projekt umfasst zahlreiche Datenquellen, deren Daten miteinander *integriert* werden müssen. Zudem werden viele der Analysen, wie die Erstellung von Dashboards, periodisch ausgeführt und profitieren somit von *bereinigten* und *vorverarbeiteten* Daten.

Aus der Betrachtung dieser repräsentativen Datenanalyseprojekte werden sieben funktionale Anforderungen (zusätzlich zur Speicherung von Rohdaten) an eine Datenorganisation für einen Data Lake abgeleitet:

- Daten sollten in vorverarbeitetem Zustand verfügbar sein, d. h. sie sind nicht mehr im Rohformat, sondern aggregiert oder gefiltert, um mehrere Anwendungsfälle zu unterstützen.
- Daten sollten in einem bereinigten Format verfügbar sein, d. h. syntaktische Fehler wurden behoben.

Geforderte Funktionalität durch Datenanalyseprojekt	Finanzen	Qualitätsmanagement	Produktion	Endkundenservice
Vorverarbeitung	✓	✓	✓	✓
Bereinigung	✓	X	✓	✓
Integration	✓	✓	✓	✓
Governance	✓	✓	✓	✓
Reporting & OLAP	✓	✓	✓	✓
Fortgeschrittene Analysen	X	✓	✓	✓
Zurückschreiben	X	✓	✓	✓

Tabelle 7.1: Geforderte Verwaltungsfunktionalität der betrachteten Datenanalyseprojekte an einen Data Lake [GGH+20b]

- Daten aus verschiedenen Quellsystemen sollen in einem integrierten Format verfügbar sein, wo sie konsolidiert und verbunden sind.
- Governance für sensible und kritische Daten wird benötigt, d. h. Zugriffe müssen beschränkt, Änderungen nachverfolgt und Qualität gesichert werden.
- Reporting und OLAP soll unterstützt werden, z. B. für die Berechnung von KPIs
- Fortgeschrittene Analysen mit, z. B. Machine Learning, sollen unterstützt werden.
- Um die Ergebnisse anderen Nutzer*innen des Data Lake verfügbar zu machen, soll das Zurückschreiben von Ergebnissen möglich sein.

Tabelle 7.1 stellt all diese Anforderungen dar und wie die verschiedenen Datenanalyseprojekte zu ihnen beitragen. Diese Anforderungen werden genutzt, um im folgenden Abschnitt existierende Zonenmodelle zu evaluieren.

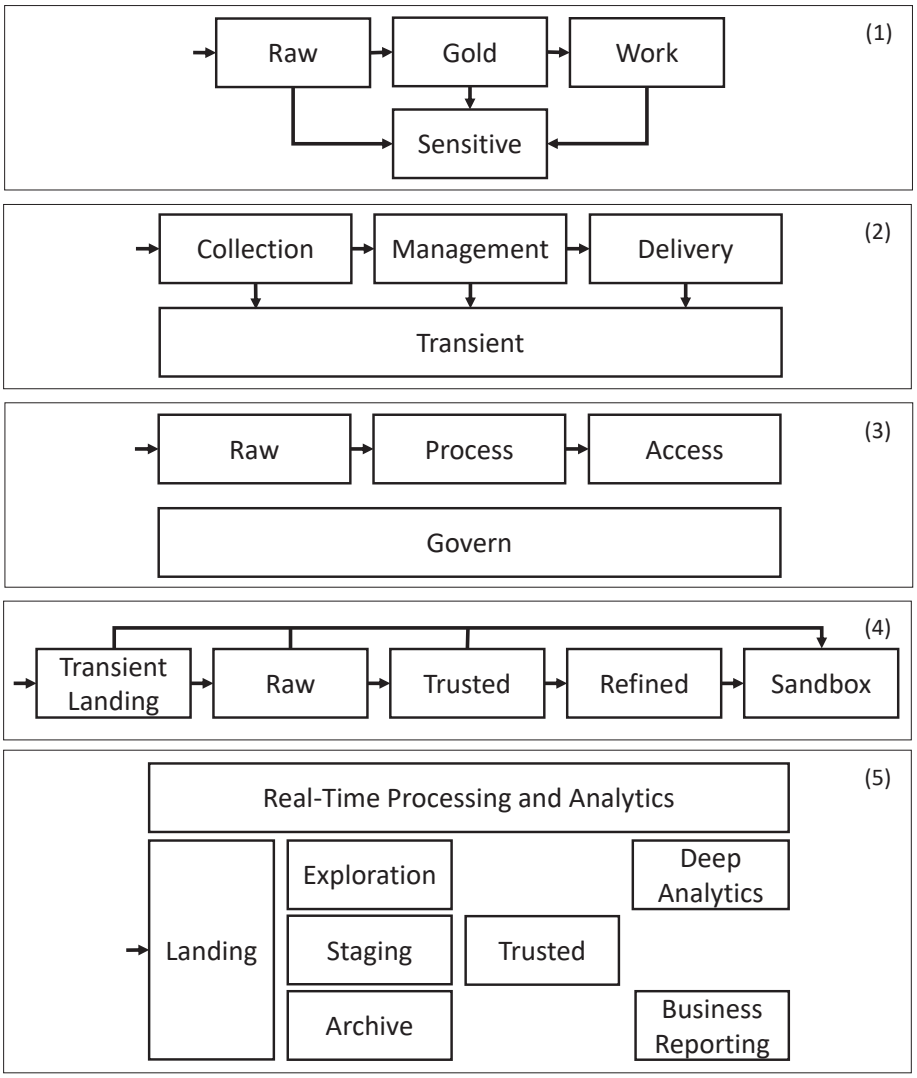


Abbildung 7.1: Übersicht über die fünf verschiedenen Zonenmodelle [GGH+20b]: 1) Gorelik [Gor16], 2) Madsen [Mad15], 3) Ravat [RZ19], 4) Sharma [Sha18] und 5) Zikopoulos [ZDB+15]

7.2 Verwandte Arbeiten

In der Literatur werden verschiedene Zonenmodelle zur Anwendung in Data Lakes vorgestellt. Dieser Abschnitt evaluiert die existierenden Zonenmodelle anhand der in Abschnitt 7.1 abgeleiteten Anforderungen. Zwar können auch Inmons Data Ponds [Inm16] vorverarbeitete Daten verwalten, aufgrusnd der in Abschnitt 6.1 genannten Probleme wird dieses Konzept nicht weiter in Betracht gezogen.

Die Betrachtung der unterschiedlichen Zonenmodelle in Abschnitt 6.4 zeigte, dass sich Zonenmodelle nicht nur in den Zonen unterscheiden, die sie umfassen, sondern auch in der Anzahl der Zonen, den unterstützten Nutzergruppen und ihrem Fokus (Verarbeitung [Mad15] vs. Governance [Gor16]). Die grundlegende Idee allerdings, wie in Kapitel 6 dargestellt, bleibt erhalten: Jede Zone definiert Eigenschaften, die die Daten in ihr haben. Dabei sind die Zonen nicht disjunkt, sondern Daten können in verschiedenen Verarbeitungsgraden in mehreren Zonen vorliegen. Rohdaten sind immer in was meist Raw Zone genannt wird vorhanden. Erneut werden die fünf Zonenmodelle verwendet, die in der Literaturrecherche identifiziert wurden: Goreliks Zonen [Gor16], Madsens Data Architecture [Mad15], Ravats Data Lake Functional Architecture [RZ19], Sharmas Data Lake Reference Architecture [Sha18] und Zikopoulos Data Zones Model [ZDB+15]. Abbildung 7.1 zeigt eine Übersicht dieser fünf Zonenmodelle in einem systematischen Format, um ihre zentralen Unterschiede zu visualisieren.

Neben den in Abschnitt 7.1 beschriebenen funktionalen Anforderungen werden auch zwei methodische Anforderungen bewertet: 1) Den Detailgrad in dem das Zonenmodell beschrieben wird, inklusive Details zur Implementierung, und 2) die Details, die zur Herleitung und Tauglichkeit des Modells gegeben werden, z. B. ein Herleitungsprozess auf Grundlage einer Literaturrecherche oder eine Bewertung des Modells.

Die Ergebnisse dieser Bewertung sind in Tabelle 7.2 und Tabelle 7.3 dargestellt. Ein geklammerter Haken zeigt, dass das Zonenmodell zwar einige der geforderten Eigenschaften aufweist, die Anforderung aber nicht als ausreichend erfüllt angesehen wird. Die folgenden Abschnitte detaillieren unsere Bewertung.

	Funktionalität Zonenmodell	Gorelik [Gor16]	Madsen [Mad15]	Ravat [RZ19]	Sharma [Sha18]	Zikopoulos [ZDB+15]
Funktional	Vorverarbeitung	✓	✓	✓	✓	✓
	Bereinigung	X	✓	X	✓	✓
	Integration	X	✓	X	✓	✓
	Governance	✓	X	(✓)	✓	(✓)
	Reporting & OLAP	✓	✓	✓	X	✓
	Fortgeschrittene Analysen	✓	✓	✓	✓	✓
	Zurückschreiben	✓	X	X	X	X

Tabelle 7.2: Bewertung der existierenden Zonenmodelle anhand der aus den Datenanalyseprojekten geforderten Verwaltungsfunktionalitäten [GGH+20b]
✓- Anforderung erfüllt, (✓) - Anforderung teilweise erfüllt, X - Anforderung nicht erfüllt

	Funktionalität Zonenmodell	Gorelik [Gor16]	Madsen [Mad15]	Ravat [RZ19]	Sharma [Sha18]	Zikopoulos [ZDB+15]
Methodisch	Beschreibungsdetail	(✓)	X	X	(✓)	X
	Herleitungs-/ Bewertungsdetail	X	X	X	X	X

Tabelle 7.3: Bewertung der existierenden Zonenmodelle anhand der aus den Datenanalyseprojekten geforderten Verwaltungsfunktionalitäten (cont.) [GGH+20b]
✓- Anforderung erfüllt, (✓) - Anforderung teilweise erfüllt, X - Anforderung nicht erfüllt

In Goreliks Zonenmodell [Gor16] (siehe Abbildung 7.1 (1)) erlaubt die Gold Zone die Verwaltung vorverareiteter Daten. Es gibt allerdings keine Zone für explizit bereinigte oder integrierte Daten. Die Sensitive Zone erlaubt die Governance-konforme Verwaltung sensibler Daten und unterstützt so Regelkonformität. Reporting und OLAP kann in der Gold Zone ausgeführt werden. Die Work Zone stellt Daten für fortgeschrittene Analysen bereit. Ergebnisse aus der Work Zone können in die Gold Zone für die zukünftige Nutzung zurückgeschrieben werden. Zwar gibt es für jede Zone eine detaillierte Beschreibung, allerdings fehlt es an Implementierungsdetails. Daher ist diese Anforderung nur teilweise erfüllt. Es sind weder Details zur Herleitung noch zur Bewertung gegeben.

In Madsens Data Architecture [Mad15] (Abbildung 7.1 (2)) bietet die Management Zone einen Speicherort für vorverarbeitete, bereinigte und integrierte Daten. Das Modell umfasst keine Informationen zur Governance von Daten. Sowohl Reporting und OLAP als auch fortgeschrittene Analysen werden in zwei unterschiedlichen Zonen unterstützt. Es gibt keine Möglichkeit, Ergebnisse in den Data Lake zurückzuschreiben. Die Zonen werden nur kurz mit wenig Detail beschrieben. Es gibt keinerlei Details zur Herleitung oder Bewertung.

Ravats Data Lake Functional Architecture [RZ19] (Abbildung 7.1 (3)) bietet nur einen Ort für vorverarbeitete Daten. Weder bereinigte noch integrierte Daten sind Teil des Modells. Es wird allerdings hervorgehoben, dass alle Daten Governance unterliegen. Da die Konformität mit rechtlichen Regularien im Modell nicht diskutiert wird, ist diese Anforderung jedoch nur teilweise erfüllt. Das Modell unterstützt sowohl Reporting und OLAP als auch fortgeschrittene Analysen in der Access Zone. Das Zurückschreiben von Ergebnissen ist nicht Teil des Modells. Es wird nur sehr wenig Beschreibungsdetail zur Verfügung gestellt und es gibt keinerlei Herleitungs- und Bewertungsdetail. Beide Anforderungen sind darum nicht erfüllt.

Die Data Lake Reference Architecture von Sharma [Sha18] (Abbildung 7.1 (4)) bietet die Möglichkeit zur Verwaltung von vorverarbeiteten, bereinigten, integrierten und regulierten Daten. Das Modell beinhaltet ebenfalls rechtliche Regularien in die Verwaltung der Daten indem es die Maskierung und Anonymisierung von Daten erlaubt. Zwar gibt es eine Zone für fortgeschrittene Analyse, allerdings gibt es keine Möglichkeit für Reporting und OLAP. Das

Zurückschreiben von Ergebnissen ist nur für Systeme möglich, nicht aber für menschliche Nutzer. Diese Anforderung ist daher nicht erfüllt. Der Beschreibung fehlt es an Implementierungsdetails und es gibt keine Informationen zur Herleitung oder Bewertung.

Abschließend bietet Zikopoulos Data Zones Model [ZDB+15] (Abbildung 7.1 (5)) Zonen für vorverarbeitete, bereinigte und integrierte Daten. Es gibt allerdings nur wenige Informationen zur Governance von Daten. Insbesondere rechtliche Regularien werden nicht angesprochen, weshalb diese Anforderung nicht vollständig erfüllt ist. Das Modell bietet sowohl Orte für Reporting und OLAP als auch für fortgeschrittene Analysen in der Business Reporting Zone und der Exploration Zone. Es existiert keine Möglichkeit, Ergebnisse in den Data Lake zurückzuschrieben. Die Zonenbeschreibungen dieses Modells vermischen konzeptionelle und physische Sichtweisen und es fehlt die Beschreibung der Interaktion zwischen Zonen. Daher wird das Beschreibungsdetail als unzureichend bewertet. Auch hier ist keine Herleitung oder Bewertung gegeben.

Diese Bewertung zeigt, dass alle funktionalen Anforderungen durch wenigstens ein existierendes Zonenmodell erfüllt werden. Allerdings konnte keines der existierenden Modelle alle Anforderungen erfüllen. Insbesondere ist das Beschreibungsdetail der Zonenmodelle unzureichend für eine praktische Implementierung. In vielen Fällen werden die Zonen lediglich vage beschrieben. Nur Zikopoulos Data Zones Model bietet Hinweise auf eine Implementierung, allerdings sind diese stark mit der konzeptionellen Beschreibung vermischt. Es bleibt unklar, wie die Zonen realisiert werden können, welches standardisierte Datenmodell verwendet werden sollte, usw. Zudem wurde keines der Zonenmodelle in einer systematischen Art hergeleitet und keinerlei Bewertung oder Diskussion verwandter Arbeiten ist gegeben.

7.3 Beschreibung des Zonenreferenzmodells

Das Metamodell für Zonen aus Kapitel 6 erlaubt eine Vielzahl möglicher Instanziierungen. Allerdings sind nicht alle Zonenmodelle, die nach dem Metamodell valide sind, auch für die praktische Nutzung geeignet. Unsere Untersuchung verwandter Arbeiten zeigt, dass bestimmte Zonen und Dateneigenschaften immer

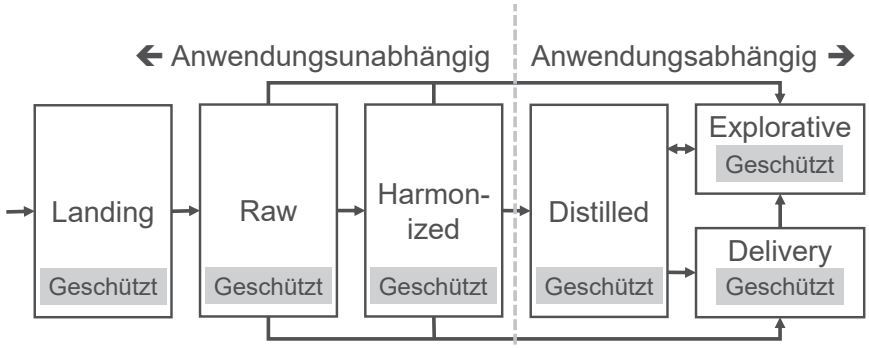


Abbildung 7.2: Das erstellte Zonenreferenzmodell umfasst sechs Zonen. Von links nach rechts werden Daten mehr und mehr für bestimmte Anwendungen aufbereitet [GGH+20b].

wieder auftauchen, wie Zonen für bereinigte oder integrierte Daten. Darum werden die häufig angetroffenen Konzepte der Literatur mit dem allgemeinen Anforderungen aus Abschnitt 7.1 kombiniert, um ein Zonenreferenzmodell zu entwickeln, dass den Anforderungen der Praxis gerecht wird. Durch die systematische Definition der Zonen und ihrer Merkmale bietet das Zonenreferenzmodell Anleitung und einen Rahmen hin zur Implementierung von zonenbasierter Datenorganisation in Data Lakes. Innerhalb dieses Rahmens kann das Zonenreferenzmodell auf die Bedürfnisse des Anwendungsszenarios angepasst werden, z. B. durch Überspringen bestimmter Zonen, die in der angestrebten Implementierung nicht benötigt werden.

Unser Zonenreferenzmodell kann sowohl für Stapel- als auch Datenstromverarbeitung für Daten beliebiger Struktur verwendet werden. Im Fall der Stapelverarbeitung werden Daten in den Zonen in verschiedenen Verarbeitungsgraden gespeichert. Bei der Datenstromverarbeitung definieren die Zonen Verarbeitungsschritte für den fließenden Datenstrom. Dabei ist der Datenstrom nach jeder Zone in einem höheren Verarbeitungsgrad abrufbar.

Die folgenden Unterabschnitte beschreiben die einzelnen Zonen im Zonenreferenzmodell in mehr Detail. Während die Landing Zone und die Raw Zone entsprechend existierender Literatur benannt sind (z. B. [Sha18]), haben die weiteren Zonen neue Namen erhalten, die ihre Funktion widerspiegeln. Das Metamodell für Zonen wird genutzt, um jede der Zonen zu beschreiben. Eine Beschreibung der Implementierung des Zonenreferenzmodell ist in Kapitel 8 enthalten. Abbildung 7.2 stellt das Zonenreferenzmodell inklusive der Bezie-

hungen zwischen den Zonen dar. Tabelle 7.4 fasst die Attribute der Zonen zusammen.

Das Zonenreferenzmodell besteht aus einem anwendungsunabhängigen und einem anwendungsabhängigen Teil. Zonen im anwendungsunabhängigen Teil behalten die gesamte originale Information der Daten bei, abgesehen von eventuellen Änderungen für rechtliche Konformität, während Zonen im anwendungsabhängigen Teil Informationsverlust akzeptieren, um bestimmte Anwendungsfälle besser zu unterstützen. Nach dem Zonenreferenzmodell kann jede Zone ausgelassen werden, abgesehen von der Raw Zone. Wie in Abbildung 7.2 und Tabelle 7.5 dargestellt, verfügt jede Zone über einen geschützten Bereich. Dieser Bereich ist verschlüsselt und besonders gesichert und enthält Daten, die besonderen Schutz benötigen (z. B. personenbezogene Daten). Daten wandern aus dem geschützten Bereich einer Zone in den geschützten Bereich der nächsten Zone. Sie dürfen den geschützten Bereich nur nach Desensibilisierung verlassen, z. B. durch Anonymisierung. Daten in diesem Bereich sind strengen Zugrifsskontrollen und Governance unterworfen. Der geschützte Bereich teilt alle anderen Attribute mit der Zone, in der er sich befindet.

7.3.1 Landing Zone

Die Landing Zone ist die erste Zone des Data Lake. Daten werden in Stapeln oder als Datenstrom von den Quellen aufgenommen. Die Landing Zone ist insbesondere dann von Vorteil, wenn sich die Anforderungen der aufgenommenen Daten und die der Raw Zone unterscheiden. Zum Beispiel kann es sein, dass Daten aufgrund ihrer Menge und Geschwindigkeit mit einer hohen Rate aufgenommen werden müssen. Falls die technische Implementierung der Raw Zone diese hohe Aufnahmerate nicht bieten kann, dient die Landing Zone als ein Mediator: Daten werden mit einer hohen Rate aufgenommen, gepuffert und als Stapel an die Raw Zone weitergeleitet.

	Landing Zone	Raw Zone	Harmonized Zone	Distilled Zone	Explorative Zone	Delivery Zone
Granularität (Roh-Aggregiert)	Roh	Roh	Roh	Aggregiert	Beliebig	Beliebig
Schema (Beliebig-Konsolidiert)	Beliebig	Beliebig	Konsolidiert	Konsolidiert, Angereichert	Beliebig	Beliebig
Syntax (Unverändert-Konsolidiert)	Grundtrans-formationen	Grundtrans-formationen	Konsolidiert	Konsolidiert	Beliebig	Beliebig
Semantik (Unverändert-Verarbeitet)	Größtenteils unverändert, außer für Regelkon-formität benötigt	Größtenteils unverändert, außer für Regelkon-formität benötigt	Größtenteils unverändert, außer für Regelkon-formität benötigt	Komplexe Verarbeitung	Beliebig	Beliebig

Tabelle 7.4: Übersicht über die Attribute der Zonen im Zonenreferenzmodell [GGH+20b]

	Landing Zone	Raw Zone	Harmonized Zone	Distilled Zone	Explorative Zone	Delivery Zone
Merkmale	Reguliert, nicht historisiert, nicht persistent, geschützter Bereich, anwendungsunabhängig	Reguliert, historisiert, persistent, geschützter Bereich, anwendungsunabhängig	Reguliert, historisiert, persistent, geschützter Bereich, anwendungsunabhängig	Reguliert, historisiert, persistent, geschützter Bereich, anwendungsabhängig	Nicht reguliert, nicht persistent, geschützter Bereich, anwendungsabhängig	Reguliert, persistent, geschützter Bereich, anwendungsabhängig
Nutzergruppen	Systeme, Prozesse	Data Scientists, Systeme, Prozesse	Data Scientists, Systeme, Prozesse	Data Scientists, Domänenexpert*innen, Systeme, Prozesse	Data Scientists	Alle menschlichen Nutzer, Systeme, Prozesse
Modellierungstechnik	Beliebig	Beliebig	Standardisiert	Standardisiert	Beliebig	Beliebig

Tabelle 7.5: Übersicht über die Attribute der Zonen im Zonenreferenzmodell (cont.) [GGH+20b]

Daten in der Landing Zone bleiben vorwiegend in ihrem Rohformat. Ihre Granularität bleibt roh, wie in den Quellsystemen. Das Schema der Daten wird nicht verändert, sie können in ihrem Quellsystemformat kopiert werden. Allerdings ändert sich unter Umständen die Syntax der Daten. Grundtransformationen sind bei der Aufnahme in die Landing Zone erlaubt, wie die Anpassung von Zeichensätzen von Strings oder die Umwandlung von Zeitstempeln in ein gemeinsames Format. Zudem können Daten maskiert oder anonymisiert werden, um rechtlichen Regularien zu folgen. Neben diesen Änderungen bleibt die Semantik der Daten erhalten.

Wie in Tabelle 7.5 gezeigt sind die Merkmale der Landing Zone: Reguliert, nicht historisiert, nicht persistent (d. h. Daten werden gelöscht, wenn sie in die Raw Zone weitergeleitet werden) und anwendungsunabhängig. Dazu verfügt die Landing Zone, wie alle anderen Zonen, über einen geschützten Bereich. Die Landing Zone ist nicht für die Nutzung durch Endnutzer*innen ausgelegt. Lediglich Systeme und Prozesse dürfen Daten in und aus der Zone schreiben. Abschließend ist keine spezifische Modellierungstechnik definiert, da Daten in einem beliebigen Format aufgenommen werden können.

7.3.2 Raw Zone

Alle Daten des Data Lake sind im größtenteils rohen Format in der Raw Zone verfügbar. Lediglich Grundtransformationen (siehe Landing Zone, Abschnitt 7.3.1) werden auf die Daten angewandt. Falls die Landing Zone ausgelassen wird, werden diese Transformationen bei der Aufnahme in die Raw Zone ausgeführt. Die Attribute der Landing Zone und der Raw Zone unterscheiden sich in zwei Punkten (siehe Tabelle 7.5): Die Merkmale und die Nutzergruppe. In der Raw Zone werden Daten historisiert und persistent gespeichert. Historisiert bedeutet, dass Änderungen in den Daten nachverfolgbar sind. Im Allgemeinen sollten Daten in der Raw Zone weder manipuliert noch gelöscht werden. Nach unserer Erfahrung allerdings ist ein solcher Ansatz in der Praxis nicht durchführbar, da die Menge der Daten rapide wächst (z. B. Sensormesswerte) und manche Daten unter rechtliche Regularien fallen, die eine Löschung fordern (z. B. DSGVO). Daher können Daten in der Raw Zone verändert, komprimiert oder sogar gelöscht werden, was in einer Abwägung zwischen Speicherverbrauch und Konformität

sowie Vollständigkeit der Daten resultiert. Verschiedene Kompressionsalgorithmen beeinflussen hierbei die Genauigkeit der Daten unterschiedlich [WSL18].

Was die Nutzergruppen der Raw Zone betrifft, so dürfen Data Scientists auf die Daten zugreifen. Diese Nutzergruppe hat ein tiefes Verständnis für Datenanalysen. Sie können Daten in z. B. die Explorative Zone kopieren, um dort Analysen auszuführen. Die Nutzung der Daten aus dem geschützten Bereich ist allerdings stark reglementiert.

7.3.3 Harmonized Zone

Eine Untermenge der Daten aus der Raw Zone wird bei Bedarf in die Harmonized Zone weitergeleitet. Es ist hierbei wichtig anzumerken, dass diese Daten nicht aus der Raw Zone gelöscht werden. Stattdessen enthält die Harmonized Zone eine Kopie oder eine Sicht der Daten in der Raw Zone. Die Harmonized Zone ist der Ort, wo Stammdaten (siehe Abschnitt 3.2.1) zur Analyse zur Verfügung stehen. Da diese Daten von kritischer Bedeutung für Unternehmen sind, ist das Management von Stammdaten von hoher Bedeutung im Data Lake. Daher sollten Stammdaten nur in bereinigtem and qualitätsgesicherten Zustand zugegriffen werden.

Die Dateneigenschaften in der Harmonized Zone unterscheiden sich grundlegend von denen der Raw Zone (siehe Tabelle 7.4). Daten aus unterschiedlichen Quellsystemen werden in ein konsolidiertes Schema integriert, unabhängig von ihrer Struktur (z. B. über link-based-Integration [GSM14]). Die Syntax der Daten wird in der Harmonized Zone ebenfalls konsolidiert. Wenn Daten aus mehreren Quellsystemen zusammengelegt werden (z. B. aus mehreren Tabellen in eine einzelne Tabelle), müssen Datentypen angepasst werden. Die Merkmale und Nutzergruppen verändern sich verglichen mit der Raw Zone nicht (siehe Tabelle 7.5).

Das Ziel der Harmonized Zone ist es, eine harmonisierte und konsolidierte Sicht auf die Daten zu bieten. Hierfür nutzt die Harmonized Zone eine standardisierte Modellierungstechnik (z. B. dimensionale Modellierung oder Data Vault [Lin12]), mithilfe derer alle Unternehmensdaten modelliert werden. Dies bedeutet allerdings nicht, dass alle Daten Teil eines einzelnen, übergreifenden Schemas sind. Stattdessen existieren in der Harmonized Zone mehrere Teilschemata, die verschiedene Datenquellen und Kontexte abdecken. Jedes

dieser Teilschemata wächst inkrementell wenn neue Daten bei Bedarf zur Harmonized Zone hinzugefügt werden. Es kann so passieren, dass mehrere dieser Teilschemata zu einem größeren Teilschema verbunden werden. Allerdings ist es nicht Ziel der Harmonized Zone, alle Daten des Unternehmens miteinander zu verbinden. Innerhalb der Teilschemata sollte ein hohes Level von Datenintegrität (z. B. Primär- und Sekundärschlüsselbedingungen) sichergestellt werden.

7.3.4 Distilled Zone

Im Gegensatz zur Raw Zone und Harmonized Zone, wo der Fokus auf die schnelle Verfügbarkeit von Daten liegt, fokussiert sich die Distilled Zone auf die Erhöhung der Effizienz von Analysen, indem die Daten entsprechend aufbereitet werden. Dadurch unterscheiden sich die Dateneigenschaften im Hinblick auf Granularität und Semantik (siehe Tabelle 7.4). Die Granularität der Daten kann verändert werden, z. B. können Daten zur Berechnung von KPIs aggregiert werden. Komplexe Verarbeitungsschritte werden auf die Daten angewandt, welche die Semantik der Daten verändern und zu weitläufig für die Landing Zone, Raw Zone und Harmonized Zone sind. Auch das Schema kann sich leicht ändern, abhängig vom unterstützten Anwendungsfall. Beispielsweise können Felder oder sogar Tabellen hinzugefügt werden, um die Daten anzureichern.

Betrachtet man die Merkmale, so ist die Distilled Zone die erste anwendungsabhängige Zone (siehe Tabelle 7.5). Daten werden für eine bestimmte Gruppe an Anwendungsfällen vorbereitet. Dies trifft sowohl für ruhende Daten als auch für Datenströme zu. Es sind unter Umständen mehrere verschiedene Versionen der selben Daten in der Distilled Zone verfügbar. Die Nutzergruppen der Harmonized Zone haben auch Zugriff auf die Distilled Zone. Zusätzlich können auch Domänenexpert*innen mit weniger Erfahrung in Datenanalyse auf die Daten zugreifen. Um die Daten abzurufen sollten einfach nutzbare Interfaces zur Verfügung gestellt werden, welche eine Abfrage der Daten mit bekannten Abfragesprachen ermöglichen (z. B. Structured Query Language (SQL)).

7.3.5 Explorative Zone

Die Explorative Zone ist der Ort, wo Data Scientists mit den Daten spielen können und sie flexibel nutzen können. Es gibt hier keine allgemeinen Dateneigenschaften, die auf alle Daten in der Explorative Zone zutreffen (siehe Tabelle 7.4). Data Scientists können Daten beliebig nutzen und explorieren, abgesehen von sensiblen Daten. Diese sind nur unter strengen Regeln nutzbar. Granularität, Schema, Syntax und Semantik können so verändert werden, wie es für die Analyse erforderlich ist.

Die Merkmale unterscheiden sich von denen der Distilled Zone (siehe Tabelle 7.5). Die Explorative Zone ist nicht reguliert, wodurch Nutzer*innen die Daten beliebig verwenden können. Abhängig vom Anwendungsfall sind Daten nicht unbedingt historisiert, weshalb über dieses Merkmal keine Aussage gemacht wird. Zudem ist die Explorative Zone nicht persistent, d. h. Daten werden nach ihrer Nutzung gelöscht. Allerdings können Ergebnisse aus den ausgeführten Analysen in die Distilled Zone weitergeleitet werden, um für zukünftige Analysen zur Verfügung zu stehen.

Es gelten wenige Regeln und Restriktionen für die Nutzung der Daten, wodurch die Explorative Zone die flexibelste Zone ist. Manchmal ist es nötig, fortgeschrittene Analysen auf sensiblen Daten auszuführen. Diese Analysen werden im geschützten Bereich der Explorative Zone ausgeführt, welcher verschlüsselt und gesichert ist.

Nur Data Scientists sind als Nutzer*innen dieser Zone zugelassen. Sie können Daten aus jeder beliebigen Zone in die Explorative Zone importieren und dort transformieren. Jede Modellierungstechnik ist erlaubt, da die Data Scientists die Daten wie benötigt verarbeiten.

7.3.6 Delivery Zone

In der Delivery Zone werden keine Teilmengen der Daten für ganz bestimmte Nutzungen und Anwendungen vorbereitet. Dies beinhaltet nicht nur analytische Anwendungen, wie Reporting und OLAP, sondern auch operative Anwendungen, wie das Bereitstellen von Daten für alltägliche Aufgaben. Diese Zone bietet daher eine Funktionalität ähnlich zu Data Marts und operativen Datenspeichern, wie sie aus Data Warehouses bekannt sind. Daten aus dieser

Zone können zu externen Datensenken weitergeleitet werden. Wie bei der Explorativen Zone hängen die Dateneigenschaften stark vom jeweiligen Anwendungsfall ab. Dabei unterscheidet sich die Delivery Zone von der Distilled Zone in soweit, dass Daten für ganz konkrete Werkzeuge und Fragestellungen aufbereitet werden im Gegensatz zu Gruppen von Anwendungsfällen. Mehrere Versionen der selben Daten können in der Delivery Zone verfügbar sein.

Wie in Tabelle 7.5 dargestellt unterscheidet sich die Delivery Zone von der Explorative Zone in ihren Merkmalen und Nutzergruppe. Daten in der Delivery Zone werden reguliert und persistent gespeichert, es sei denn, der angestrebte Anwendungsfall wird nicht weiter ausgeführt. Daten in dieser Zone können von einer Vielzahl von Nutzer*innen zugegriffen werden. Menschliche Nutzer*innen (Data Scientists, Domänenexpert*innen, Geschäftsnutzer*innen) sowie Systeme und Prozesse lesen Daten aus der Delivery Zone. Die Delivery Zone unterstützt dabei insbesondere Nutzer*innen mit geringem Wissen zu Datenanalysen. Daten müssen darum einfach auffindbar sein und in Analysewerkzeuge importiert werden können. Daten sind in beliebigen Modellierungstechniken verfügbar, abhängig vom Format, das den angestrebten Anwendungsfall bestmöglich unterstützt, z. B. dimensionale Modellierung für OLAP oder flache Tabellen für Visualisierungen.

Das erstellte Zonenreferenzmodell wird in Kapitel 8 protypisch implementiert. Hierfür notwendige Überlegungen folgen in Abschnitt 7.4. Eine umfassende Evaluation erfolgt in Abschnitt 9.2.

7.4 Implementierungsmuster für Zonenmodelle

Zur Implementierung des Zonenreferenzmodell sind neben der Definition der Zonen weitere Überlegungen nötig. Solche Überlegungen umfassen, zum Beispiel, wie die einzelnen Zonen umgesetzt werden, welche Zonen genau in welchem Fall verwendet werden, und welche Daten wie in jeder Zone vorliegen. Hierzu gehört wie Zonen repräsentiert werden (etwa durch Dateistrukturen oder Metadaten) wie Speichersysteme in Kombination mit dem Zonenmodell verwendet werden (d. h. welche Art von Speichersystemen finden in welcher Zone Anwendung) und wie unterschiedliche Verarbeitungsmodi (Stapel- und Datenstromverarbeitung) im Zonenmodell umgesetzt werden.

Existierende Literatur bietet zu diesen Fragestellungen bislang keine Hinweise oder Entscheidungsunterstützung. Zwar gibt es unterschiedliche Ansätze für die Implementierung von Data Lakes, etwa die Nutzung eines einzelnen Speichersystems [MBG+17] oder mehrerer Systeme [Dix10]. Allerdings werden diese Ansätze nicht auf ihre Umsetzung und Implikationen mit Zonenmodellen diskutiert. Auch für die Verbindung zwischen Stapel- und Datenstromverarbeitung in Zonenmodellen existieren unterschiedliche Möglichkeiten der Umsetzung, beispielsweise mit einer einzelnen, separierten Datenstromzone, wie in [ZDB+15], oder die Umsetzung beider Verarbeitungsmodi in den gleichen Zonen, wie in [RZ19]. Bislang gibt es jedoch keinerlei Übersicht und Diskussion über die verschiedenen Möglichkeiten, Zonenmodelle zu implementieren. Es bleibt daher unklar, welche Optionen für die Umsetzung von Zonen existieren und welche Vor- und Nachteile diese bieten.

Dieser Abschnitt definiert und diskutiert darum drei unterschiedliche Arten von Implementierungsmustern für Zonenmodelle:

- Zonenstrukturmuster, die beschreiben, wie die Zonenstruktur im Data Lake repräsentiert wird
- Zonenspeichermuster, die beschreiben, wie der Aspekt des Datenspeichers mit dem Zonenreferenzmodell interagiert
- Zonenflussmuster, die beschreiben, wie der Aspekt des Datenflusses mit dem Zonenreferenzmodell interagiert

Diese Muster werden in den folgenden Unterabschnitten herausgearbeitet und diskutiert. Hierbei wird das Zonenreferenzmodell zur Illustration der Muster verwendet. Es ist anzumerken, dass alle Muster allgemein auf beliebige Zonenmodelle anwendbar sind. Abschließend folgt in Abschnitt 7.4.4 eine Zusammenfassung zu den Implementierungsmustern.

7.4.1 Zonenstrukturmuster

Die Zonenstruktur stellt die interne Aufteilung des Data Lake dar. Mit ihrer Hilfe kann festgestellt werden, zu welcher Zone ein Datenobjekt gehört und über welche Daten sich eine bestimmte Zone erstreckt. Zonenstrukturmuster stellen verschiedene Varianten dar, diese Zonenstruktur abzubilden. Aus der Literatur

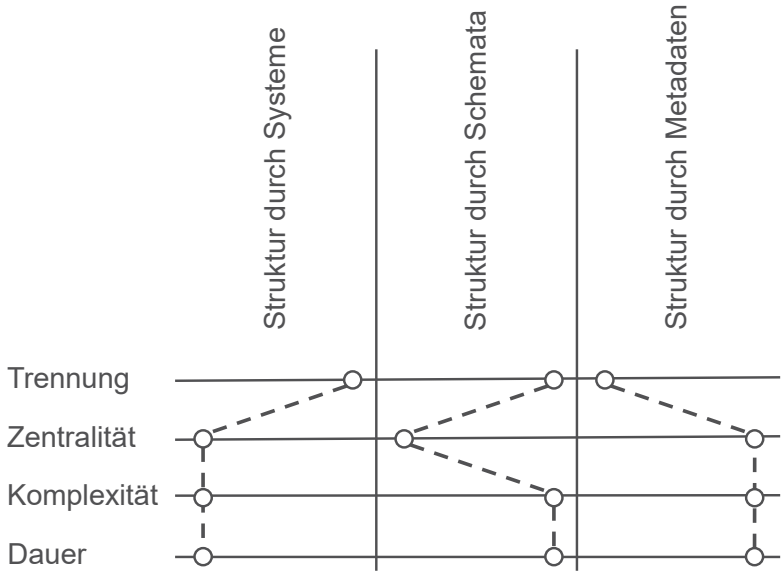


Abbildung 7.3: Gegenüberstellung der Zonenstrukturmuster. Je weiter rechts der Punkt, desto besser schneidet das Muster in der entsprechenden Metrik ab.

wurden hierfür drei Muster abgeleitet. In der Praxis sind auch Mischformen dieser Muster denkbar. Zur Evaluation dieser Muster werden im Folgenden vier Evaluationsmetriken verwendet: die Trennung der Zonen innerhalb des Systems (wie gut sind die Zonen voneinander abgegrenzt), die Zentralität der Struktur (wird die Struktur an zentraler Stelle verwaltet), die Komplexität des entstehenden Systems sowie die Dauer der Verarbeitungspipeline. Eine Übersicht über die Bewertung der einzelnen Muster ist in Abbildung 7.3 dargestellt. Die linke Seite stellt hierbei eine negative Bewertung dar, während die rechte Seite eine positive Bewertung darstellt.

Struktur durch Systeme. In diesem Muster wird jede einzelne Zone in einem eigenen System oder einer eigenen Systeminstanz umgesetzt. Dies bedeutet, dass in jeder Systeminstanz lediglich Daten einer einzelnen Zone liegen. So gibt es etwa eine Instanz des Hadoop Distributed File System (HDFS) für die Landing Zone und eine separate Instanz für die Raw Zone. Auch können weitere Systeme Anwendung finden, wie relationale Datenbankmanagementsysteme (RDBMS) oder NoSQL-Datenbanken. Abgeleitet ist dieses Muster von dem Data Zones Model von Zikopoulos et al. [ZDB+15]. Zwar wird hier nicht für jede Zone ein eigenes System genutzt, allerdings wird ein Großteil der Zonen

anhand der Technik voneinander abgegrenzt, mit der sie umgesetzt werden. So ist beispielsweise die Deep Analytic Zone dadurch definiert, dass hier Systeme zugrunde liegen, die komplexere und fortgeschrittenere Analysen ermöglichen als Hadoop [ZDB+15]. Dieses Muster bedeutet jedoch nicht, dass jede Zone nur in einem System umgesetzt wird. Vielmehr können mehrere Systeme zur Umsetzung einer einzelnen Zone verwendet werden. Dabei ist allerdings jede Systeminstanz genau einer Zone zugeordnet.

Der Vorteil dieses Musters ist, dass durch die Verwendung separater Systeme und Instanzen die Zonen klar voneinander getrennt sind. Wann immer auf Daten zugegriffen wird ist klar, zu welcher Zone diese gehören. Zu den Nachteilen dieses Musters gehört allerdings, dass es keine zentrale Stelle gibt, an der die Zonenzugehörigkeit verwaltet wird. Werden Daten über etwa einen Datenkatalog gefunden, so gibt lediglich der Speicherort Rückschluss darauf, zu welcher Zone die Daten gehören. Auch verfügt dieses Muster über eine Vielzahl unterschiedlicher Systeme und Systeminstanzen, was die Komplexität und den Verwaltungsaufwand des Data Lake erhöht. Zudem müssen Daten immer, wenn sie für eine neue Zone aufbereitet werden, von einem System in ein anderes überführt werden. Dies führt zu hohen Latenzen in der Verarbeitungspipeline.

Struktur durch Schemata. Um Zonen durch Schemata abzugrenzen, wird auf interne Strukturierungsmethoden der verwendeten Systeme zurückgegriffen. So werden beispielsweise Schemata in RDBMS oder Ordner im HDFS verwendet, um Zonen innerhalb des Systems abzubilden. Im Gegensatz zur „Struktur durch Systeme“ werden hier mehrere Zonen auf demselben System oder derselben Systeminstanz umgesetzt. Zu finden ist dieses Muster zum Beispiel im AIRPORTS DL [MBG+17], wo Zonen durch Ordner im HDFS umgesetzt sind. Auch hier kann eine einzelne Zone auf mehreren Systemen umgesetzt werden. Dabei besitzt jedes verwendete System eine interne Struktur mit dem Namen der Zone, etwa ein Ordner in HDFS sowie ein Schema in MongoDB für die Raw Zone.

Vorteile dieses Musters sind, dass erneut Zonen innerhalb der Speichersysteme klar voneinander abgegrenzt sind. Auch ist dieses Muster deutlich weniger komplex als das Muster „Struktur durch Systeme“ und es finden weniger systemübergreifende Datentransfers statt. Dies reduziert die Dauer der Verarbeitungspipeline gegenüber des Musters „Struktur durch Systeme“. Nach-

teilig an diesem Muster ist jedoch, dass erneut keine zentrale Zonenverwaltung existiert. Insbesondere bedeutet dies hier, dass Teilmengen von Zonen, die sich über mehrere Systeme erstrecken, voneinander isoliert sind. So existiert etwa die Raw Zone in HDFS komplett losgelöst von der Raw Zone in MongoDB.

Struktur durch Metadaten. Das letzte Zonenstrukturmuster fokussiert sich auf die Repräsentation von Zonen durch Metadaten. Dies bedeutet, dass für jedes Datum im Data Lake ein Metadatum existiert, welches die zugehörige Zone repräsentiert. Auch hier können sich Zonen über mehrere Systeme erstrecken, innerhalb der Systeme findet allerdings keine Strukturierung statt. Die Metadaten werden in einem zentralen Metadatenrepository verwaltet, welches außerhalb der Speichersysteme liegt. In der Literatur ist dieses Muster durch das Metadatenmodell HANDLE [EGG+20] repräsentiert.

Der Vorteil dieses Musters ist, dass Zonen an zentraler Stelle definiert werden. So können alle Daten einer Zone virtuell zusammengehalten werden, was Isolation verhindert. Komplexität sowie Dauer der Verarbeitungspipeline gleichen dem „Struktur durch Schemata“. Als Nachteil allerdings stellt sich dar, dass innerhalb der Systeme keinerlei Strukturierung existiert. Werden die Daten also nicht über Metadaten, sondern direkt zugegriffen, ist es unmöglich, die Zone zu identifizieren, zu der ein Datum gehört.

Wie bereits erwähnt sind auch Kombinationen dieser Muster denkbar. So könnten beispielsweise die Muster „Struktur durch Schemata“ und „Struktur durch Metadaten“ kombiniert werden. In diesem Fall wären die Zonen sowohl innerhalb der Systeme strukturiert, als auch in den Metadaten repräsentiert. Somit werden die Vorteile beider Muster kombiniert und die Nachteile umgangen. Allerdings bedeutet dies auch, dass die Zonenzugehörigkeit an zwei unterschiedlichen Stellen definiert ist, wodurch sich Inkonsistenzen ergeben können. Dies ist ein Nachteil, den die Muster im Einzelnen nicht besitzen.

7.4.2 Zonenspeichermuster

Die zweite Art von Implementierungsmustern, die Zonenspeichermuster, beschreiben, wie der Aspekt des Datenspeichers (siehe Abschnitt 5.2) mit einem Zonenmodell interagiert. Dies umfasst die Anzahl der Speichersysteme, die bei der Umsetzung Anwendung finden, sowie die Verteilung der Daten auf diese Speichersysteme. In der Literatur finden sich hierfür zwei Alter-

nativen, namentlich Einzelspeicherarchitekturen und Polyglottarchitekturen, d. h. Architekturen bestehend aus mehreren Systemen (siehe Abschnitt 2.1). Einzelspeicherarchitekturen kommen vor allem in Arbeiten zu praktischen Implementierungen vor, etwa in [MBG+17; MM18]. Meist wird hierbei das HDFS als einziger Datenspeicher verwendet. Polyglottarchitekturen fanden bereits im ersten Artikel zu Data Lakes Erwähnung. Dieser schlug vor, Daten mit den Werkzeugen zu verwalten, die am besten für die Anforderungen geeignet sind [Dix10]. Auch andere Arbeiten zu Data Lakes greifen auf Polyglottarchitekturen zurück, wie beispielsweise Zikolpoulos et al. [ZDB+15]. Es existiert allerdings keinerlei Diskussion der Ansätze in der Literatur. Die Details, wie konkrete Eigenschaften, Aufbau sowie Vor- und Nachteile der Ansätze bleiben unklar.

In diesem Abschnitt werden daher vier Architekturmuster vorgestellt, welche aus zwei Data-Lake-Implementierungen [MM18; MBG+17], sowie weiteren Indikatoren aus verschiedenen Arbeiten zu Data-Lake-Architekturen abgeleitet wurden. Für die Diskussion der Muster wird auf eine Menge von Evaluationsmetriken zurückgegriffen: Verwaltungsfunktionalität (wie passend können gespeicherte Daten verwaltet werden), die Komplexität des Gesamtsystems, die Wartbarkeit des Gesamtsystems, die Anzahl an Schnittstellen zwischen Systemen, die zeitliche Dauer der Verarbeitungspipeline und die Redundanz in der Datenhaltung. Die vorgestellten Zonenspeichermuster bieten eine Leitlinie für die Implementierung, sind aber nicht als Instruktionen oder Vorgaben zu verstehen.

Abbildung 7.4 stellt die Bewertung der Zonenspeichermuster dar. Hierbei wurden Wartbarkeit sowie Schnittstellenzahl vernachlässigt, da diese stark mit der Komplexität zusammenhängen. Wie bereits in Abbildung 7.3 stellt die linke Seite eine negative Bewertung dar, während die rechte Seite eine positive Bewertung darstellt.

Einzelspeicher. Das erste Zonenspeichermuster, welches in diesem Abschnitt diskutiert wird, ist das Einzelspeichermuster. Nach diesem Muster wird das gesamte Zonenmodell auf einem einzelnen Datenspeichersystem implementiert. In der Literatur wird typischerweise das HDFS für ein solches Einzelspeichersystem verwendet, etwa in [MBG+17]. Oft werden zusätzlich weitere Werkzeuge aus dem Hadoop-Ökosystem verwendet, wie Hive oder Impala. Auch dann

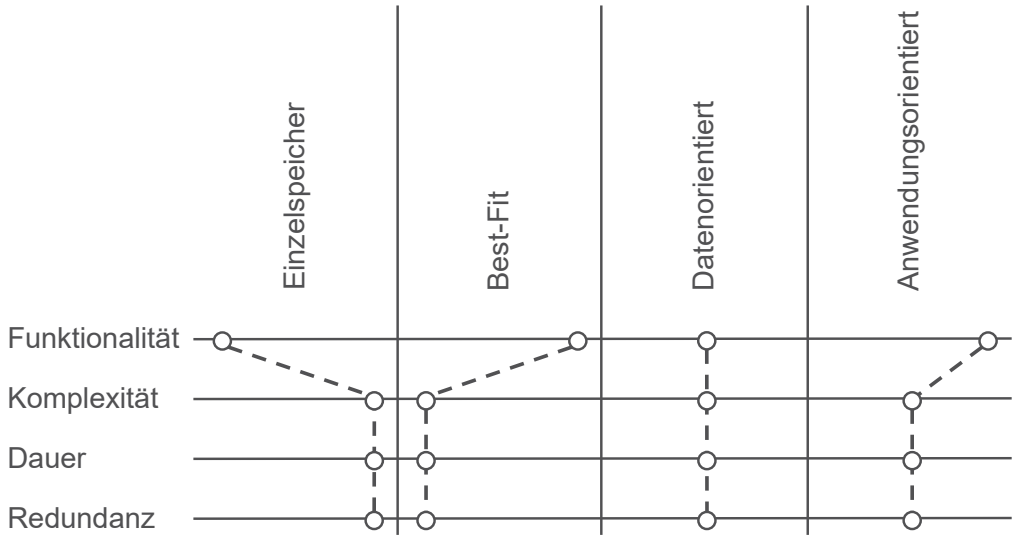


Abbildung 7.4: Gegenüberstellung der Zonenspeichermuster. Je weiter rechts der Punkt, desto besser schneidet das Muster in der entsprechenden Metrik ab.

gilt ein solches System weiterhin als Einzelspeichersystem, da alle Werkzeuge aus dem selben Ökosystem stammen und so nahtlos miteinander verbunden werden können. Obwohl es sich bei HDFS und verwandten Werkzeugen um die meist verwendeten Technologien für dieses Muster handelt, ist dieses Muster keinesfalls auf das Hadoop-Ökosystem limitiert. Andere mögliche Alternativen zur Umsetzung sind ein Einzelspeichersystem mit MySQL, MongoDB, Neo4J[1] oder einem beliebigen anderen skalierbaren Speichersystem.

Der Hauptvorteil dieses Musters ist die geringe Komplexität des entstehenden Gesamtsystems. Nur wenige Schnittstellen müssen beachtet werden, da alle Daten im gleichen Speichersystem verbleiben, auch über Zonen hinweg. Dies führt auch zu einer hohen Wartbarkeit, da nur ein System gewartet werden muss. Durch die Verwaltung der Daten in einem einzelnen System kann Redundanz verhindert werden, da Daten für weitere Zonen nicht unbedingt kopiert werden müssen, sondern auf Sichten zurückgegriffen werden kann. Es finden in diesem Muster keine Datentransfers zwischen Systemen statt, wodurch die Datenbewegung zwischen den Zonen schneller ist als in einem polyglotten System. Dies führt zu einer kurzen Dauer der Verarbeitungspipeline. Der Nachteil dieses Musters ist jedoch die Verwaltungsfunktionalität, d. h.

[1] https://neo4j.com/

wie gut die Eigenschaften und Anforderungen der Daten und Anwendungen unterstützt werden. Da alle Daten im gleichen System verwaltet werden, ist die Funktionalität bei der Verwaltung und Abfrage auf dieses System limitiert. Beispielsweise werden relationale Daten in MongoDB ohne Fremdschlüsselbedingungen gespeichert oder Videos müssen als binäre Blobs in relationalen Datenbanken gespeichert werden. So sind Abfragen auf diesen Daten unter Umständen langsamer und weniger effizient, als wenn sie in einem passenderen System gespeichert werden, und die benötigte Governance kann nicht unbedingt gewährleistet werden, z. B. aufgrund fehlender Randbedingungen.

Dieses Muster ist nicht mit dem Zonenstrukturmuster „Struktur durch Systeme“ kombinierbar, da hier auf eine einzelne Instanz eines einzelnen Systems zurückgegriffen wird.

Polyglott - Best-Fit. Den Einzelspeichersystemen gegenüber stehen die polyglotten Systeme, in denen mehrere Speichersysteme zur Umsetzung des Zonenmodells verwendet werden. Da die Literatur keine Hinweise darauf gibt, wie ein solches polyglottes System realisiert werden könnte, werden im folgenden drei Zonenspeichermuster für polyglotte Systeme mit unterschiedlichen Ausrichtungen definiert. Das erste dieser Muster ist das Best-Fit-Muster. Hier werden die Daten entsprechend ihrer Eigenschaften und ihres Verwendungszwecks auf verschiedene Systeme verteilt. So werden beispielsweise relationale Daten in einer relationalen Datenbank und stark vernetzte Daten in einer Graphdatenbank gespeichert. Das bedeutet, dass Daten zwischen Systemen verschoben werden, wenn sich ihre Eigenschaften oder ihr Verwendungszweck ändern. So können die Daten das System wechseln, wenn sie in eine andere Zone verschoben werden. Darüber hinaus werden die Daten je nach Verwendungszweck über mehrere Systeme hinweg repliziert. Beispielsweise können Kundendaten in einer relationalen Datenbank für die betriebliche Nutzung, in einem dimensionalen Datenspeicher für das Reporting und in einer Graphdatenbank zur Identifizierung von Verbindungen zwischen Kunden und den von ihnen gekauften Produkten gespeichert werden. Die Anzahl der verwendeten Datenspeichersysteme und Verarbeitungswerkzeuge ist bei diesem Muster unbegrenzt.

Ein Vorteil dieses Musters ist die optimale Verwaltungsfunktionalität, da Daten immer in dem System gespeichert werden, das ihre Eigenschaften und

geplante Nutzung bestmöglich unterstützt. Allerdings kommt dieses Muster durch die Vielzahl verwendeter Systeme auch mit sehr hoher Komplexität, geringer Wartbarkeit und einer hohen Anzahl an Schnittstellen. Hinzu kommt, dass Daten über Speichersysteme repliziert werden müssen, was zu hoher Redundanz führt. Auch benötigt der Transfer der Daten zwischen Speichersystemen zusätzliche Zeit, was die Dauer der Verarbeitungspipeline erhöht.

Dieses Muster kann mit beliebigen Zonenstrukturmustern kombiniert werden.

Polyglott - Datenorientiert. Das zweite polyglotte Muster, welches im Rahmen dieser Arbeit identifiziert wurde, ist datenorientiert. Im Gegensatz zum Best-Fit-Muster zielt dieses Muster nicht darauf ab, sowohl die Datencharakteristika als auch die geplante Nutzung zu unterstützen. Stattdessen fokussiert es sich auf die zu verwaltenden Daten. Eine anwendungsorientierte Variante wird als viertes und letztes Muster diskutiert. Beim datenorientierten Muster wird der Data Lake auf mehreren Speichersystemen aufgebaut, die alle unterschiedliche Dateneigenschaften unterstützen, wie etwa strukturiert, semistrukturiert, unstrukturiert, relational oder stark vernetzt. Je nach diesen Eigenschaften werden die Daten in dem System gespeichert, das sie am besten unterstützt, z. B. strukturierte relationale Daten in einem relationalen System und stark vernetzte Daten in einer Graphdatenbank. Wenn Daten jedoch diese Eigenschaften ändern, etwa weil semistrukturierte Daten in ein strukturiertes Format gebracht werden, werden sie von einem Speichersystem in das andere übertragen. Obwohl dieses Muster nicht anwendungsorientiert ist, können Daten dennoch in ein Format gebracht werden, das auf eine bestimmte Verwendung ausgerichtet ist, wie eine flache Tabelle für maschinelles Lernen. Das liegt daran, dass in späteren Zonen einer Zonenarchitektur die Daten vorverarbeitet werden, um für bestimmte Anwendungsfälle geeignet zu sein. Die Entscheidung, in welchem Speicher sie gespeichert werden sollen, hängt jedoch von den Eigenschaften der Daten ab, nicht von den beabsichtigten Abfragen.

Ein Vorteil dieses Musters ist eine gute Verwaltungsfunktionalität, wenn es um die Unterstützung der Dateneigenschaften geht. Die Schwächen des Musters liegen in der Komplexität, die aus einer hohen Anzahl von Schnittstellen resultiert und zu einer geringen Wartbarkeit führt. Dies liegt daran, dass in allen Zonen mehrere Systeme enthalten sind. Jede Zone ist mit Merkmalen

verknüpft, die die Daten in ihr haben. Da die Daten verarbeitet werden, um diesen Merkmalen zu entsprechen, wechseln die Daten möglicherweise mit jeder Zone das Speichersystem. Dies führt zu einer hohen Redundanz und zu einer langen Dauer der Verarbeitungspipeline.

Wie auch die anderen polyglotten Muster kann dieses Muster mit einem beliebigen Zonenstrukturmuster kombiniert werden.

Polyglott - Anwendungsorientiert. Das letzte der polyglotten Muster ist das anwendungsorientierte Muster. Es zielt darauf ab, Daten entsprechend ihrer beabsichtigten Verwendung und nicht entsprechend ihrer Eigenschaften zu verwalten. Solange die beabsichtigte Verwendung von Daten noch unklar ist, werden sie in einem Einzelspeichersystem gespeichert, ähnlich wie bei dem bereits beschriebenen Einzelspeichermuster. Im Zonenreferenzmodell besteht dieser anwendungsfallunabhängige Teil aus der Landing Zone, der Raw Zone und der Harmonized Zone (siehe Abbildung 7.2). Die anderen Zonen, nämlich die Distilled Zone und die Delivery Zone, sind anwendungsfallabhängig. Im anwendungsorientierten Muster werden die Daten in diesen Zonen in Speichersystemen gespeichert, die ihren Verwendungszweck optimal unterstützen. So werden etwa Sensordaten in einer Zeitreihendatenbank gespeichert, um zeitorientierte Abfragen zu ermöglichen. Dies bedeutet auch, dass mehrere Versionen der gleichen Daten in verschiedenen Speichersystemen vorhanden sind, da ein Datenelement typischerweise mehrere Verwendungszwecke hat.

Dieses Muster hat den Vorteil einer guten Verwaltungsfunktionalität, wenn es um die Abfrage und Verwendung der Daten geht. Da der anwendungsfallunabhängige Teil mit einem einzelnen Speichersystem realisiert wird, hat dieser Teil des Zonenmodells eine geringe Komplexität, hohe Wartbarkeit, wenige Schnittstellen und eine kurze Dauer der Verarbeitungspipeline. Dies ist besonders vorteilhaft, da der anwendungsfallunabhängige Teil des Data Lake typischerweise den Großteil der Daten enthält, da Daten nur bedarfsorientiert in den anwendungsfallabhängigen Teil übertragen werden. Da in diesem anwendungsfallunabhängigen Teil kein Transfer zwischen Systemen stattfindet, ist auch die resultierende Redundanz gering.

Die Nachteile dieses Musters liegen im anwendungsfallabhängigen Teil. Hier ist die Komplexität hoch, da mehr Schnittstellen benötigt werden, um die Daten beim Wechsel von Zone zu Zone in die richtigen Systeme zu übertragen. Das

senkt die Wartbarkeit und erhöht die Dauer der Verarbeitungspipeline. Außerdem erhöht die Speicherung verschiedener Zustände von Daten in mehreren Speichersystemen die benötigte Redundanz in diesem Teil des Zonenmodells.

Wie auch die anderen polyglotten Muster kann dieses Muster mit einem beliebigen Zonenstrukturmuster kombiniert werden.

7.4.3 Zonenflussmuster

Die letzte Art von Implementierungsmuster für Zonenmodelle sind Zonenflussmuster. Diese Muster beschreiben, wie die verschiedenen Modi von Datenbewegung (ruhende Daten und Datenströme) in einem Zonenmodell umgesetzt werden. Damit sind diese Muster stark mit dem Aspekt des Datenflusses (siehe Abschnitt 5.2) verbunden. In der Literatur existieren zwei Alternativen für die Umsetzung des Datenflusses in Zonenmodellen: eine einzelne Zone für die Datenstromverarbeitung und ein Zonenmodell für die Datenstromverarbeitung. Wie auch bei den vorangegangenen Mustern gibt es jedoch weder eine detaillierte Beschreibung der Muster, noch eine Gegenüberstellung der Alternativen. Darum diskutieren die folgenden Abschnitte beide Muster in mehr Einzelheiten. Hierfür werden verschiedene Evaluationsmetriken verwendet, nämlich: Die Verfügbarkeit der Zwischenschritte aus der Datenstromverarbeitung, die Komplexität des Gesamtsystems, die Wartbarkeit des Gesamtsystems, die Anzahl der Schnittstellen und die zeitliche Dauer der Pipeline.

Die Bewertung dieser Muster ist in Abbildung 7.5 dargestellt. Auch hier wurden Wartbarkeit sowie Schnittstellenzahl mit der Komplexität zusammengefasst. Wie bereits zuvor stellt die linke Seite eine negative Bewertung dar, während die rechte Seite eine positive Bewertung darstellt.

Datenstromzone. Das erste der beiden Zonenflussmuster ist eine einzelne Datenstromzone. Dieses Muster ist etwa in der Arbeit von Zikopoulos et al. [ZDB+15] zu finden. Bei diesem Muster umfasst die verwendete Zonenarchitektur eine einzelne separate Zone zur Datenstromverarbeitung, die neben der abgestuften Zonenarchitektur für die Stapelverarbeitung existiert. Der eingehende Datenstrom wird in dieser Zone verarbeitet und erst zugänglich gemacht, wenn er vollständig verarbeitet ist. In Abbildung 7.6 ist eine solche einzelne Datenstromzone neben der Zonenarchitektur für die Stapelverarbeitung dargestellt.

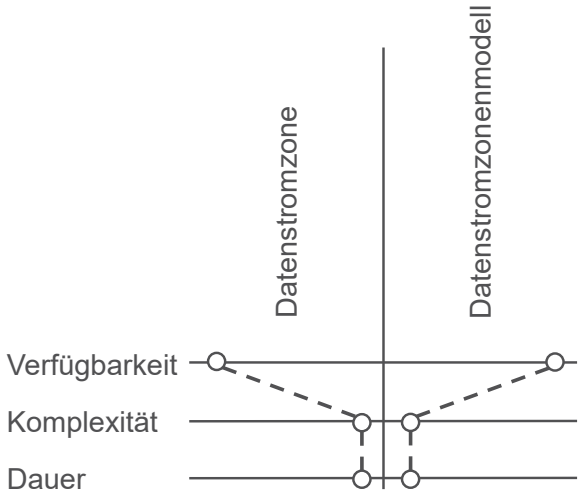


Abbildung 7.5: Gegenüberstellung der Zonenflussmuster. Je weiter rechts der Punkt, desto besser schneidet das Muster in der entsprechenden Metrik ab.

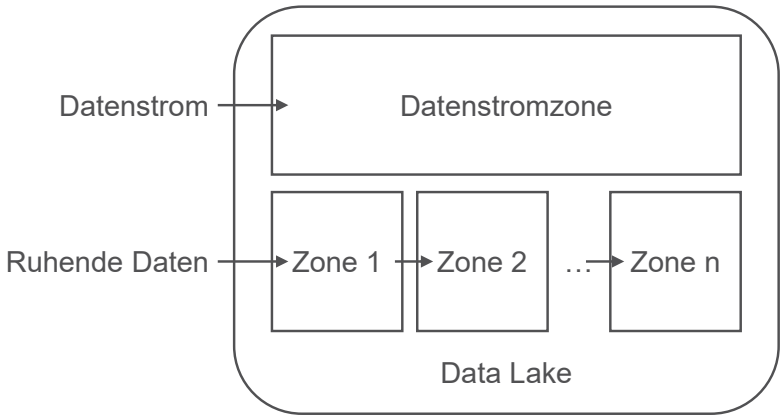


Abbildung 7.6: Das Zonenflussmuster für eine einzelne Datenstromzone

Die Vorteile dieses Musters sind die geringe Komplexität und die schnelle Pipeline. Die Daten werden nur in ein einziges Stream-Processing-System übertragen und können vollständig verarbeitet werden, sobald sie ankommen. Die geringe Komplexität beruht auf den wenigen Schnittstellen, die in diesem Muster benötigt werden, was zu einer hohen Wartbarkeit führt. Der Nachteil des Musters ist die fehlende Zugänglichkeit für Zwischenverarbeitungsschritte. Da der Datenstrom in einem Durchgang verarbeitet wird, sind nur der rohe Datenstrom oder der fertige Ergebnisstrom verfügbar.

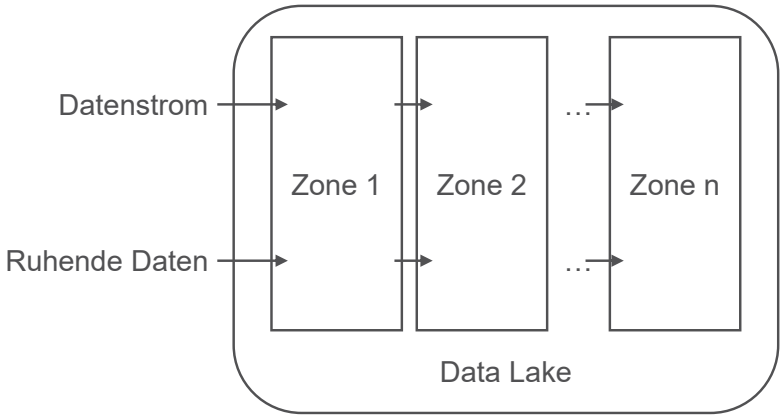


Abbildung 7.7: Das Zonenflussmuster für das Datenstromzonenmodell

Datenstromzonenmodell. Die Alternative zur einzelnen Datenstromzone ist die Verwendung eines Zonenmodells für die Datenstromverarbeitung. Dieses Muster ist in Ravat und Zhaos Arbeit zu Data Lakes zu finden [RZ19]. Eine Darstellung des Musters ist in Abbildung 7.7 abgebildet. Bei diesem Muster wird nicht zwischen Datenströmen und ruhenden Daten unterschieden. Beide werden mit denselben Zonen als Basis verarbeitet. Der einzige Unterschied besteht darin, dass ruhende Daten in der jeweiligen Zone gespeichert werden, während auf Datenströme über APIs im jeweiligen Verarbeitungsgrad zugegriffen werden kann.

Dieses Muster hat den Vorteil, dass alle dazwischenliegenden Verarbeitungsgrade des Datenstroms zugänglich sind. Über die APIs ist es möglich, auf den Rohdatenstrom, einen bereinigten Strom oder einen fertig verarbeiteten Datenstrom zuzugreifen.

Die Nachteile liegen in der Datenübertragung zwischen den verschiedenen Zonen. Jede der Zonen ist eine eigene Verarbeitungseinheit, d. h. ein Skript oder ein Microservice. Dies führt zu einer großen Anzahl von Schnittstellen, geringer Wartbarkeit und hoher Komplexität. Darüber hinaus erhöht die Übertragung von Daten von einer Zone zur anderen die Gesamtdauer der Verarbeitungspipeline.

7.4.4 Zusammenfassung Implementierungsmuster

Insgesamt bieten die in diesem Abschnitt erarbeiteten Implementierungsmuster nicht nur eine Übersicht über verschiedene Implementierungsmöglichkeiten für

Zonenmodelle, sondern auch eine Orientierung für die Auswahl eines geeigneten Musters für ein konkretes Anwendungsszenario. Hierzu sind insbesondere die Gegenüberstellungen in Abbildung 7.3, Abbildung 7.4 und Abbildung 7.5 hilfreich. Mit ihrer Hilfe kann auf Grundlage einer Anforderungsanalyse im Anwendungsszenario für Zonenstruktur, Zonenspeicher und Zonenfluss das Implementierungsmuster gewählt werden, das die beste Unterstützung für die gewünschten Eigenschaften bietet. So wird die Implementierung eines Zonenmodells angeleitet und unterstützt.

7.5 Methodik für die Umsetzung einer zonenbasierten Datenorganisation

Die existierenden Konzepte für zonenbasierte Data Lakes sind rein konzeptionell (z. B. [Sha18]), nicht allgemeingültig (z. B. [MBG+17]) oder so vage beschrieben, dass eine Reproduktion nicht möglich ist (z. B. [ZDB+15]). Bei der Implementierung eines Data Lake stellt sich daher die Frage, wie ein Zonenmodell für die gegebenen Bedürfnisse instantiiert werden kann. Um diese Frage zu beantworten, stellt dieser Abschnitt eine Methodik vor, die die Definition einer implementierbaren Instanz eines beliebigen Zonenmodells unterstützt. Hierzu nutzt die Methodik die verfügbaren Daten und ihre angestrebte Nutzung als Eingabe. Von dort aus leitet sie Entwickler durch die Schritte, die notwendig sind, um das gegebene Zonenmodell zu implementieren. Die Methodik ist dabei unabhängig vom genutzten Zonenmodell und kann mit jedem beliebigen Modell verwendet werden (z. B. [Gor16; Sha18; RZ19]). Sie umfasst neun Schritte, die im Folgenden beschrieben sind. Abbildung 7.8 stellt die Methodik visuell dar.

Schritt A: Voraussetzungen identifizieren. Der erste Schritt der Methodik erfordert eine Identifikation der Voraussetzungen, mit denen das Zonenmodell implementiert wird. Solche Voraussetzungen sind die verfügbaren Daten, ihre Anforderungen, die angestrebte Nutzung, und andere Randbedingungen, die durch das Anwendungsszenario gegeben sind. Die angestrebten Analysen sind deshalb wichtig, da Daten für ihre spätere Nutzung vorverarbeitet werden (siehe z. B. die Delivery Zone in Kapitel 7). Während Zonenmodelle selbst allgemeingültig sind, fokussiert sich ihre Implementierung auf konkrete Anwen-

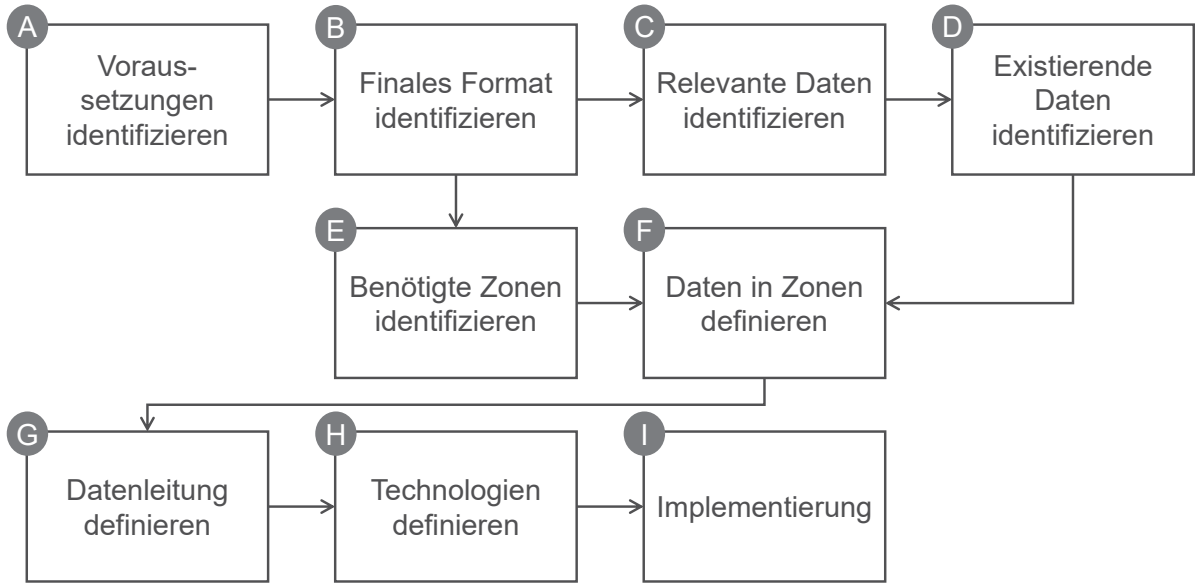


Abbildung 7.8: Überblick über die Methodik zur Definition einer zonenbasierten Data-Lake-Architektur

dungen. Ein Data Lake verfügt typischerweise nur über ein Zonenmodell, dieses wird allerdings für verschiedene Anwendungen unterschiedlich instantiiert, da Anwendungen verschiedene Zonen, Daten und Formate benötigen.

Schritt B: Finales Format identifizieren. Im nächsten Schritt gilt es, das finale Format der Daten zu identifizieren, d. h. das Format, in dem sie (Geschäfts-)Nutzer*innen zur Verfügung stehen. Ein solches Format kann eine Tabelle, ein ganzes Schema, ein Dokument, ein Graph oder etwas anderes sein. Das finale Format ist abhängig von den angestrebten Analysen und Werkzeugen, die für diese Analysen verwendet werden. So gibt es beispielsweise Werkzeuge, die auf Reporting ausgelegt sind, die wiederum Daten in einem bestimmten Format erwarten, z. B. in einem dimensionalen Schema. Nach diesem Schritt gibt es zwei Pfade in der Methodik, die unabhängig voneinander ausgeführt werden können: Die Identifikation von relevanten und existierenden Daten (C, D) und die Identifikation der benötigten Zonen (E).

Schritt C: Relevante Daten identifizieren. In diesem Schritt werden die Daten identifiziert, die für die Realisierung der betrachteten Anwendung notwendig sind, z. B. die richtige Kunden- und Bestelltabelle für Verkaufsberichte. Dies beinhaltet die Datenquellen, die benötigt werden, die benötigten Daten aus diesen Quellen, deren Struktur und wie diese Daten verbunden werden müssen, um das finale Format aus Schritt B zu erreichen.

Schritt D: Existierende Daten identifizieren. Basierend auf den Ergebnissen aus Schritt C ist es notwendig zu identifizieren, welche Daten bereits im Zonenmodell verfügbar sind. Abhängig davon, ob andere Anwendungsfälle bereits implementiert wurden, kann es sein, dass benötigte Daten bereits bereinigt, integriert oder anderweitig vorbereitet vorliegen. Beispielsweise können die Kundendaten für die Reporting-Anwendungsfälle aus Abschnitt 8.1 bereits aus einem anderen Anwendungsfall passend vorverarbeitet im Data Lake vorhanden sein. In einem solchen Fall können die Daten wiederverwendet werden und müssen nicht erneut transformiert werden. In diesem Fall ist die Implementierung für diese Daten abgeschlossen. Sind die Daten gar nicht oder nur in einem unpassenden Format vorhanden (z. B. als flache Tabelle, aber es wird ein dimensionales Schema benötigt), folgt Schritt F, sobald Schritt E abgeschlossen ist.

Schritt E: Benötigte Zonen identifizieren. Unabhängig von den Schritten C und D ist das Ziel von Schritt E zu identifizieren, welche Zonen für das gegebene Anwendungsszenario notwendig sind. Abhängig vom genutzten Zonenmodell sind unterschiedliche Zonen für die Implementierung verfügbar, z. B. die Trusted Zone [Sha18]. Nicht alle Zonen eines gegebenen Zondenmodells sind unbedingt für die praktische Nutzung in einem bestimmten Anwendungsszenario notwendig. Beispielsweise benötigen Anwendungsszenarien ohne explorative Analysen keine Explorative Zone.

Schritt F: Daten in Zonen definieren. Nachdem in den vorangegangenen Schritten die benötigten Daten und Zonen identifiziert wurden, wird in Schritt F definiert, wie die Daten in den jeweiligen Zonen vorliegen. Hierzu müssen die Eigenschaften der Zonen zur Betrachtung hinzugezogen werden. Zum Beispiel muss festgelegt werden, wie Daten bereinigt werden, oder wie sie für Zonen vorverarbeitet werden, die anwendungsabhängig sind (z. B. die Delivery Zone aus Kapitel 7).

Schritt G: Datenpipeline definieren. Nachdem alle Zwischenschritte im Zonenmodell in den vorangegangenen Schritten ausdefiniert wurden, muss die Datenpipeline spezifiziert werden. Diese transformiert und bewegt Daten zwischen den Zonen. Typischerweise wird diese Datenpipeline mithilfe von ETL-Prozessen (Extract-Transform-Load) umgesetzt. Das Ziel ist es, die Daten wie in Schritt F definiert von einer Zone zur anderen zu bewegen und transformieren.

Schritt H: Technologien definieren. Um eine zonenbasierte Datenorganisation und ihre Prozesse zu implementieren, müssen geeignete Speicher- und Verarbeitungstechnologien für jede Zone gewählt werden. Hierbei gilt es, auf die in der Data Lake Architecture Framework (DLAF)-Methodik getroffenen Entscheidungen Rücksicht zu nehmen. Es kann hierbei sein, dass unterschiedliche Zonen mithilfe verschiedener Technologien realisiert werden und Daten sich zwischen Speichersystemen bewegen, während sie durch das Zonenmodell wandern.

Schritt I: Implementierung. Der letzte Schritt stellt die Implementierung der zonenbasierten Datenorganisation dar.

Die Durchführung all dieser Schritte ist notwendig, um sicherzustellen, dass alle Aspekte des DLAF ausdefiniert und ihre Abhängigkeiten untereinander berücksichtigt werden. Da sich die Reihenfolge der Schritte an den allgemeingültigen Abhängigkeiten zwischen den Aspekten orientiert, ist auch die Methodik selbst allgemein anwendbar und nicht an bestimmte Anwendungsszenarien gebunden. Durch die Verwendung der Methodik wird die Definition einer vollständigen Data-Lake-Architektur systematisch angeleitet und unterstützt.

7.6 Zusammenfassung

In diesem Kapitel wurde das Zonenreferenzmodell vorgestellt, welches auf Grundlage existierender Zonenmodelle sowie der Anforderungen aus der Praxis erstellt wurde. Das Metamodell für Zonen wurde verwendet, um die einzelnen Zonen des Zonenreferenzmodells strukturiert und detailliert zu beschreiben. So kann das Zonenreferenzmodell für die strukturierte interne Organisation eines Data Lake verwendet werden. Auch sind Implementierungsmuster sowie eine Methodik zur Umsetzung beliebiger Zonenmodelle vorgestellt. Durch den Vergleich der verschiedenen Implementierungsmuster ist es möglich, eine interne Datenorganisation für einen Data Lake praktisch umzusetzen und dabei die für das Anwendungsfall beste Implementierung zu wählen.

Kapitel 8

Prototypische Umsetzung

Existierende Arbeiten zu Data Lakes bieten keine allgemeingültigen Leitlinien zur möglichen Implementierung. Zwar gibt es Arbeiten, die die Implementierung einzelner Aspekte abdecken (z. B. Datenfluss [MM18] oder Metadatenmanagement [HGQ16; EGG+20]), allerdings existieren keine allgemeingültigen Umsetzungsleitlinien für eine vollständige Data-Lake-Architektur. Implementierungen, die eine vollständige Data-Lake-Architektur beinhalten, sind auf spezifische Anwendungsfälle zugeschnitten (z. B. [MBG+17]), und dadurch nicht allgemein anwendbar. Auch zum Aspekt der Datenorganisation im Speziellen existieren keine Leitlinien zur Implementierung. Vorhandene Implementierungen zu diesem Aspekt sind entweder anwendungsspezifisch oder verfügen nicht über genügend Details zum Nachbau (siehe [ZDB+15]).

Dieses Kapitel befasst sich daher mit der Umsetzung eines Data-Lake-Prototypen auf Basis des Data Lake Architecture Framework (DLAF) als Vorarbeit für die Evaluation der in dieser Arbeit entwickelten Konzepte. Hierzu stellt Abschnitt 8.1 ein konkretes Anwendungsszenario des Fallbeispiels (siehe Abschnitt 3.2) vor, welches die Grundlage für die Definition der Data-Lake-Architektur bietet. Abschnitt 8.2 leitet eine vollständige Data-Lake-Architektur für dieses Anwendungsszenario unter Anwendung der DLAF-Methodik (siehe Abschnitt 5.4) her. Abschnitt 8.3 befasst sich anschließend mit der Umsetzung

einer zonenbasierten Datenorganisation innerhalb dieser Data-Lake-Architektur auf Grundlage des Zonenreferenzmodells (siehe Kapitel 7). Abschnitt 8.4 stellt die prototypische Realisierung der erstellten Data-Lake-Architektur vor. Es wird im Folgenden zwischen konzeptioneller Umsetzung und Implementierung unterschieden. Eine konzeptionelle Umsetzung beinhaltet Überlegungen und Spezifizierung auf einem konzeptionellen Level, d. h. es finden theoretische Vorüberlegungen aber keine Implementierung statt. Die Implementierung dagegen beschreibt eine Realisierung der erarbeiteten Konzepte in Code und Speichersystemen. Zuletzt schließt Abschnitt 8.5 das Kapitel ab.

Dieses Kapitel stellt eine Zusammenstellung mehrerer überarbeiteter vorausgegangener Publikationen der Autorin dar [GGH+20b; GGH+21].

8.1 Anwendungsszenario

Das Anwendungsszenario, das der Definition und Umsetzung der Data-Lake-Architektur zugrunde gelegt wird, entstammt dem in Abschnitt 3.2 vorgestellten Fallbeispiel eines großen, global agierenden Herstellers. Wie bereits erwähnt ist das Geschäft dieses Herstellers hochgradig divers und reicht von der Zulieferung für andere Industrien über die Herstellung von Produkten für Endkunden bis hin zur Bereitstellung digitaler Services für diese Produkte. Über das gesamte Unternehmen hinweg werden Datenanalyseprojekte in das Tagesgeschäft integriert, welche sowohl Reporting und Online Analytical Processing (OLAP), als auch fortgeschrittene Analysen, wie Machine Learning, umfassen (siehe Abschnitt 3.2.2). Die hierfür verwendeten heterogenen Daten sind in Abschnitt 3.2.1 näher beschrieben. Diese werden in einem zentralen, unternehmensweiten Data Lake verwaltet, auf den zahlreiche Nutzer*innen mit unterschiedlichen Erfahrungsgraden zur Datenanalyse zugreifen (siehe Abschnitt 3.2.2). Konkret fokussiert sich das zugrundeliegende Szenario auf die Entwicklung und Nutzung von Endkundenprodukten, wie Haushaltsgeräte, Elektrowerkzeuge oder Smart-Home-Infrastruktur. Jedes der verkauften Produkte verfügt über eine Vielzahl von Sensoren, die unterschiedliche Daten über ihre Nutzung erfassen, wie Geschwindigkeit, Ort oder Batteriestand. Zudem sind strukturierte Daten zu jedem Produkt vorhanden, wie die Stückliste, Daten zum Kunden, der das Produkt erworben hat, oder Daten zur Bestellung selbst.

Während die in dieser Arbeit erstellten Konzepte allgemeingültig und damit unabhängig von einer konkreten Anwendung sind, benötigt ihre Implementierung festgelegte Randbedingungen. Nur so können passende Konzepte, Strategien und Werkzeuge ausgewählt werden. Die Wahl eines konkreten Anwendungsszenarios ist daher für die Implementierung notwendig.

Innerhalb des Anwendungsszenarios fokussiert sich die prototypische Umsetzung auf fünf unterschiedliche Data-Lake-Anwendungsfälle, welche in Tabelle 8.1 zusammengefasst sind. Diese Anwendungsfälle sind repräsentativ für das Tagesgeschäft des Herstellers und umfassen sowohl Reporting und OLAP als auch fortgeschrittene Analysen. Zudem befindet sich ein operativer Anwendungsfall darunter, in welchem ein Serviceblatt zu einem Produkt zur Verfügung gestellt wird. Die folgenden Abschnitte enthalten zusätzliche Details zu den einzelnen Anwendungsfällen.

Machine Learning. In diesem Anwendungsfall werden die Daten mithilfe von Machine Learning analysiert. Diese Analysen sind typischerweise diagnostisch oder prädiktiv [Hag16]. Für unseren Zweck werden drei konkrete Fragestellungen innerhalb dieses Anwendungsfalls genutzt. Die Grundlage für die erste Fragestellung (ML1) sind Felddaten, die durch Sensoren in einem Produkt während dessen Nutzung gesammelt werden. Diese Sensoren erfassen etwa Geschwindigkeit, Drehmoment oder Batterieladung. Die so gesammelten Daten werden verwendet, um Vorhersagen zur Batterieladung zu treffen, oder eine intelligente Motorsteuerung zu ermöglichen, welche die benötigte Antriebskraft selbstständig ermittelt. Die zweite Fragestellung (ML2) fokussiert sich auf die Identifikation von Mustern in der Bestellhistorie, z. B. wie sich Bestellungen zwischen Jahreszeiten oder Nutzergruppen unterscheiden. In der dritten Fragestellung (ML3) werden Daten aus sozialen Netzwerken verwendet, um aktuelle Markttrends zu identifizieren sowie die Kundenzufriedenheit zu evaluieren. Hierzu werden aktuelle Trends, sowie Nachrichten zu Produkten aus etwa Twitter analysiert. Die Nutzer*innen, die für diesen Anwendungsfall verantwortlich sind, sind Data Scientists, mit hohem Wissensstand im Bezug auf Datenaufbereitung und -analyse. Sie können daher direkt auf Rohdaten arbeiten und diese so aufbereiten, wie sie für die gewünschten Analysen benötigt werden.

Anwendungsfall	Beschreibung	Nutzer
Machine Learning	ML1: Vorhersagen auf Felddaten	Data Scientists
	ML2: Identifikation von Mustern in der Bestellhistorie	
	ML3: Marktanalyse auf Daten aus sozialen Netzwerken	
Reporting	R1: Verkaufszahlen für bestimmte Komponenten und Versionen	Geschäftsnutzer*innen
	R2: Verteilung von Bestellungen über Nutzergruppen	
Visuelle Analysen	VA: Visualisierung von Daten zur Fehlerursachenerkennung	Domänenexpert*innen
Streaming	S: Identifikation von Fehlfunktionen in Echtzeit	Domänenexpert*innen
Operativ	O: Angebot eines Serviceblatts für Kunden	Kunden

Tabelle 8.1: Die Anwendungsfälle für die prototypische Umsetzung einer vollständigen Data-Lake-Architektur

Reporting. Für diesen Anwendungsfall werden verfügbare Daten deskriptiv analysiert. Aus ihnen werden Berichte (eng. Reports) erstellt mithilfe von traditionellen Analysetechniken, wie OLAP. Auch hier liegt der Fokus auf zwei Fragestellungen, nämlich (R1) wie viele Komponenten eines bestimmten Typs und einer bestimmten Version wurden in einem festgelegten Zeitraum verkauft und (R2) wie sind Bestellungen über Nutzergruppen verteilt, z. B. nach Alter oder Demographie. Geschäftsnutzer*innen, d. h. Nutzer*innen mit wenig Wissen zu Datenaufbereitung und -Analyse, sind die Hauptnutzer*innengruppe dieses Anwendungsfalls. Um diese Nutzer*innen und ihre verwendeten

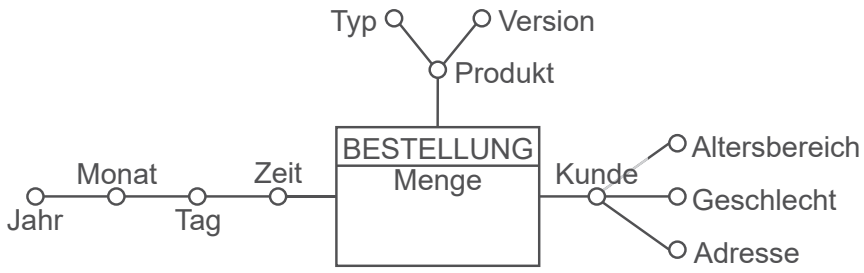


Abbildung 8.1: Das dimensionale Schema für den Reporting-Anwendungsfall

Werkzeuge bestmöglich zu unterstützen, sollten Daten in einem dimensionalen Schema vorliegen. Dieses ist in Abbildung 8.1 dargestellt.

Visual Analytics. In unserem Fall stellen Visual-Analytics-Anwendungsfälle [TC05] eine Mischung aus explorativen und Reporting-Anwendungsfällen dar mit einem Fokus auf deskriptive Analysen. Hierfür steht eine Teilmenge von Daten zur Verfügung, welche zwar vorverarbeitet sind, allerdings noch eine flexiblere Verwendung ermöglichen als im Reporting-Anwendungsfall gegeben. Diese Daten werden mithilfe passender Werkzeuge visualisiert und ausgewertet. Die Hauptnutzer*innengruppe in diesem Anwendungsfall stellen Domänenexpert*innen dar, mit tiefem Wissen zur Domäne aber weniger Wissen zu Datenaufbereitung und -Analyse. Für die Implementierung steht eine konkrete Fragestellung dieses Anwendungsfalls im Vordergrund (VA): Domänenexpert*innen stellen fest, dass viele der verbauten Motoren nach einiger Zeit ähnliche Defekte aufweisen. Aufgrund von Erfahrungswerten stellen sie eine Vermutung für die Fehlerursache auf, welche sie dann mithilfe der Visualisierung von Produkt- und Sensordaten überprüfen. Für diesen Anwendungsfall ist es wichtig, dass Daten miteinander verknüpft und allgemein aufbereitet werden. Das Format, in dem die Daten vorliegen sollen, ist abhängig von den Visualisierungswerkzeugen, die im Einsatz sind.

Streaming. Während der Produktnutzung werden Sensordaten als kontinuierlicher Datenstrom gesammelt. Diese werden verwendet, um Echtzeit-Dashboards zu erstellen oder Complex-Event-Processing [WDR06] zu ermöglichen. In dem eingehenden Datenstrom sind mehrere Sensormesswerte zu einer JavaScript Object Notation (JSON)-Nachricht verbunden. Eine typische Fragestellung (S) in diesem Anwendungsfall ist es, Fehlfunktionen des Produkts

in Echtzeit zu identifizieren. Daher müssen die großen JSON-Nachrichten in einzelne Datenströme aufgespalten werden, um einfachere Analysen zu ermöglichen. Zusätzliche Daten sind auch relevant, wie beispielsweise Produktdaten oder Wetterdaten, welche dann mit dem Datenstrom verbunden werden müssen. Die Analysen dieses Anwendungsfalls können sowohl deskriptiv als auch prädiktiv sein [FBT+12].

Operativ. Wie bereits erwähnt sollen Kunden Zugriff auf ein Serviceblatt für ihr gekauftes Produkt erhalten (O). Dieses Serviceblatt soll nicht nur Informationen zum Produkt enthalten, sondern auch aggregierte Informationen zur Produktnutzung, wie beispielsweise Durchschnittsgeschwindigkeit, die Entwicklung der Batterieladung oder die zurückgelegte Strecke. Dieser Anwendungsfall ist rein deskriptiv. Eine Kombination aus aggregierten Sensordaten und anderen Daten ist als flache Tabelle, d. h. eine einzelne, denormalisierte Tabelle, verfügbar, die durch eine App zugegriffen werden kann.

8.2 Konzeptionelle Umsetzung des DLAF

In diesem Abschnitt wird mithilfe der Methodik des DLAF (siehe Abschnitt 5.4) eine vollständige Data-Lake-Architektur für das gegebene Anwendungsszenario erstellt. Diese ist auch für andere Anwendungsszenarien innerhalb des Data Lake anwendbar, da die für das vorgestellte Anwendungsszenario geltenden Randbedingungen typisch im Geschäft des Herstellers sind. Tabelle 8.2 gibt einen Überblick über die Ergebnisse der einzelnen Schritte der DLAF-Methodik. Die folgenden Absätze beschreiben die Entscheidungsprozesse, die zu dem jeweiligen Ergebnis geführt haben.

Schritt 1: Szenario identifizieren. Um eine geeignete Grundlage für die Definition einer prototypischen Data-Lake-Architektur zu legen, wird an dieser Stelle auf die in **Abschnitt 5.4 Schritt 1** definierten Fragen zur Identifikation der Randbedingungen des Szenarios zurückgegriffen. Basierend auf den Antworten zu diesen Fragen werden die verbleibenden Schritte zur Konfiguration einer Data-Lake-Architektur durchgeführt, um diese passend für die Bedürfnisse des Herstellers im Fallbeispiel zu gestalten.

1) Welche Art von Daten werden im Data Lake verwaltet? Was sind ihre Eigenschaften? - Die Daten, die im betrachteten Anwendungsszenario verwen-

Schritt	Ergebnis
1: Szenario identifizieren	Strukturierte, semi-strukturierte und unstrukturierte Daten
	Stapel- und Datenstromverarbeitung
	Traditionelle und fortgeschrittene Analysen
2: Datenfluss	BRAID
3: Datenorganization	Zonenreferenzmodell
4: Datenspeicher	Mehrspeichersystem
5: Infrastruktur	Hadoop (HDFS, MapReduce), Kafka, MySQL, Apache Spark, ..., teilweise Cloud-basiert
6: Datenmodellierung	Data Vault, Link-Based-Integration
7: Metadata als Unterstützer	Metadatentypen basierend auf HANDLE
8: Datenprozesse	Organisationsspezifische Prozesse
9: Metadata als Funktion	Data Catalog

Tabelle 8.2: Übersicht über die Ergebnisse bei der Definition einer prototypischen Data-Lake-Architektur [GGH+21].

det werden, spiegeln die allgemeine Diversität im Geschäft des Herstellers wider. Daten von unterschiedlichsten Quellsystemen, z. B. Enterprise Resource Planning (ERP)-Systeme, Manufacturing-Execution-Systeme (MESs) oder dem Internet of Things (IoT), werden für die Analysen benötigt. Hinzu kommen Daten aus externen Quellen, wie sozialen Netzwerken. Diese Daten sind strukturiert (z. B. Produktdaten), semi-strukturiert (z. B. Sensordaten) oder unstrukturiert (z. B. Computer Aided Design (CAD)-Dateien zu hergestellten Produkten). Zudem unterscheiden sie sich stark in anderen Eigenschaften, von sensiblen Stammdaten und personenbezogenen Kundendaten bis hin zu voluminösen IoT-Daten unbekannter Qualität (vgl. Abschnitt 3.2.1).

2) Welche zeitlichen Anforderungen sind mit den Daten und ihrer Nutzung assoziiert? - Unterschiedliche zeitliche Anforderungen sind mit den Daten und deren Nutzung verbunden. So ist es für die Reporting-Anwendungsfälle ausrei-

chend, wenn die Daten stündlich aktualisiert werden, während im Streaming-Anwendungsfall Daten in Echtzeitnähe verarbeitet werden müssen. Diese als Strom eintreffenden Daten müssen zudem für die spätere Nutzung gespeichert werden. Für diesen Anwendungsfall sind somit sowohl Stapel- als auch Datenstromverarbeitung notwendig.

3) Wie werden Daten verwendet? - Die Anwendungsfälle, die das oben beschriebene Anwendungsszenario umfasst, reichen von traditionellem Reporting und OLAP hin zu fortgeschrittenen Analysen, wie Data Mining und Machine Learning.

Schritt 2: Datenfluss entwerfen. Wie in Schritt 1 beschrieben nutzt das zugrundeliegende Anwendungsszenario sowohl Stapel- als auch Datenstromverarbeitung für die Analyse der Daten. Für das Konzept zur Umsetzung des Datenflusses für Datenströme in der prototypischen Data-Lake-Architektur wurde BRAID [GSSM18] verwendet. In diesem Konzept werden Datenströme sowohl an einen persistenten Speicher als auch in eine Datenstromverarbeitungsumgebung weitergeleitet. Dieses Verhalten ist ähnlich zu dem der Lambda-Architektur, wie in Abschnitt 2.2.2 beschrieben. BRAID erlaubt es allerdings, Ergebnisse aus der Stapelverarbeitung in der Datenstromverarbeitung zu verwenden. So kann zum Beispiel ein Machine-Learning-Modell auf ruhenden Daten gelernt werden und zur Klassifikation von Datenströmen verwendet werden. Da die Eigenschaften von BRAID mit den Anforderungen des Herstellers übereinstimmen, ist dieses Konzept für die prototypische Data-Lake-Architektur gut geeignet. Daten, die als Stapel in den Data Lake geladen werden, werden direkt persistiert und per Stapelverarbeitung verarbeitet. Dieses Datenflusskonzept ist in Abbildung 8.2 dargestellt.

Schritt 3: Datenorganisation entwerfen. Da Daten in diesem Anwendungsszenario oft zugegriffen und verwendet werden, wird eine Segmentierung des Data Lake benötigt. Zudem werden Daten unterschiedlicher Struktur oft miteinander kombiniert (siehe z. B. VA) und die Verfügbarkeit von Rohdaten ist von immenser Wichtigkeit (siehe ML1, ML2, ML3). Auf Grundlage des in Abbildung 5.5 dargestellten Entscheidungsprozesses wurde eine Zonenarchitektur als Datenorganisationskonzept gewählt. Da im Datenflusskonzept sowohl Stapel- als auch Datenstromverarbeitung Anwendung finden, muss die Datenorganisation dies ebenfalls unterstützen. Hierzu wurde das Zonenreferenzmodell

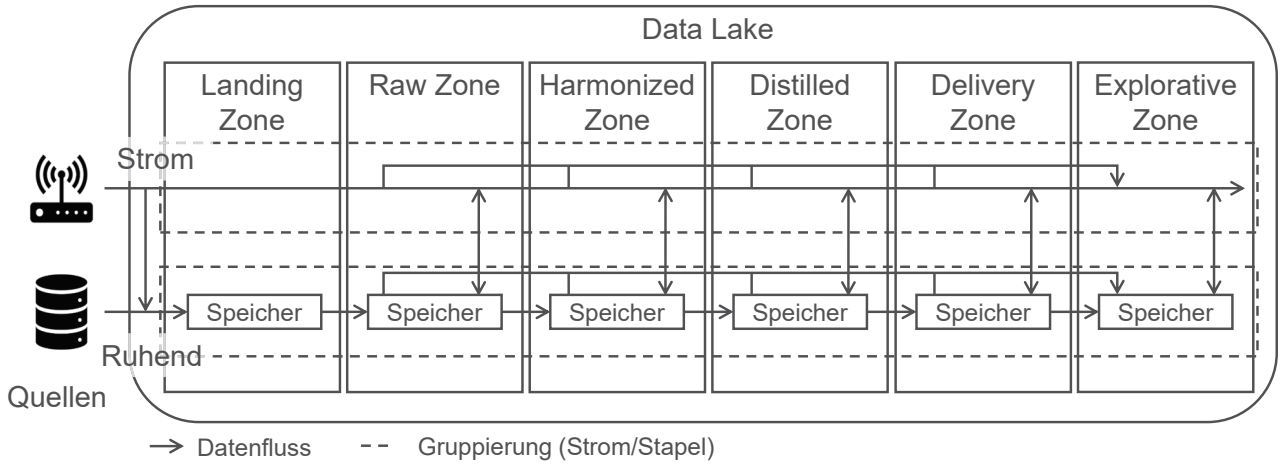


Abbildung 8.2: Ein Ausschnitt der prototypischen Data-Lake-Architektur, inklusive Datenfluss (Schritt 2) und Datenorganisation (Schritt 3) [GGH+21].

aus Kapitel 7 verwendet, da es die Anforderungen für einen praktischen Einsatz erfüllt. Zudem beinhaltet es Informationen für die Realisierung beider Verarbeitungsvarianten. Abbildung 8.2 zeigt das Zonenreferenzmodell und seine Interaktion mit ruhenden Daten und Datenströmen.

Schritt 4: Datenspeicher entwerfen. Da die Daten des Anwendungsszenarios stark divers sind, umfasst das Konzept für Datenspeicher mehrere verschiedene Speichersysteme (Mehrspeichersystem). So können Daten dort verwaltet werden, wo ihre Eigenschaften und Nutzung bestmöglich unterstützt werden. Beispielsweise können Sensordaten in einer Zeitreihendatenbank gespeichert werden, welche effektive zeitorientierte Abfragen ermöglicht, während unstrukturierte Daten in einem verteilten Dateisystem abgelegt werden. Das Datenspeicherkonzept für den Anwendungsfall umfasst relationale Datenbankmanagementsysteme (RDBMS), spaltenorientierte Datenbanken, Zeitreihendatenbanken und ein verteiltes Dateisystem. Zudem sind Umgebungen für die Stapel- und Datenstromverarbeitung enthalten.

Schritt 5: Infrastruktur definieren. Wie in Schritt 4 erwähnt nutzt der prototypische Data Lake verschiedene Speichersysteme und Werkzeuge. Für die Speicherung wird MySQL[1] als relationale Datenbank, Cassandra[2] als spal-

[1] https://www.mysql.com/
[2] https://cassandra.apache.org/

Zone	Eigenschaften mit Einfluss auf Datenmodellierung	Modellierungstechnik
Landing Zone	Temporär	Rohformat
Raw Zone	Große Datenmengen	Rohformat
Harmonized Zone	Standardisierte Modellierungstechnik	Data Vault (Raw Vault), Link-based-Integration
Distilled Zone	Standardisierte Modellierungstechnik, anwendungsabhängig	Data Vault (Business Vault), Link-based-Integration
Explorative Zone	Modellierung durch Data Scientists	Modellierung nach Bedürfnissen
Delivery Zone	Für bestimmte Werkzeuge aufbereitet	Modellierung nach Bedürfnissen

Tabelle 8.3: Übersicht über Entscheidungen zur Datenmodellierung [GGH+21]

tenorientierte Datenbank, InfluxDB[3] für die Speicherung von Zeitreihendatenbanken und das Hadoop Distributed File System (HDFS)[4] als verteiltes Dateisystem verwendet. Zur Verarbeitung werden Apache Kafka[5] und Apache Spark[6] verwendet. Implementiert werden die Verarbeitungsschritte in Python[7] und Jupyter Notebooks[8].

Schritt 6: Datenmodellierung entwerfen. Die Nutzung eines Zonenmodells für die Datenorganisation bedeutet, dass die Modellierungstechnik für jede Zone individuell festgelegt werden muss. So können die geforderten Dateneigenschaften und Merkmale der Zonen unterstützt werden. Die gewählten Modellierungen für die verschiedenen Zonen sind in Tabelle 8.3 dargestellt. Während Daten in der *Landing Zone* und *Raw Zone* vorrangig in ihrem Roh-

[3] https://www.influxdata.com/
[4] https://hadoop.apache.org/docs/r1.2.1/index.html#HDFS
[5] https://kafka.apache.org/
[6] https://spark.apache.org/
[7] https://www.python.org/
[8] https://jupyter.org/

format beibehalten werden, wird Data Vault für die strukturierten Daten in der *Harmonized Zone* und *Distilled Zone* verwendet. In einer vorausgegangenen umfassenden Studie der Autorin wurde festgestellt, dass sich Data Vault aufgrund der Flexibilität, Skalierbarkeit und der anwendungsunabhängigen Modellierung sehr gut für die Anwendung in Data Lakes eignet [GGH+19b]. Zusätzlich wird Link-Based-Integration [GSM14] verwendet, um strukturierte und semi-strukturierte Daten mit unstrukturierten Daten zu verbinden. Obwohl sowohl die *Harmonized Zone* als auch die *Distilled Zone* Data Vault verwenden, ergeben sich dennoch Unterschiede in der Modellierung. In der *Harmonized Zone* wird Raw Vault [LO15] verwendet, da hier die originale Granularität und Syntax der Daten erhalten bleibt. Lediglich so genannte harte Regeln werden angewandt, welche die Bedeutung der Daten nicht verändern, z. B. Änderungen des Schemas, Konsolidierung von Zeichensätzen, etc. Für die *Distilled Zone* wird stattdessen Business Vault [LO15] verwendet. Hier werden weiche Regeln angewandt, wie beispielsweise Aggregationen, zusätzliche Tabellen für effizientere Abfragen oder andere Geschäftslogik. Da durch die Anwendung der weichen Regeln die Granularität und Semantik der Daten geändert werden kann, eignet sich der Business Vault nur für die *Distilled Zone*, nicht aber für die *Harmonized Zone*. Die Daten in den letzten beiden Zonen, die *Explorative Zone* und die *Delivery Zone*, werden nach den Bedürfnissen der Nutzer, der Anwendung und der verwendeten Werkzeuge modelliert.

Schritt 7: Metadaten als Unterstützer entwerfen. Um die gespeicherten Daten zu verwalten und ihre Nutzung zu ermöglichen, wird ein passendes Metadatenmanagement benötigt. Da es sich bei Metadaten ebenfalls um Daten handelt, müssen die Schritte 1-6 für diese erneut ausgeführt werden. Metadaten können im betrachteten Anwendungsszenario strukturiert oder semi-strukturiert vorliegen und werden auf die selbe Weise aufgenommen wie die Daten, zu denen sie gehören (z. B. in Stapeln für ruhende Daten). Das Datenflusskonzept ist daher das gleiche für Daten und Metadaten. Die Datenorganisation für Metadaten ist unsegmentiert, da Metadaten sich über Zonen hinaus erstrecken, beispielsweise um mehrere Versionen der selben Daten zu verknüpfen. Da Metadaten stark verknüpft sind, z. B. Lineage-Metadaten, die die Quelldaten und alle abgeleiteten Versionen verbinden, wurde beim Datenspeicher eine Graphdatenbank gewählt. Für die Infrastruktur wird Neo4J genutzt,

eine der populärsten Graphdatenbanken. Die Datenmodellierung wird nach dem Metadatenmodell HANDLE ausgeführt, welches Lineage-Metadaten, Zonenzugehörigkeit, Zugriffsinformationen und mehr abbilden kann [EGG+20]. So können auch Metadaten zu Datensicherheit & Privatheit und Datenqualität verwaltet werden.

Schritt 8: Datenprozesse entwerfen. Datenprozesse müssen in ihren zwei Unteraspekten ausdefiniert werden: Prozesse für Datenlebenszyklusmanagement und Prozesse zur Datenpipeline.

1) Datenlebenszyklusmanagement. Prozesse zum Datenlebenszyklusmanagement für dieses Anwendungsszenario werden nach [DAM17] definiert. Diese Prozesse verwalten Daten in allen Schritten des Datenlebenszyklus, welcher von Erstellung über Speicherung, Nutzung und Anreicherung bis hin zur Löschung reicht. In all diesen Schritten werden Metadaten erfasst und mit den Daten gespeichert, z. B. Lineage-Metadaten zur Erstellung eines Datensatzes oder Metadaten zu Datenzugriffen. So können Zugriffskontrollen oder Änderungsnachverfolgung realisiert werden. Aus Platzgründen wird nicht näher auf den Aspekt der Prozesse für Datenlebenszyklusmanagement eingegangen.

2) Datenpipeline. Prozesse zur Datenpipeline sind eng mit dem Zonenreferenzmodell verknüpft, welches zur Datenorganisation verwendet wird. Ruhende Daten werden in der Landing Zone gepuffert, bevor Extract-Transform-Load (ETL)-Prozesse sie in die Raw Zone weiterleiten. Von hier aus überführen weitere ETL-Prozesse die Daten in die anderen Zonen. Diese Prozesse führen Transformationen auf den Daten aus, um sie an die in der Zone geforderten Dateneigenschaften anzupassen. Zum Beispiel werden Daten nach den Regeln von Data Vault transformiert, wenn sie aus der Raw Zone in die Harmonized Zone übertragen werden. Datenpipelineprozesse sind auch für die Realisierung von Datesicherheits- & Privatheits- und Datenqualitätskonzepten relevant. Beispielsweise müssen personenbezogene Daten anonymisiert werden, wenn sie in die Raw Zone gelangen, oder die Datenqualität wird bei der Speicherung in der Harmonized Zone sichergestellt.

Schritt 9: Metadaten als Funktion entwerfen. Der letzte Schritt ist zur Spezifikation von Konzepten für Metadaten als Funktion vorgesehen. In diesem Anwendungsszenario werden Data Catalogs [CSN+14] genutzt, die Funktionen

bereitstellen, die über die Fähigkeiten von Daten als Unterstützer hinausgehen, z. B. die Ermöglichung von Nutzerkollaboration.

Hiermit ist die konzeptionelle Umsetzung einer prototypischen Data-Lake-Architektur auf Grundlage des DLAF abgeschlossen.

8.3 Konzeptionelle Umsetzung des Zonenreferenzmodells

Im zweiten Teil der konzeptionellen Umsetzung eines Data-Lake-Prototypen liegt der Fokus auf dem Zonenreferenzmodell. Dessen Nutzung wurde in Schritt 3 der DLAF-Methodik ausgewählt. Einige der für die Umsetzung notwendigen Überlegungen wurden bereits in Abschnitt 8.2 angeschnitten, hinsichtlich der Implementierung fehlt es jedoch noch an Details. Für die Umsetzung werden folgende Implementierungsmuster verwendet (siehe Abschnitt 7.4): Struktur durch Schemata, Polyglott - Anwendungsorientiert, sowie Datenstromzonenmodell. Eine genauere Erklärung, warum welches der Muster Anwendung findet, ist an der jeweiligen Stelle der Beschreibung des Prototyps enthalten. Mithilfe der in Abschnitt 7.5 vorgestellten Methodik kann die zonenbasierte Datenorganisation für das in Abschnitt 8.1 vorgestellte Anwendungsszenario definiert und implementiert werden. Als unterliegendes Zonenmodell wird das Zonenreferenzmodell verwendet (siehe Kapitel 7), wie in Schritt 3 der DLAF-Methodik spezifiziert.

Die folgenden Abschnitte detaillieren die Ergebnisse der einzelnen Schritte der Methodik für das implementierte Anwendungsszenario. Dabei werden die fünf Anwendungsfälle aus Tabelle 8.1 zur Veranschaulichung verwendet. Die resultierende Umsetzung ist jedoch nicht auf diese Anwendungsfälle und ihre assoziierten Fragestellungen limitiert, sondern kann weitere Anwendungsfälle auf den im Data Lake vorhandenen Daten unterstützen.

Schritt A: Voraussetzungen identifizieren. Wie bereits in Abschnitt 8.1 beschrieben, umfasst das betrachtete Anwendungsszenario mehrere analytische Anwendungsfälle, von Reporting und OLAP bis hin zu fortgeschrittenen Analysen mit Machine Learning. Zudem enthält es einen operationalen Anwendungsfall, der die Zusammenstellung eines Serviceblattes beinhaltet. Die konkreten Fragestellungen sowie Nutzergruppen sind bereits in Abschnitt 8.1

detailliert. Die folgenden Abschnitte gehen darum lediglich auf die benötigten Daten näher ein.

Die benötigten Daten für die Fragestellungen ML1, ML2 und ML3 des Machine-Learning-Anwendungsfalls unterscheiden sich. Die Basis von ML1 (Vorhersage auf Felddaten) sind Felddaten, die bei der Nutzung des Produkts durch Sensoren gesammelt werden. Sie erfassen z. B. Geschwindigkeit, Ort und Batterieladung. Diese Daten werden mit Daten zum Produkt selbst verknüpft, wie einem Produktnamen und der Stückliste. In der zweiten Fragestellung ML2 (Identifikation von Mustern in der Bestellhistorie) werden Bestelldaten und Kundendaten benötigt. Die Produktdaten aus ML1 werden auch in Fragestellung ML3 benötigt, hier werden sie allerdings mit Daten aus sozialen Netzwerken angereichert. Im Reporting-Anwendungsfall werden vor allem Bestelldaten benötigt. Für die Beantwortung von Fragestellung R1 (Verkaufszahlen für bestimmte Komponenten und Versionen) werden diese mit Produktdaten verknüpft, um daraus die Versionen abzulesen. Fragestellung R2 (Verteilung von Bestellungen über Nutzergruppe) benötigt zusätzlich zu den Bestelldaten die zugehörigen Kundendaten. Der Anwendungsfall der Visual Analytics VA (Visualisierung von Daten zur Fehlerursachenerkennung) benötigt die Felddaten, sowie die Produktdaten, um Analysen durchzuführen. Die gleichen Daten werden auch vom Streaming-Anwendungsfall S (Identifikation von Fehlfunktionen in Echtzeit) benötigt, allerdings müssen sie hier als Datenstrom mit möglichst geringer Wartezeit verarbeitet werden. Zusätzlich werden hier Wetterdaten in die Analyse miteinbezogen, da z. B. Luftfeuchtigkeit Einfluss auf das Produktverhalten nimmt. Der operationale Anwendungsfall O schließlich benötigt eine Zusammenstellung verschiedenster Daten, namentlich Kunden-, Produkt-, Bestell- und Sensordaten. Diese werden miteinander verbunden und als Serviceblatt zur Verfügung gestellt.

Schritt B: Finales Format identifizieren. Im Fall des Machine-Learning-Anwendungsfalls nehmen Data Scientists die benötigten Transformationen selbst vor. Daher muss das Zonenmodell in diesem Fall kein vordefiniertes Format bereitstellen. Ganz anders ist das im Fall des Reporting-Anwendungsfalls. Hier wird für R1 ein dimensionales Schema benötigt, welches durch die verschiedenen Reporting-Werkzeuge importiert werden kann. Dieses Schema ist bereits in Abbildung 8.1 dargestellt. Im Fall von R2 dagegen wird eine flache

Tabelle benötigt. Die Faktentabelle beinhaltet hierbei die Menge der bestellten Produkte pro Bestellung, während die Dimensionstabellen Informationen zu Zeit, Produkt und Kunde bereitstellt. Im Visual-Analytics-Anwendungsfall müssen die Daten verknüpft vorliegen. Ein konkretes Format ist nicht gefordert, da die Domänenexpert*innen selbstständig kleinere Transformationen vornehmen können, abhängig vom Format, das durch das verwendete Werkzeug gefordert wird. Das finale Format im Streaming-Anwendungsfall ist ein einzelner Datenstrom, welcher Produktdaten, Messwerte zu einem einzelnen Sensor und Wetterdaten aus einer externen Quelle beinhaltet. Im operativen Anwendungsfall schließlich ist das finale Format eine flache Tabelle, die Produktdaten, Kundendaten, Bestelldaten und aggregierte Sensordaten enthält.

Schritt C: Relevante Daten identifizieren. Für die betrachteten Anwendungsfälle sind fünf Datensätze relevant: Kundendaten (ML2, R2, O), Produktdaten (alle Anwendungsfälle), Bestelldaten (ML2, R1, R2, O), Sensordaten (ML1, VA, S, O) und externe Daten, wie Wetterdaten oder Daten aus sozialen Netzwerken (ML3, S). Als soziales Netzwerk wird in diesem Fall auf Twitter zurückgegriffen. Dabei wird beispielhaft ein Datensatz von Kaggle[9] für den Prototypen verwendet. Zusätzlich wird die Stückliste als Teil der Produktdaten benötigt, welche als separater Datensatz gespeichert wird. Daten zu Kunden, Produkten, Bestellungen und Stückliste sind in einem strukturierten Format verfügbar. Kundendaten und Bestelldaten sind in den Quellsystemen mit Data Vault modelliert, während Produktdaten in dritter Normalform modelliert sind. Die Stückliste ist als Comma Separated Values (CSV)-Datei verfügbar. Die Sensor- und Wetterdaten sind semi-strukturiert und liegen als JSON-Nachrichten vor.

Schritt D: Existierende Daten identifizieren. In dem betrachteten Fallbeispiel ist der unternehmensweite Data Lake noch in einer frühen Entwicklungsphase. Darum sind keine der relevanten Daten bereits im Data Lake vorhanden. Wie in Schritt C allerdings dargestellt, ergeben sich zwischen den betrachteten Anwendungsfällen Synergien, die ausgenutzt werden können. Zum Beispiel können Kundendaten, die für den Reporting-Anwendungsfall aufbereitet wurden, für den operativen Anwendungsfall wiederverwendet werden. Das bedeutet,

[9] https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020

Zone	Benötigt für
Landing Zone	Alle Anwendungsfälle
Raw Zone	Alle Anwendungsfälle
Harmonized Zone	R1, R2, VA, S, O (, ML1, ML2, ML3)
Distilled Zone	R1, R2, VA, S, O (, ML1, ML2, ML3)
Explorative Zone	ML1, ML2, ML3
Delivery Zone	R1, R2, VA, S, O

Tabelle 8.4: Die Ergebnisse der Identifikation der benötigten Zonen

dass Daten zwar bislang nicht im Data Lake existieren, die Datenaufbereitung aber zwischen den Anwendungsfällen wiederverwendet werden können.

Schritt E: Benötigte Zonen identifizieren. Das Zonenreferenzmodell, welches für die Implementierung verwendet wird, beinhaltet sechs Zonen mit unterschiedlichem Fokus. Je nach betrachtetem Anwendungsfall werden andere Zonen für die Umsetzung benötigt. Die Ergebnisse dieses Schritts sind in Tabelle 8.4 zusammengefasst.

Die Landing Zone ist die Eintrittszone in den Data Lake und dient als Puffer. Diese Zone wird für Anwendungsfälle benötigt, in denen Daten mit einer Geschwindigkeit eintreffen, die zu hoch für die direkte Aufnahme ist. Daher ist die Landing Zone für die Anwendungsfälle ML1, Visual Analytics, Streaming und operativ verfügbar, da hier Sensordaten in hoher Geschwindigkeit aufgenommen werden müssen. Darüber hinaus werden personenbezogene Daten in der Landing Zone anonymisiert, um rechtlichen Regularien zu folgen. Darum ist die Landing Zone auch für ML2, ML3 und die Reporting-Anwendungsfälle notwendig.

Die Raw Zone enhält Daten in ihrem (größtenteils) rohen Format. Diese Zone stellt die Basis des Zonenreferenzmodells dar und kann nicht übersprungen werden. Sie ist darum in allen Anwendungsfällen notwendig.

In der Harmonized Zone werden Daten in ein standardisiertes Format gebracht und miteinander verbunden. Während diese Zone in allen Anwendungsfällen Vorteile bietet, da Daten in Verbindung gesetzt werden, ist sie für den

Machine-Learning-Anwendungsfall (ML1, ML2, ML3) nicht erforderlich. Dies ist der Fall, da Data Scientists auch direkt auf Rohdaten arbeiten können. Da in manchen Fällen die Nutzung der Harmonized Zone auch für diese Anwendungsfälle vorteilhaft sein kann, sind sie in Tabelle 8.4 in Klammern eingetragen. Für alle anderen Anwendungsfälle (R1, R2, VA, S, O) ist diese Zone allerdings notwendig, um die ausgeführten Analysen zu unterstützen.

Daten in der Distilled Zone werden für eine bestimmte Klasse von Anwendungsfällen vorbereitet. Wie auch die Harmonized Zone wird diese Zone nicht für die Machine-Learning-Anwendungsfälle benötigt, ist aber von großer Wichtigkeit für alle anderen Anwendungsfälle (R1, R2, VA, S, O). Diese Notwendigkeit entsteht dadurch, dass die Nutzer*innen aller anderen Anwendungsfälle nur geringe bis keine Erfahrung mit Datenaufbereitung und -Analyse besitzen und die Distilled Zone hier entsprechende Unterstützung bietet. Wie bereits bei der Harmonized Zone erwähnt, können auch hier die Machine-Learning-Anwendungsfälle potentiell von der Nutzung der Zone profitieren. Da dies allerdings nicht unbedingt erforderlich ist, sind ML1, ML2 und ML3 in Klammern eingetragen.

Die Explorative Zone ist der Ort, wo Data Scientists ihre benötigten Daten importieren und transformieren. Sie wird lediglich für explorative Analysen benötigt. Daher findet die Explorative Zone nur im Machine-Learning-Anwendungsfall Verwendung.

Die Delivery Zone schließlich bietet Daten in einem anwendungsspezifischen Format an, welches auf die Nutzung bestimmter Werkzeuge ausgelegt ist (z. B. Werkzeuge für OLAP). Diese Zone bietet auch einen Self-Service-Zugriffspunkt für alle Nutzer, die keine Data Scientists sind. Daher nutzen alle Anwendungsfälle, die nicht auf Data Scientists ausgerichtet sind, diese Zone (R1, R2, VA, S, O).

Schritt F: Daten in Zonen definieren. Für jede Zone, die in einem der Anwendungsfälle enthalten ist, muss definiert werden, wie Daten in dieser Zone modelliert sind. Hierzu wurden bereits in Abschnitt 8.2 Schritt 6 einige Hinweise gegeben, welche hier nur kurz wieder aufgegriffen und erweitert werden sollen.

In der Landing Zone und der Raw Zone bleiben Daten in ihrem Rohformat, wie in den Quellsystemen abgespeichert. Lediglich Kundendaten und Daten

aus sozialen Medien werden anonymisiert, d. h. Namen, Accountnamen und konkrete Adressen werden aus den Daten entfernt. Accountnamen werden dabei durch ihren Hash, Adressen durch Städtenamen ersetzt. Alle anderen Daten bleiben unverändert.

In der Harmonized Zone wird, wie bereits erwähnt, Data Vault als Raw Vault verwendet. Dabei werden nicht alle Daten im Data Lake in ein einzelnes Schema integriert, sondern mehrere Teil-Schemata erstellt. Während Kunden- und Bestelldaten bereits in Data Vault aus ihren Quellen verfügbar sind, müssen Produkt- und Sensordaten erst noch transformiert werden. Das entstehende Schema ist in Abbildung 8.3 dargestellt. Dieses besteht aus den drei Data-Vault-Komponenten: Hubs, Links, und Satellites. Hubs, in weiß dargestellt, repräsentieren ein Geschäftsobjekt, wie einen Kunden. Sie beinhalten einen Geschäftsschlüssel sowie relevante Metadaten. Andere Daten zum Geschäftsobjekt, wie Geburtsdatum oder Geschlecht, werde in Satellites (hellgrau) gespeichert, die mit den Hubs verbunden sind. Schließlich verbinden Links (dunkelgrau) verschiedene Hubs miteinander als Beziehungen, wie einen Kunden, der eine Bestellung getätigt hat. Für die Daten der Machine-Learning-Anwendungsfälle ist die Harmonized Zone nicht unbedingt notwendig, da Data Scientists auch direkt auf den rohen Daten arbeiten können. Dennoch kann die Nutzung dieser Zone einen Vorteil für die weitere Umsetzung auch solcher Anwendungsfälle bieten. Im Prototypen wird daher Fragestellung ML3 auch mithilfe der Harmonized Zone umgesetzt. Hierbei werden die erfassten Tweets in Inhalt und Verfasser zum einen, sowie in Kommentare, Retweets, Likes und andere zusätzliche Daten zum anderen aufgespalten, um ein normalisiertes Zielschema zu erreichen. Auch werden Unix-Zeitstempel in ein für Menschen lesbares Format überführt.

In der Distilled Zone bleibt das allgemeine Schema der Daten gleich, allerdings wird Geschäftslogik auf die Daten angewandt (Business Vault). So werden zusätzliche Konstrukte eingefügt, wie eine Point-In-Time-Tabelle, die nur die aktuell gültigen Werte des Kunden-Satellites beinhaltet. Das Geburtsdatum des Kunden wird durch einen Altersbereich ersetzt, welcher für das Clustering verwendet wird. Die Sensor-Satellites werden durch Satellites für aggregierte Sensordaten (10 Sekunden Intervalle) und für Key-Performance-Indikatoren (KPIs) (z. B. minimale, maximale und durchschnittliche Geschwindigkeit) er-

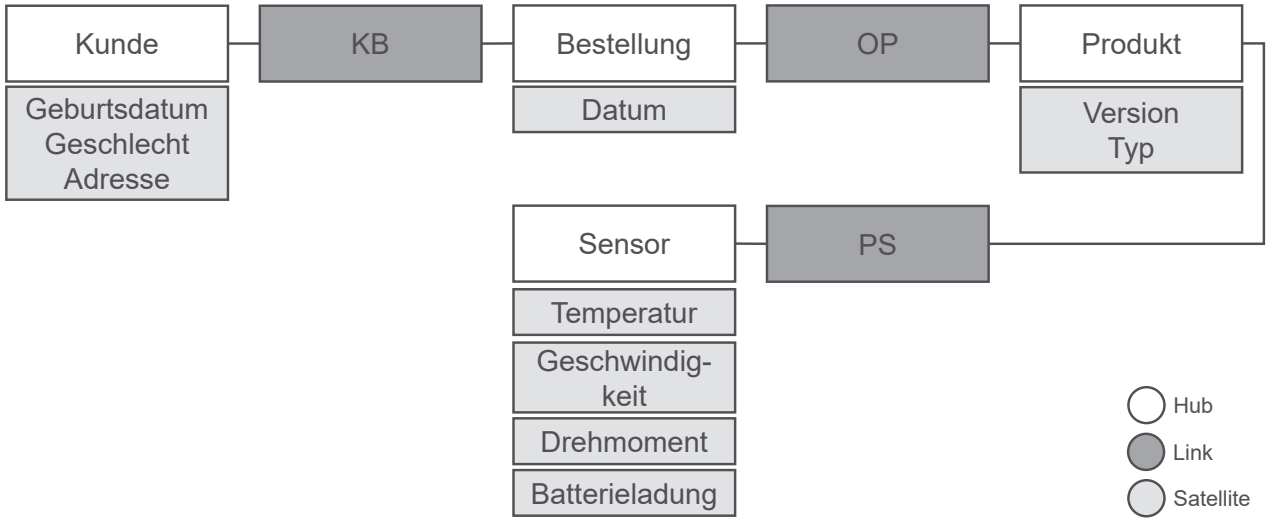


Abbildung 8.3: Das Data-Vault-Schema für das Anwendungsszenario

weitert. Die Distilled Zone enthält auch eine Kopie der Kundendaten als flache Tabelle für die Nutzung in R2. Auch die Twitter-Daten für Fragestellung ML3 werden in dieser Zone abgelegt. Hierzu werden die Daten nach ihren enthaltenen Firmendaten gruppiert und ihre Anzahl pro Monat erfasst. Die so aggregierten Daten können zum einen direkt für Fragestellung ML3 verwendet, als auch für spätere Analysen wieder aufgegriffen werden.

Für die Explorative Zone müssen die Daten nicht definiert werden, da sie durch Data Scientists so aufbereitet werden, wie diese sie benötigen. Hier wird die Analyse der Twitter-Daten für Fragestellung ML3 durchgeführt. Die Delivery Zone schließlich enthält die Daten im definierten Finalen Format (siehe Schritt B).

Schritt G: Datenpipeline definieren. Die Datenpipeline, wie sie in diesem Abschnitt definiert wird, ist nicht nur dafür zuständig, Daten zwischen Zonen zu bewegen, sondern auch dafür, Daten in die entsprechenden Formate zu transformieren. Für alle Anwendungsfälle außer dem Streaming-Anwendungsfall werden hierfür periodisch ausgeführte ETL-Prozesse verwendet. Diese Prozesse werden auf großen Mengen von Daten ausgeführt, was in einer langen Ausführungszeit resultiert. Da die Zeitanforderungen in diesen Anwendungsfällen nicht sehr streng sind, spielt dies jedoch keine entscheidende Rolle. Der Reporting-Anwendungsfall fordert, dass Daten innerhalb von einer Stunde verfügbar sind, während alle anderen Anwendungsfälle sogar noch längere Wartezeiten erlauben. Für den Streaming-Anwendungsfall dagegen sind die

Zeitanforderungen deutlich essentieller. Hier müssen Daten so schnell wie möglich verarbeitet werden, bevorzugt in Echtzeit. Daher werden Daten nicht in Stapeln sondern als Datenstrom verarbeitet. Mehr Information zu den ETL-Prozessen und der Datenstromverarbeitung folgt in Abschnitt 8.4.

Schritt H: Technologien definieren. Dieser Schritt wird in Übereinstimmung mit Schritt 5 der DLAF-Methodik ausgeführt. Abbildung 8.4 zeigt eine Übersicht über die Technologien, die in den verschiedenen Zonen verwendet wurden. Wie in Schritt 2 der DLAF-Methodik definiert, wird BRAID als Konzept für den Datenfluss verwendet. MySQL als RDBMS ist verfügbar in allen Zonen außer der Explorative Zone und enthält strukturierte Daten zu Kunden, Bestellungen, Produkten und Sensoren. Die semi-strukturierten Sensordaten werden allerdings nicht in MySQL gespeichert aufgrund ihrer Menge. In der Landing Zone und Raw Zone werden sie als JSON-Nachrichten im HDFS gespeichert. Beginnend ab der Harmonized Zone werden sie in der Zeitreihendatenbank InfluxDB gespeichert. Der Grund hierfür ist, dass Zeitreihendatenbanken zeitorientierte Abfragen wie Aggregationen oder Filterung nach Zeitintervallen effizient unterstützen. Da alle Sensormesswerte einen Zeitstempel beinhalten, lassen sie sich gut mit Zeitreihendatenbanken verwalten. Die Daten aus sozialen Netzwerken, die für Fragestellung ML3 benötigt werden, werden ausschließlich im HDFS gespeichert. Zusätzlich zu HDFS, MySQL und InfluxDB wird Cassandra für die Speicherung flacher Tabellen in der Distilled Zone und der Delivery Zone verwendet. Der Grund hierfür ist, dass Kundendaten oft spärlich besetzt sind und auf spaltenorientierte Art und Weise abgefragt werden. Obwohl das HDFS, MySQL, InfluxDB und Cassandra mehrmals in Abbildung 8.4 abgebildet sind, können sie alle mit einer einzelnen Instanz über alle Zonen realisiert werden. Für die Verarbeitung wird Python mit verschiedenen Datenbankkonnektoren verwendet, zusammen mit PySpark und Spark Streaming (siehe Legende unten in Abbildung 8.4). Mithilfe von Python werden die verschiedenen Datenbanken verbunden und Transformationen auf den Daten ausgeführt. PySpark interagiert mit HDFS, um Sensormesswerte zu lesen und zu verarbeiten. Spark Streaming verarbeitet die eingehenden Datenströme für den Streaming-Anwendungsfall. Zudem wird Apache Kafka verwendet, um Daten in der Landing Zone zu puffern und Datenströme von einer Zone zur anderen zu übertragen.

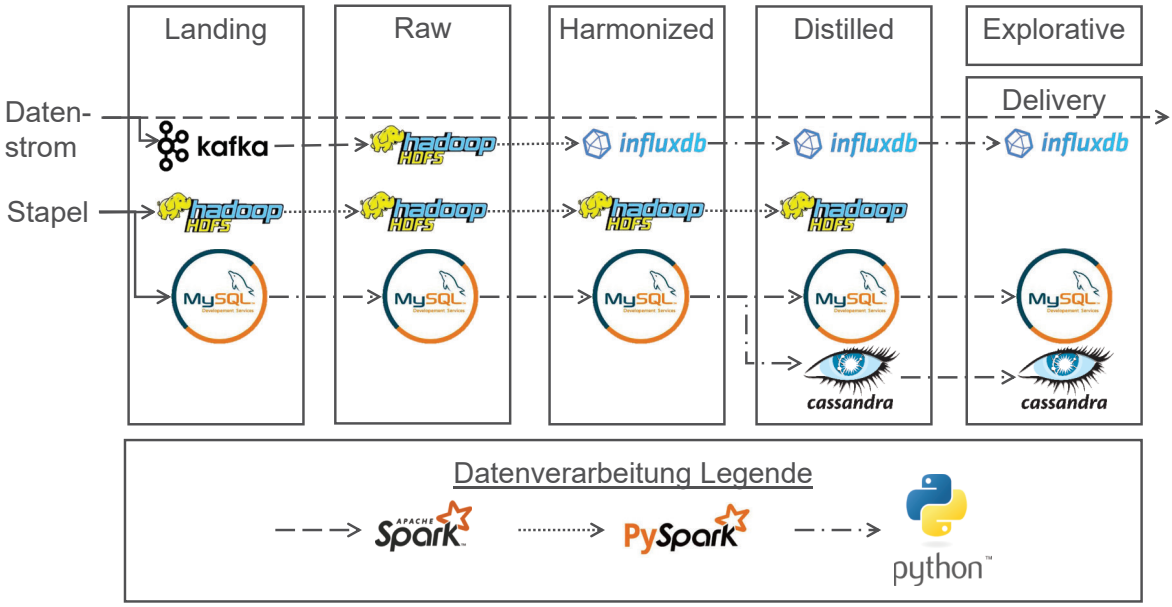


Abbildung 8.4: Übersicht über die Technologien für die Umsetzung des Zonenreferenzmodells

Schritt I: Implementierung. Eine detaillierte Implementierungsbeschreibung folgt in Abschnitt 8.4.

8.4 Implementierung des prototypischen Data Lake

Nachdem die konzeptionelle Umsetzungen des DLAF und des Zonenreferenzmodells abgeschlossen sind, befasst sich dieser Abschnitt mit der physischen Implementierung des prototypischen Data Lake. Ein Data Lake in seiner Gesamtheit umfasst zahlreiche (Teil-)Konzepte, die es umzusetzen gilt. Um beispielsweise den Data Lake wie in Abschnitt 8.2 beschrieben umzusetzen, müssen alle neun Aspekte des DLAF realisiert werden. Diese Umsetzung würde allerdings den Rahmen dieser Arbeit sprengen. Darum fokussiert sich die Implementierung des prototypischen Data Lake auf eine Teilmenge stark verknüpfter Aspekte, namentlich Infrastruktur, Datenspeicher, Datenfluss, Datenmodellierung und Datenorganisation. Diese werden dabei wie in Abschnitt 8.3 beschrieben umgesetzt. Teilweise wurden Sichten auf die Daten verwendet, anstatt die Daten zu materialisieren. Eine Materialisierung der Daten ist allerdings unumgänglich, wenn Daten von einem Speichersystem in ein anderes übertragen werden, oder wenn neue Primär- und Fremdschlüssel erstellt werden müssen.

Der prototypische Data Lake erstreckt sich über mehrere Virtual Machines (VMs) im gleichen Netzwerk, auf denen die verschiedenen Systeme laufen. Hierbei werden die Daten nach dem Zonenspeichermuster „Polyglott - Anwendungsorientiert“ gespeichert und verwaltet (siehe Abschnitt 7.4.2). Dieses Zonenspeichermuster wurde ausgewählt, da es hohe Funktionalität bietet und dabei eine geringere Komplexität besitzt als das Muster „Polyglott - Best-Fit“. Für die Umsetzung der Machine-Learning-Anwendungsfälle wird Jupyter Lab zusammen mit PySpark in Version 3.1.1 und Python Version 3.7.0 auf einer VM mit 8 CPUs und 2GB RAM verwendet. Diese Umsetzung ist gesondert vom Rest des Data Lake, da Data-Science-Anwendungsfälle typischerweise in einer separaten Umgebung, auch Sandbox genannt, umgesetzt werden. Der Rest des Data Lake setzt sich zusammen wie folgt: MySQL in Version 8.0.21 läuft auf einer VM mit 16GB RAM und 8 virtuellen CPUs. Hadoop läuft in Version 3.0.0 auf einem Cluster aus 11 Knoten (1 Master, 10 Slaves) mit jeweils 32GB RAM und 6 CPUs. Dem HDFS stehen 3,4TB Speicherplatz zur Verfügung. Cassandra in Version 3.11.6 und InfluxDB in Version 1.7.10 laufen jeweils auf einer eigenen VM mit 8GB RAM und 4 virtuellen CPUs. Die ETL-Prozesse wurden mit Python 3.8 und Spark in Version 2.4.0 implementiert. Die Speicherung und

Verarbeitung der Datenströme läuft auf einer eigenen VM mit 32GB RAM und 6 virtuellen CPUs. Hier werden Spark in Version 2.4.6 und Kafka in Version 2.4.0 eingesetzt.

Innerhalb der einzelnen Systeme sind die verschiedenen Zonen als Schemata umgesetzt. So gibt es z. B. in MySQL für die Raw Zone, Harmonized Zone, Distilled Zone und Delivery Zone jeweils ein eigenes Schema oder entsprechende Ordnerstrukturen im HDFS. Dies entspricht dem Zonenstrukturmuster „Struktur durch Schemata“ aus Abschnitt 7.4.1. Dieses Muster wurde gewählt, da die Komplexität des resultierenden Gesamtsystems gering bleibt und die Trennung direkt in den Systemen enthalten ist. Durch die Verwendung von HANDLE [EGG+20] als Metadatenmanagementmodell kann zusätzlich noch das Zonenstrukturmuster „Struktur durch Metadaten“ angewandt werden. Darauf wurde jedoch in diesem Prototypen aus Einfachheitsgründen verzichtet. Innerhalb von MySQL werden die Daten mithilfe von Structured Query Language (SQL)-Anfragen transformiert, welche über einen Python-Konnektor an die Datenbank gesendet werden. Listing 8.1 zeigt die Verwendung eines solchen Konnektors. In Zeile 1 bis 7 wird die Verbindung zur MySQL-Datenbank aufgebaut. Die If-Abfrage in Zeile 14 prüft, ob der Produkt-Hub, der in der Harmonized Zone aus den Produktdaten der Raw Zone erstellt werden soll, bereits existiert. Ist dies nicht der Fall, wird in Zeile 30 die zuvor zusammengesetzte SQL-Anfrage zur Erstellung des Hubs an die Datenbank geschickt.

Das Verschieben und Transformieren von Daten innerhalb eines Systems funktioniert immer ähnlich, unabhängig vom konkreten System. Sobald allerdings Daten zwischen Systemen verschoben werden sollen, müssen die Daten abgefragt, in Python transformiert und in ein anderes System geschrieben werden. Listing 8.2 zeigt die Berechnung der KPIs zur Batterieladung auf Grundlage von Daten aus InfluxDB für die Distilled Zone in MySQL. In Zeile 2 werden die Daten aus InfluxDB abgerufen, bevor sie in Zeile 5 verarbeitet werden. In Zeile 14 wird der Abfall der Batterieladung berechnet. Dieses Ergebnis wird schließlich in Zeile 16 über einen SQL-Befehl in die MySQL-Datenbank geschrieben.

Die Daten aus dem HDFS sind nicht per Konnektor direkt aus Python zugreifbar. Hier wurde mithilfe von PySpark auf die Daten zugegriffen. So konnte Python-Code in Spark ausgeführt werden. Der ausgeführte Code

```
    # Set up the connection to MySQL
    mysql_conn = mysql.connect(
                 host = dbConfig.mysql_ip,
                 database = dbConfig.mysql_harm_db,
                 user = dbConfig.mysql_username,
                 passwd = dbConfig.mysql_password
                 )

    # Log what's happening
    print("---------------")
    print("Create Product Hub started")

    # Check whether the hub exists
    if checkIfTableExists(mysql_conn, n.h_productHub):
        # Hub exists, don't do anthing else aside from log
        # Log
        print("Table {} already exists, no further steps
        ↪ necessary".format(n.h_productHub))

    else:
        # Hub does not exist, create it
        # Create harmonized Data Vault table on data from the
        ↪ raw table
        select = """SELECT MD5(businesskey) AS hashkey,
        ↪ businesskey AS businesskey,
                '2019-01-01' AS loaddate, 'SAP' AS
↪ recordsource FROM {}""".format(n.r_product)
        columnsToCreate = """hashkey VARCHAR(32) PRIMARY KEY,
        ↪ businesskey VARCHAR(25) NOT NULL,
                        loaddate DATETIME, recordsource
↪ VARCHAR(100)"""
        createTable = "CREATE TABLE {} ({}) AS
        ↪ ({})".format(n.h_productHub, columnsToCreate,
        ↪ select)

        # Execute the statement
        c = mysql_conn.cursor()
        c.execute(createTable)
```

Quelltext 8.1: Pythoncode zur Erstellung des Produkt-Hubs für die Harmonized Zone

```
    # Get the KPIs from InfluxDB
    query = "SELECT value FROM {} WHERE value > 0 GROUP BY
     ↪  sid".format(n.h_battery)
    result = influx_conn.query(query)

    for series in result.raw['series']:
        sid = series['tags']['sid']
        firstTSString = series['values'][0][0]
        lastTSString = series['values'][len(series['values'])
         ↪  - 1][0]

        # m = (y2-y1)/(x2-x1)
        deltaY = -100 # y2 = 0, y1 = 100
        deltaX = (lastTS - firstTS).total_seconds() # Seconds
         ↪  from batterycharge = 100 to batterycharge = 0

        slope = deltaY/deltaX

        insertKPIs = """UPDATE {} SET
                    s_slopeBattery = {}
                    WHERE hashkey =
↪  MD5('{}')""".format(n.d_sensorKPI, slope, sid)
```

Quelltext 8.2: Pythoncode zur Berechnung von KPIs und Übertragung von InfluxDB in MySQL

selbst ähnelt wieder den dargestellten Quelltexten. Für die Machine-Learning-Anwendungsfälle werden die verarbeiteten Twitterdaten wieder ins HDFS zurückgeschrieben, um für spätere Analysen zur Verfügung zu stehen.

Während ruhende Daten in jeder Zone persistiert werden und dort direkt abgefragt werden können, ist dies für Datenströme nicht möglich. Stattdessen stellt jede Zone im Zonenreferenzmodell einen Verarbeitungsschritt dar, nach dem die Daten wieder als Datenstrom über verschiedene Kafka-Topics verfügbar sind. Dabei wurden die Zonen als einzelne Anwendungen realisiert, wie in Abbildung 8.5 dargestellt. Dies entspricht dem Zonenflussmuster „Datenstromzonenmodell“ aus Abschnitt 7.4.3. Dieses Zonenflussmuster wurde gewählt, um von der Verfügbarkeit von Zwischenergebnissen in der Daten-

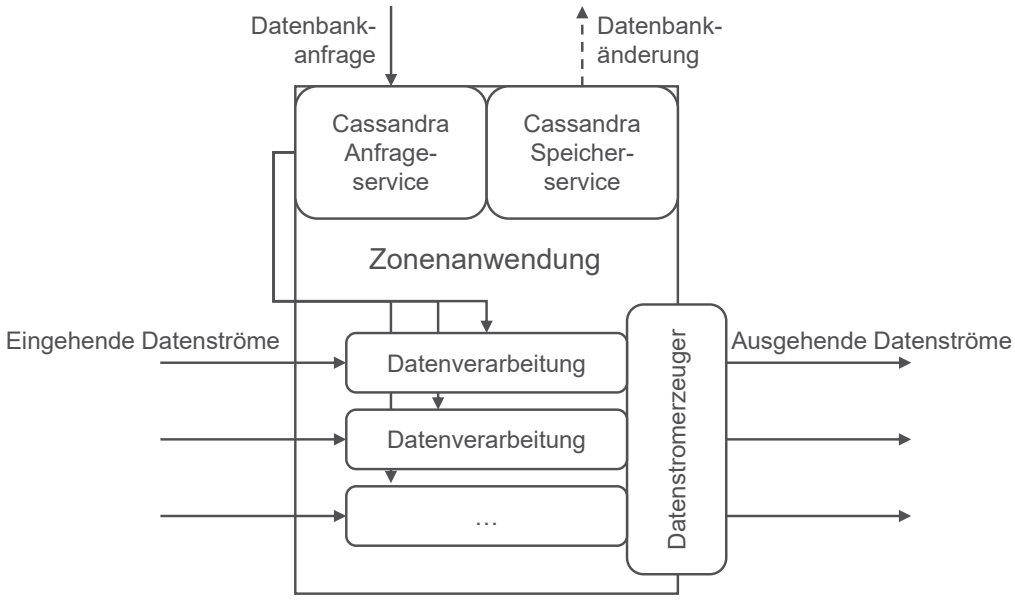


Abbildung 8.5: Die Zonenanwendung für die Verarbeitung von Datenströmen

stromverarbeitung Gebrauch machen zu können. Jede Zone enthält mehrere Datenverarbeitungsprozesse, die auf unterschiedliche Eingangsdatenströme angewandt werden. Die Ergebnisse werden durch den Datenstromerzeuger (rechts in Abbildung 8.5) in Ausgangsdatenströme umgewandelt. Zudem verfügt jede Zonenanwendung über eine Verbindung zu Cassandra, welche einerseits Anfragen an die Datenbank erlaubt (Cassandra Anfrageservice) und andererseits die Ergebnisse der Datenstromverarbeitung persistiert (Cassandra Speicherservice). So können bereits gespeicherte Ergebnisse wiederverwendet werden und berechnete Ergebnisse weiterhin zur Verfügung gestellt werden.

8.5 Zusammenfassung

Dies schließt die Implementierung des prototypischen Data Lake ab. Zwar wurden im Rahmen dieser Arbeit nur ausgewählte Aspekte des DLAF physisch umgesetzt, namentlich Infrastruktur, Datenspeicher, Datenfluss, Datenmodellierung und Datenorganisation, jedoch bietet Abschnitt 8.2 Hinweise zur Umsetzung der fehlenden Aspekte. So können diese in einer aufbauenden Arbeit ebenfalls realisiert werden, um so eine vollständige Implementierung einer vollständigen Data-Lake-Architektur zu erreichen. Auch wurde für die Implementierung des Zonenreferenzmodells lediglich auf strukturierte und

semi-strukturierte Daten zurückgegriffen. Eine Betrachtung zur Nutzung unstrukturierter Daten steht aus Zeit- und Platzgründen bislang aus. Hierdurch wird die Allgemeingültigkeit des Zonenreferenzmodells jedoch nicht eingeschränkt, da es sich hierbei lediglich um eine Fragestellung zur Umsetzung und nicht zum Konzept handelt.

Die prototypische Implementierung belegt die Anwendbarkeit der in dieser Arbeit entwickelten Konzepte für ein reales Anwendungsszenario. Mithilfe des DLAF und der zugehörigen Methodik (siehe Kapitel 5) sowie der Methodik für die Umsetzung einer zonenbasierten Datenorganisation (siehe Abschnitt 7.5) können die Details für die Umsetzung der Data-Lake-Architektur ausdefiniert werden. Insbesondere bietet die in Abschnitt 7.4 geführte Diskussion verschiedener Implementierungsmuster eine Entscheidungsgrundlage für die Realisierung des gewählten Zonenmodells. Während dieser Prototyp auf Grundlage eines konkreten Anwendungsszenarios erstellt wurde, sind die gebotene Anleitung und Entscheidungsunterstützung der in dieser Arbeit vorgestellten Konzepte allgemeingültig anwendbar. Damit bieten die Ergebnisse dieser Arbeit einen systematischen Leitfaden für die Definition und Implementierung einer vollständigen Data-Lake-Architektur inklusive einer zonenbasierten Datenorganisation.

Kapitel 9

Gesamtevaluation

Insgesamt acht Forschungsbeiträge zu drei Forschungszielen wurden im Rahmen dieser Arbeit erbracht. Hierzu wurden zwei neue Hauptkonzepte für Data Lakes entwickelt, namentlich das Data Lake Architecture Framework (DLAF) in Kapitel 5 und das Zonenreferenzmodell in Kapitel 7, welche Forschungsziele Z2 und Z3 erfüllen. Nachdem in Kapitel 8 bereits die Anwendbarkeit beider Konzepte gezeigt wurde, bildet die weitere Evaluation den Fokus dieses Kapitels. Die Evaluation anderer Beiträge dieser Arbeit, wie des Metamodells für Zonen (Kapitel 6) sowie der Implementierungsmuster für Zonenmodelle (Abschnitt 7.4) findet in den jeweiligen Kapiteln statt. Abschnitt 9.1 diskutiert die Vollständigkeit des DLAF, indem die enthaltenen Aspekte mit existierenden Realweltimplementierungen abgeglichen werden. So kann gezeigt werden, dass das DLAF alle notwendigen Aspekte adressiert. Zudem zeigt dieser Abgleich auch, wie existierende Data-Lake-Architekturen mithilfe des DLAF auf ihre Vollständigkeit überprüft und gegebenenfalls erweitert werden können. In Abschnitt 9.2 steht die Evaluation des Zonenreferenzmodells im Vordergrund. Zwar wird für diese Evaluation ein spezifisches Zonenmodell und ein konkretes Fallbeispiel als Grundlage verwendet, allerdings sind die allgemeinen Aussagen auf alle Zonenmodelle und Data Lakes zutreffend. Dies liegt daran, dass sowohl das Anwendungsszenario als auch die implementierten

Anwendungsfälle repräsentativ für Data Lakes sind (siehe Abschnitt 8.1). Die konkreten Zahlen hängen dabei natürlich von der Implementierung und den genutzten Technologien ab. Abschnitt 9.3 schließt dieses Kapitel mit einer kurzen Zusammenfassung ab.

Dieses Kapitel stellt eine überarbeitete Version vorangegangener Veröffentlichungen der Autorin dar [GGH+21; GGH+20b].

9.1 Evaluation des DLAF

Abschnitt 8.2 zeigt, dass das DLAF für die Konzeption einer vollständigen Data-Lake-Architektur anwendbar ist. Um darüber hinaus die Vollständigkeit des DLAF zu evaluieren, nutzt dieser Abschnitt zwei reale Data-Lake-Implementierungen als Basis, namentlich AIRPORTS DL [MBG+17] und das Smart Grid Big Data Ecosystem [MM18]. Diese Implementierungen wurden ausgewählt, da sie detaillierte Informationen zu den Konzepten bieten, die ihnen zugrunde liegen, und mit Echtweltdaten und -Anwendungsfällen evaluiert wurden. Sie decken zwei unterschiedliche Domänen ab (Luftverkehr und Smart Grids) und müssen sich darum auch mit unterschiedlichen Anforderungen zur Datenverwaltung und -verarbeitung auseinandersetzen. Keine der Veröffentlichungen zu den Implementierungen bietet eine Methodik, mit welcher der Data Lake konfiguriert wurde. Zwar existieren weitere Arbeiten zu Data-Lake-Implementierungen, jedoch bieten diese typischerweise einen geringeren Detailgrad als die gewählten Beispiele (z. B. [TSRC15]).

Tabelle 9.2 teilt die getroffenen Entwurfsentscheidungen dieser Implementierungen den jeweiligen Aspekten des DLAF zu. Basierend auf dieser Einteilung zeigt sich, dass alle getroffenen Entwurfsentscheidungen aus den existierenden Data-Lake-Architekturen in das DLAF eingeordnet werden können. Es wurden somit keine notwendigen Aspekte übersehen. Zudem können aufgrund dieser Einteilung die Vollständigkeit der betrachteten Implementierungen bewertet und passende Erweiterungen vorgeschlagen werden. Eine detaillierte Diskussion dieser Einteilung erfolgt in Abschnitt 9.1.1 für AIRPORTS DL und respektive in Abschnitt 9.1.2 für das Smart Grid Big Data Eco-System.

DLAF-Aspekt	AIRPORTS DL [MBG+17]	Smart Grid Big Data Eco-System [MM18]
A. Infrastruktur	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. Datenspeicher	Einzelnes Dateisystem	Einzelnes Ökosystem
C. Datenfluss	Daten werden als Datenstrom erfasst, aber in Stapeln verarbeitet	Basierend auf der Lambda-Architektur werden Datenströme per Datenstrom- und Stapelverarbeitung verarbeitet
D. Datenmodellierung	Rohdaten, AIRPORTS Data Model	Undefiniert
E. Datenorganisation	Vier-Zonen-Architektur	Zwei-Zonen-Architektur basierend auf der Lambda-Architektur
F. Datenprozesse	Verarbeitungspipeline für Nachrichten (ETL-Prozesse), Prozesse für Datenaufnahme und -Nutzung	Verarbeitungspipeline der Lambda-Architektur
G. Metadatenmanagement	Apache Atlas	Undefiniert
H. Datensicherheit & Privatheit	Veränderungen an den Daten nachverfolgen	Undefiniert
I. Datenqualität	Veränderungen an den Daten nachverfolgen, Qualität durch Zonen	Undefiniert

Tabelle 9.2: Kategorisierung der Entwurfsentscheidungen aus den betrachteten Implementierungen mithilfe des DLAF [GGH+21]

9.1.1 AIRPORTS DL

Der Fokus des AIRPORTS DL [MBG+17] liegt auf der Speicherung und Verwaltung von Flugüberwachungsdaten, wie die Position oder Flughöhe eines Flugzeugs. Diese Daten werden mit Daten Dritter verbunden, wie Wetterdaten, und per Datenstrom in den Data Lake aufgenommen. Die mittlere Spalte in Tabelle 9.2 zeigt die Entwurfsentscheidungen auf, wie sie in dieser Data-Lake-Implementierung hinsichtlich der DLAF-Aspekte getroffen wurden. Im Folgenden werden ausgewählte Aspekte näher betrachtet.

Die Veröffentlichung zum AIRPORTS DL [MBG+17] enthält keine explizite Beschreibung des *Datanflussaspekts* neben der Erwähnung, dass Daten als Datenstrom aufgenommen werden. Vor der Verarbeitung werden Daten allerdings gespeichert. Darüber hinaus basieren die Werkzeuge, die für die Verarbeitung verwendet werden (z. B. Hadoop MapReduce, Apache Pig), auf Stapelverarbeitung. Diese Entwurfsentscheidungen legen nahe, dass Daten in Stapeln verarbeitet werden und nicht als Datenstrom.

Für den Aspekt der *Datenmodellierung* werden Daten zunächst als rohe Schlüssel-Werte-Nachrichten gespeichert. Anschließend werden die Daten während ihrer Verarbeitung transformiert, um dem AIRPORTS-Datenmodell zu entsprechen, welches für dieses Anwendungsszenario entworfen wurde.

Die *Datenorganisation* wurde mithilfe eines Zonenmodells umgesetzt, welches aus vier Zonen besteht, die in der Veröffentlichung als „Layer“ bezeichnet werden. Diese Layer sind 1) Raw Layer, 2) Alignment Layer, 3) Flight Leg Reconstruction Layer und 4) Integration Layer. Daten werden in den Raw Layer aufgenommen und von Layer zu Layer weiter verarbeitet. Schließlich werden Daten im Delivery Layer verfügbar gemacht, wobei es sich nicht um einen Verarbeitungs-Layer handelt, sondern um ein Interface zum Data Lake.

Die *Datenprozesse* in dieser Data-Lake-Implementierung fokussieren sich vorrangig auf die Bewegung der Daten zwischen den Layern. Zusätzlich ist definiert, wie Daten in den Data Lake aufgenommen werden und wie sie analysiert und externen Systemen zur Verfügung gestellt werden.

Apache Atlas[1] wird für die Umsetzung eines Governance-Systems verwendet, welches *Metadatenmanagement* sowie grundlegende Funktionen zur Erhaltung

[1] atlas.apache.org/

von *Datensicherheit & Privatheit* und *Datenqualität* bietet. Letztere werden durch die Nachverfolgung von Änderungen an den Daten erreicht. Die Datenqualität wird auch durch das Zonenmodell des AIRPORTS DL unterstützt, wo die Qualität der Daten von einem Layer zum anderen erhöht wird.

Insgesamt können alle Entwurfsentscheidungen des AIRPORTS DL durch die Aspekte des DLAF abgebildet werden. Es zeigt sich auch, dass AIRPORTS DL für jeden der DLAF-Aspekte ein Konzept bereitstellt. So deckt der AIRPORTS DL alle Aspekte von Infrastruktur bis Datenqualität ab und erfüllt damit das Vollständigkeitskriterium.

9.1.2 Smart Grid Big Data Eco-System

Die zweite Data-Lake-Implementierung, die für diese Betrachtung herangezogen wird, findet Anwendung in einem Szenario zu Smart Grids als Teil eines Smart-Grid-Big-Data-Ökosystems [MM18]. Die Daten, die in diesem Szenario gespeichert und verarbeitet werden, sind hochgradig divers und beinhalten unter anderem Sensordaten von beispielsweise Kraftwerken und Verbrauchern, aber auch Bilder und Videos von Kraftwerküberwachungskameras. Diese Daten werden mit Daten aus zusätzlichen Quellen angereichert, z. B. Wetterdaten.

Alle Daten werden als Datenstrom in den Data Lake aufgenommen. Die rechte Spalte von Tabelle 9.2 fasst die Entwurfsentscheidungen in dieser Data-Lake-Implementierung zusammen. Im Folgenden wird auf einige davon näher eingegangen.

Der *Datenspeicher* dieses Data Lake ist über ein einzelnes Ökosystem realisiert, nämlich das Hadoop-Ökosystem wie in *Infrastruktur* detailliert. Dies beinhaltet unter anderem das Hadoop Distributed File System (HDFS), Hive und Impala.

Das Konzept für den *Datenflussaspekt* dieses Data Lake basiert auf der Lambda-Architektur [MW15]: Daten, die als Datenstrom aufgenommen werden, werden sowohl an die Stapelverarbeitung weitergeleitet, wo sie persistiert werden, als auch an die Datenstromverarbeitung, wo sie in Echtzeit verarbeitet werden (siehe Abschnitt 2.2.2). Die Ergebnisse beider Verarbeitungsmodi werden im Serving Layer kombiniert und zur Verfügung gestellt.

Die Nutzung der Lambda-Architektur nimmt Einfluss auf den Aspekt der *Datenorganisation*. Hierdurch wird der Data Lake in jeweils zwei Zonen für ruhende Daten und Datenströme unterteilt. Ruhende Daten werden in einem

Master Dataset als Rohzone persistiert und im Serving Layer zur Verfügung gestellt. Datenströme werden im Speed Layer als erste Zone verarbeitet und dann ebenfalls im Serving Layer für die Abfrage abgelegt. Auch die *Datenprozesse* werden komplett durch die Lambda-Architektur gegeben.

Während alle Entwurfsentscheidungen dieser Implementierung DLAF-Aspekten zugeordnet werden konnten, zeigt diese Kategorisierung, dass die Architektur des Smart Grid Data Lake nicht vollständig ist. Es gibt keinerlei Konzepte für *Datenmodellierung*, *Metadatenmanagement*, *Datensicherheit & Privatheit* oder *Datenqualität*. Ohne diese Konzepte riskiert der Data Lake, sich in einen Data Swamp zu wandeln, in dem Daten nicht auffindbar oder nutzbar sind [CSN+14]. Mithilfe des DLAF kann dieser Data Lake um die fehlenden Aspekte erweitert werden. Beispielsweise kann HANDLE [EGG+20] für das Metadatenmanagement verwendet werden, wodurch auch Datensicherheit & Privatheit teilweise unterstützt werden. Für die Datenqualität kann das verwendete Zonenmodell um entsprechende Vorgaben erweitert werden.

Wie der obigen Beschreibung entnommen werden kann, ermöglicht die Nutzung des DLAF eine strukturierte Herangehensweise an die Definition einer Data-Lake-Architektur. Die Einteilung der Entwurfsentscheidungen aus existierenden Data-Lake-Implementierungen zeigt, dass keine relevanten Aspekte in der Definition des DLAF übersehen wurden. Darüber hinaus wird sichtbar, wie das DLAF zur Bewertung und Verbesserung existierender Data-Lake-Architekturen verwendet werden kann. Im Falle des Data Lake aus dem Smart-Grid-Anwendungsszenario konnten durch die Bewertung durch das DLAF kritische Lücken in der Data-Lake-Architektur aufgedeckt werden. Diese lassen sich vermeiden, indem bei der Definition der Data-Lake-Architektur auf die DLAF-Methodik zurückgegriffen wird. Diese stellt sicher, dass alle relevanten Aspekte durch entsprechende Konzepte abgedeckt sind.

9.2 Evaluation des Zonenreferenzmodells

Der zweite Teil der Evaluation fokussiert sich auf das Zonenreferenzmodell. Abschnitt 9.2.1 beschreibt, wie das Zonenreferenzmodell die gestellten Anforderungen aus Abschnitt 7.1 erfüllt. Anschließend werden in Abschnitt 9.2.2 Evaluationsszenarien vorgestellt, anhand derer in Abschnitt 9.2.3 die proto-

typische Implementierung des Zonenreferenzmodells aus Abschnitt 8.4 mit einer zonenlosen Implementierung abgeglichen wird, um Vor- und Nachteile des Konzepts herauszuarbeiten.

9.2.1 Erfüllung der gestellten Anforderungen

Die konzeptionelle und physische Umsetzung des Zonenreferenzmodells in Kapitel 8 zeigt, dass das Zonenreferenzmodell anwendbar ist und auf gegebene Anwendungsszenarien angepasst werden kann. Dies ist möglich, da bestimmte Zonen für verschiedene Anwendungsfälle übersprungen werden können (z. B. die Explorative Zone in allen Anwendungsfällen außer ML1 und ML2). Das Zonenreferenzmodell bietet eine Leitlinie, indem es alle verfügbaren Zonen in systematischer Weise definiert. Alle Zonen sind physisch umsetzbar, wie die prototypische Implementierung zeigt.

In Abschnitt 7.1 wurden sieben allgemeingültige Anforderungen abgeleitet, welche ein Zonenmodell erfüllen muss, um eine Vielzahl praktisch relevanter Anwendungsfälle zu unterstützen. Die Erfüllung dieser Anforderungen ist somit für das Erreichen von Forschungsziel Z3 von hoher Wichtigkeit. Daher wurden sie als Basis zur Erstellung des Zonenreferenzmodells verwendet. Die folgenden Absätze detaillieren, wie die Anforderungen im Zonenreferenzmodell erfüllt werden.

Vorverarbeitet. Das Zonenreferenzmodell enthält vorverarbeitete Daten in der Distilled Zone, wo Daten für bestimmte Klassen von Anwendungsfällen aufbereitet sind. Beispielsweise sind Daten aggregiert, gefiltert oder anderweitig aufgewertet, um Analysen zu unterstützen.

Bereinigt. Die Harmonized Zone enthält Daten in einem syntaktisch bereinigten Format, während semantische Fehler in der Distilled Zone bereinigt werden können.

Integriert. Die standardisierte Modellierungstechnik, die in der Harmonized Zone vorgeschrieben ist, dient der Integration der Daten. Hier werden Daten aus unterschiedlichen Quellen verbunden und, falls innerhalb der Vorgaben der Harmonized Zone möglich, konsolidiert. Weitere Konsolidierung und Integration kann in der Distilled Zone vorgenommen werden.

Reguliert. Regulierung und Governance spielt in mehreren Bereichen des Zonenreferenzmodells eine Rolle. Zum einen wird in der Harmonized Zone

die Qualität der Daten sichergestellt. Hinzu kommt, dass sensible Daten im geschützen Bereich jeder Zone verwaltet werden und dort nur unter strengen Vorgaben verwendet werden können. Die systematischen Vorgaben dazu, wie Daten in verschiedenen Zonen vorzuliegen haben, ermöglichen darüber hinaus weitere Kontrolle über die Daten und ihren Status.

Reporting & Online Analytical Processing (OLAP). Die Delivery Zone bietet eine Funktionalität, die derer von Data Marts aus dem Data Warehousing ähnelt. Daten in dieser Zone sind speziell für ganz bestimmte Anwendungsfälle vorbereitet, insbesondere Reporting und OLAP.

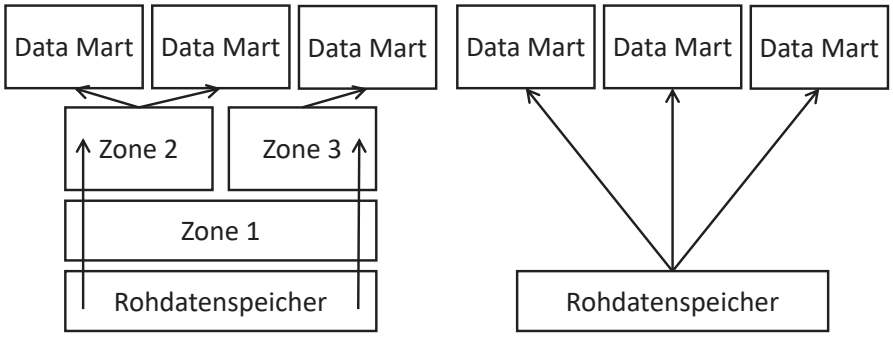
Fortgeschrittene Analysen. Die Explorative Zone bietet Data Scientists einen Ort, wo sie die Daten flexibel analysieren können. Hier sind beliebige Analysen möglich, unter anderem fortgeschrittene Analysen. Daten aus jeder anderen Zone, außer der Landing Zone, können hier importiert und zur Herleitung neuer Erkenntnisse verwendet werden.

Zurückschreiben. Während die Explorative Zone selbst nicht persistent ist, können vielversprechende Ergebnisse in die Distilled Zone zurückgeschrieben werden.

Diese Absätze zeigen, dass das Zonenreferenzmodell alle Funktionalitäten bietet, welche in Abschnitt 7.1 als notwendig für die praktische Anwendung herausgearbeitet wurden.

9.2.2 Evaluationsszenarien

Die folgenden Kapitel dienen dazu, die Vor- und Nachteile einer zonenbasierten Datenorganisation in Data Lakes zu evaulieren. Neben der prototypischen Implementierung für quantitative Messungen werden auch typische und repräsentative Szenarien für die Datennutzung in Data Lakes für die Evaluation verwendet. Aus dem Fallbeispiel wurden acht verschiedene repräsentative Evaluationsszenarien für die Datennutzung in Data Lakes identifiziert. Während diese Szenarien aus einem konkreten Data Lake stammen, konnten sie so verallgemeinert werden, dass sie auf jede Data-Lake-Implementierung zutreffen, sowohl für ruhende Daten als auch Datenströme. Sie stellen damit eine repräsentative Basis für die Evaluation dar. Die Evaluationsszenarien lauten wie folgt:



a) Zonenbasierter Data Lake b) Zonenloser Data Lake

Abbildung 9.1: Schematische Darstellung eines zonenbasierten und zonenlosen Data Lake

S1) Ein bereits implementierter Anwendungsfall wird ausgeführt.

S2) Ein neuer Anwendungsfall wird auf Daten, die bereits im Data Lake vorhanden sind, implementiert mit einer Geschäftslogik, die einem bereits implementierten Anwendungsfall ähnelt.

S3) Ein neuer Anwendungsfall wird implementiert, welcher keinerlei Ähnlichkeiten zu früheren Anwendungsfällen aufweist. Allerdings sind die benötigten Daten bereits in Rohform im Data Lake vorhanden.

S4) Ein neuer Anwendungsfall wird auf neuen Datenquellen implementiert, die erst zum Data Lake hinzugefügt werden müssen.

S5) Ein existierender Anwendungsfall wird um neue Daten ergänzt.

S6) Die Anforderungen eines existierenden Anwendungsfalls ändern sich (z. B. die geforderte Granularität einer Aggregation).

S7) Ein Data Scientist führt explorative/fortgeschrittene Analysen auf existierenden Daten aus.

S8) Ein Data Scientist führt explorative/fortgeschrittene Analysen auf Daten aus, die noch nicht im Data Lake sind.

9.2.3 Vergleichende Evaluation

In diesem Abschnitt werden die Vor- und Nachteile eines zonenbasierten Data Lake identifiziert. Hierzu wird der prototypische zonenbasierte Data Lake

aus Abschnitt 8.4 mit einem Data Lake ohne ein Zonenmodell, d. h. einem zonenlosen Data Lake, verglichen. Dieser zonenlose Data Lake umfasst einen Rohdatenspeicher und isolierte, anwendungsabhängige Data Marts, die auf diesem aufbauen. Beide Ansätze sind in Abbildung 9.1 dargestellt. Er wurde mit den selben Techniken und Systemen implementiert, die auch der zonenbasierte Data Lake nutzt. Der Unterschied ist, dass im zonenlosen Ansatz Daten direkt von ihrer Rohform in ihr finales Format transformiert und in isolierten Data Marts verfügbar gemacht werden. Weitere Verarbeitungszwischenschritte, wie bereinigte oder integrierte Daten, sind nicht wiederverwendbar. Zudem führt die isolierte Verarbeitung zu redundantem Code für Datenpipeline und Inkonsistenzen zwischen Data Marts. Das Ziel dieses Vergleiches ist es, Vor- und Nachteile eines zonenbasierten Data Lake gegenüber eines zonenlosen Data Lake zu identifizieren. Diese Erkenntnisse können Unternehmen Entscheidungsunterstützung bei der Definition ihrer Datenorganisation für den Data Lake bieten.

Der Vergleich wird über die folgenden Kriterien geführt, welche für Unternehmen von hoher Wichtigkeit sind: Transformationskomplexität, Verarbeitungszeit für die gesamte Verarbeitungspipeline von Anfang bis Ende, Echtzeitnähe für Datenstromverarbeitung, benötigter Speicherplatz, Wiederverwendbarkeit von Daten und Implementierungsunterstützung. Die Ergebnisse dieser Evaluationen sind in den folgenden Unterkapiteln gegeben, inklusive einer abschließenden Diskussion.

9.2.3.1 Transformationskomplexität

Die Transformationskomplexität umfasst die Anzahl der Operationen und die Zeit, die benötigt werden, um Daten für die nächste Zone oder das finale Format zu transformieren. Um diese Komplexität zu evaluieren, werden die Operatoren verglichen, welche für die Reporting-Anwendungsfälle R1 (Verkaufszahlen für bestimmte Komponenten und Versionen) und R2 (Verteilung von Bestellungen über Nutzergruppe) aus Tabelle 8.1 in beiden Varianten des Data Lake benötigt werden. Diese Anwendungsfälle wurden ausgewählt, da sie alle Zonen des Zonenreferenzmodells umfassen (außer der Explorative Zone) und regelmäßig im Anwendungsszenario ausgeführt werden. Da beide dieser Anwendungsfälle

```
/* Distilled Zone */
CREATE OR REPLACE VIEW distilledZone.BV_SAT_CUSTOMER AS
        SELECT hashkey, loaddate, loadenddate, recordsource,
         ↪  CASE
                WHEN (2020 - c_birthyear) > 70 THEN '70+'
                WHEN (2020 - c_birthyear) > 50 THEN '51-70'
                WHEN (2020 - c_birthyear) > 25 THEN '26-50'
                ELSE '18-25' END
        AS c_agerange, c_gender, c_address
        FROM harmonizedZone.RV_SAT_CUSTOMER

/* Point-in-Time-Tabelle in der Distilled Zone */
CREATE OR REPLACE VIEW distilledZone.BV_PIT_CUSTOMER AS
        SELECT hashkey, loaddate, recordsource, c_agerange,
         ↪  c_gender, c_address
        FROM distilledZone.BV_SAT_CUSTOMER
        WHERE loadenddate = DATE '9999-12-31'

/* Delivery Zone */
SELECT o.businesskey AS orderkey, osat.o_date, p.businesskey
  ↪  AS productkey, psat.p_version, psat.p_type, c.businesskey
  ↪  AS customerkey, csat.c_agerange, csat.c_gender,
  ↪  csat.c_address
FROM distilledZone.BV_HUB_ORDER o, distilledZone.BV_SAT_ORDER
  ↪  osat, distilledZone.RV_LINK_ORDERPRODUCT l_op,
  ↪  distilledZone.RV_HUB_PRODUCT p,
  ↪  distilledZone.RV_SAT_PRODUCT psat,
  ↪  distilledZone.BV_LINK_CUSTOMERORDER l_co,
  ↪  distilledZone.BV_HUB_CUSTOMER c,
  ↪  distilledZone.BV_PIT_CUSTOMER csat
WHERE o.hashkey = osat.hashkey
AND o.hashkey = l_op.orderkey
AND p.hashkey = l_op.productkey
AND p.hashkey = psat.hashkey
AND o.hashkey = l_co.orderkey
AND c.hashkey = l_co.customerkey
AND c.hashkey = csat.hashkey
```

Quelltext 9.1: Structured Query Language (SQL)-Anfragen für die Transformation der Kundendaten in der zonenbasierten Implementierung

```
1 SELECT o.businesskey AS orderkey, osat.o_date, p.businesskey
  ↪ AS productkey, p.productversion, p.producttype,
  ↪ c.businesskey AS customerkey, CASE
2    WHEN (2020 - EXTRACT(YEAR FROM c_birthdate)) > 70 THEN
     ↪ '70+'
3    WHEN (2020 - EXTRACT(YEAR FROM c_birthdate)) > 50 THEN
     ↪ '51-70'
4    WHEN (2020 - EXTRACT(YEAR FROM c_birthdate)) > 25 THEN
     ↪ '26-50'
5    ELSE '18-25' END
6 AS c_agerange, csat.c_gender, csat.c_address
7 FROM rawZone.RV_HUB_ORDER o, rawZone.RV_SAT_ORDER osat,
  ↪ rawZone.PRODUCT p, rawZone.RV_LINK_CUSTOMERORDER l_co,
  ↪ rawZone.RV_HUB_CUSTOMER c, rawZone.RV_SAT_CUSTOMER csat
8 WHERE csat.loadenddate = DATE '9999-12-31'
9 AND o.hashkey = osat.hashkey
10 AND p.businesskey = osat.o_productkey
11 AND o.hashkey = l_co.orderkey
12 AND c.hashkey = l_co.customerkey
13 AND c.hashkey = csat.hashkey
```

Quelltext 9.2: SQL-Anfrage für die Transformation der Kundendaten in der zonenlosen Implementierung

ausschließlich in MySQL ausgeführt werden, kann das EXPLAIN-Statement zum Vergleich der Transformationskomplexität verwendet werden.

In der zonenbasierten Implementierung wird eine SQL-Anfrage pro Transformationsschritt benötigt, d. h. es wird eine Anfrage von der Raw Zone zur Harmonized Zone ausgeführt, eine weitere Anfrage von der Harmonized Zone zur Distilled Zone und so weiter. So enthält jede Zone eine Repräsentation der Daten, auch wenn einige dieser Repräsentationen lediglich Sichten sind. Im Gegensatz dazu werden alle Transformationen in der zonenlosen Implementierung mit einer einzelnen SQL-Anfrage durchgeführt. Daher ist diese einzelne Anfrage deutlich länger und komplexer als die Teilanfragen der zonenbasierten Implementierung. Die benötigten SQL-Anfragen in der zonenbasierten Implementierung sind in Listing 9.1 dargestellt, die der zonenlosen Implementierung in Listing 9.2. Es zeigt sich, dass in der zonenbasierten Implementierung zwar

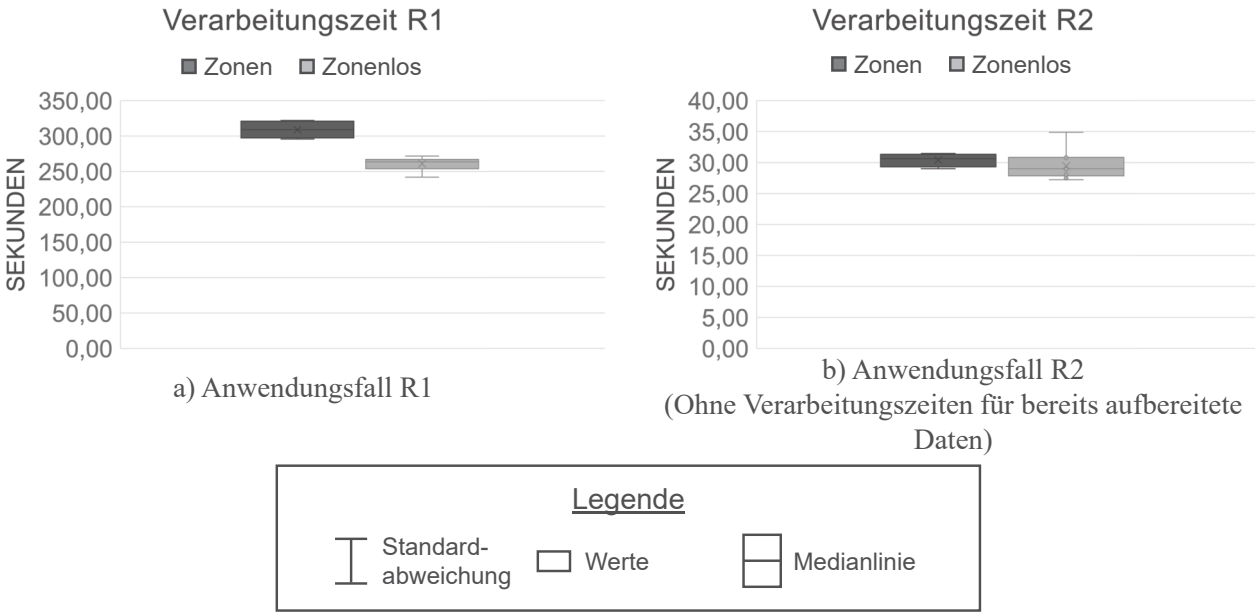
deutlich mehr Anfragen notwendig sind, diese jedoch kürzer und dadurch einfacher nachzuvollziehen sind.

Berechnet man allerdings die Operatorkosten mit dem EXPLAIN-Statement in MySQL zeigt sich, dass die Kosten für die zonenbasierte Implementierung (519,69 Sekunden) deutlich höher sind als für die zonenlose Implementierung (129,46 Sekunden). Der Grund hierfür liegt in der Verallgemeinerung des Zonenmodells: Daten werden zunächst für die Harmonized Zone in eine standardisierte Modellierungstechnik überführt, bevor sie in der Delivery Zone wieder zusammengeführt werden. Dies führt zu Verarbeitungsschritten und zusätzlichen JOIN-Operatoren, die in der zonenlosen Implementierung nicht benötigt werden, da hier Daten direkt in das finale Format überführt werden.

9.2.3.2 Verarbeitungszeit

Für das zweite Kriterium wird die Verarbeitungszeit der gesamten Aufbereitungspipeline in beiden Ansätzen verglichen. Diese Aufbereitungspipeline beginnt mit Rohdaten in der Raw Zone und transformiert die Daten, bis sie dem finalen Format entsprechen. Für den Vergleich wird die Verarbeitungszeit zur Realisierung der Anwendungsfälle R1, R2 und ML3 herangezogen. In der Praxis wird eine solche Aufbereitung bei den Reporting-Anwendungsfällen in periodischen Intervallen vorgenommen, z. B. stündlich oder täglich. Die Verarbeitungszeiten, die hier erhoben wurden, fallen somit nicht an, wenn Daten abgefragt werden (Onlinephase), sondern nach einem vordefinierten Zeitplan, bevor die Daten verfügbar werden (Offlinephase).

Abbildung 9.2 zeigt die Verarbeitungszeit in Sekunden für die zonenbasierte und die zonenlose Implementierung auf der selben Infrastruktur. Abbildung 9.2.a) stellt die Verarbeitungszeit für Anwendungsfall R1 dar, während Abbildung 9.2.b) die von Anwendungsfall R2 darstellt. Dabei gilt zu beachten, dass in Abbildung 9.2.b) die Verarbeitungszeit für bereits aufbereitete Daten aus R1 in der zonenbasierten Messung ausgeklammert wurde. Der Vergleich zeigt, dass die zonenbasierte Implementierung insgesamt längere Verarbeitungszeiten benötigt. Dies kann darauf zurückgeführt werden, dass zusätzliche Transformationen für die Wiederverwendbarkeit der Daten benötigt werden, wie bereits unter Transformationskomplexität beschrieben.



a) Anwendungsfall R1

b) Anwendungsfall R2
(Ohne Verarbeitungszeiten für bereits aufbereitete Daten)

Abbildung 9.2: Verarbeitungszeiten für die Aufbereitungspipelines in Anwendungsfällen R1 und R2

9.2.3.3 Echtzeitnähe

Wird ein Data Lake für die Verarbeitung von Datenströmen genutzt, ist es von immenser Wichtigkeit, dass eine echtzeitnahe Verarbeitung möglich ist. Das Ziel ist es, Fehlverhalten direkt in den Daten zu erkennen und zeitnah darauf zu reagieren. Daher werden die zonenbasierte und die zonenlose Implementierung auf ihre Echtzeitnähe hin verglichen, d. h. ihre Latenz in der Datenstromverarbeitung. Hierzu wurde der Streaming-Anwendungsfall, wie in Abschnitt 8.1 beschrieben, als Basis verwendet und in beiden Ansätzen implementiert. Die Messungen wurden einige Minuten nach dem Start der Verarbeitung vorgenommen um sicherzustellen, dass keine Latenzen durch die Startzeiten von Komponenten entstehen. Der Durchschnitt sowie der Median der Verarbeitungszeit von 30.000 JavaScript Object Notation (JSON)-Nachrichten wurde erfasst. Diese Verarbeitungszeit umfasst die Zeit vom Eintreffen der Daten in ihrem Rohformat bis hin zum Vorliegen im finalen Format.

Abbildung 9.3 stellt die Ergebnisse dieser Messungen dar. Es ist klar sichtbar, dass Durchschnitt und Median der Verarbeitungszeit von Datenströmen in der zonenbasierten Implementierung mehr als 100-mal höher ist als in der zonenlosen Variante. Betrachtet man die Werte allerdings genauer, so sieht man, dass der Median der tatsächlichen Verarbeitung in jeder Zone zwischen 8

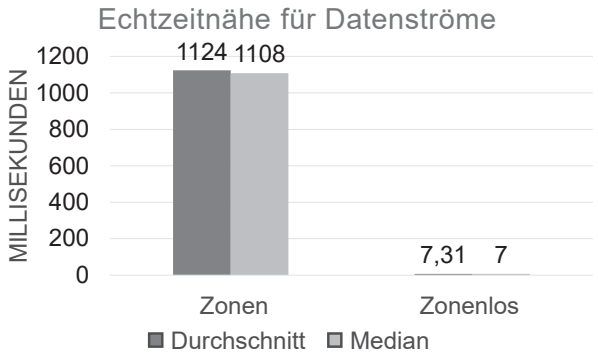


Abbildung 9.3: Durchschnitt und Median der Verarbeitungszeit für Datenströme sowohl im zonenbasierten als auch im zonenlosen Ansatz

und 10 Millisekunden liegt. Dies summiert sich zu lediglich 50 Millisekunden für alle fünf verwendeten Zonen im Gegensatz zu 7 Millisekunden aus dem zonenlosen Ansatz. Der hohe Mehraufwand entsteht durch das Übertragen der Daten von einer Zone zur nächsten über Kafka-Topics, welches zwischen 300 und 400 Millisekunden pro Transfer benötigt. Um daher Echtzeitverarbeitung in einem zonenbasierten Data Lake zu ermöglichen, muss dieser Mehraufwand reduziert werden. Selbst dann jedoch bietet der zonenlose Ansatz deutlich schnellere Verarbeitung für Datenströme, da wie unter Transformationskomplexität beschrieben bestimmte Transformationsschritte übersprungen werden können. Es bleibt zu prüfen, inwiefern die Echtzeitnähe bei der Nutzung des Zonenreferenzmodells erhöht werden kann.

9.2.3.4 Speicherplatz

Um den benötigten Speicherplatz zu evaluieren, den die Daten und ihre Möglichen Zwischenschritte einnehmen, werden sowohl Messungen aus der Implementierung als auch die Evaluationsszenarien aus Abschnitt 9.2.2 verwendet.

Als Basis für die Messungen wird erneut auf die Implementierung aus Anwendungsfällen R1 und R2 in sowohl dem zonenbasierten als auch dem zonenlosen Ansatz zurückgegriffen. Es zeigte sich, dass selbst in der prototypischen Implementierung der zonenbasierte Data Lake (73,79MB) mehr als doppelt so viel Speicherplatz benötigte wie der zonenlose Data Lake (27,54MB). Obwohl nicht-materialsierte Sichten verwendet wurden, wo es möglich war, mussten

manche Daten dennoch repliziert werden. Insbesondere bei der Transformation der Produktdaten von 3NF nach Data Vault mussten die Daten in der Harmonized Zone repliziert werden, da neue Primär- und Fremdschlüssel benötigt wurden. Dies führte zu den zusätzlichen 46,25MB an benötigtem Speicherplatz. Dieser zusätzliche Speicherbedarf steigt zudem erheblich, wenn mehrere unterschiedliche Systeme für die Speicherung und Verwaltung der Daten verwendet werden, da keine Sichten über Systeme hinweg möglich sind.

Die Evaluationsszenarien bieten keine konkreten Zahlen, sondern ermöglichen die Einschätzung, wie sich der benötigte Speicherplatz entwickelt. Tabelle 9.3 und Tabelle 9.4 fassen diese Entwicklung für die unterschiedlichen Evaluationsszenarien zusammen. Zusätzlich zu der textuellen Beschreibung wird die Entwicklung auch graphisch dargestellt: Ein weißer Kreis bedeutet keinen zusätzlichen Speicherplatz, ein halb gefüllter Kreis zeigt, dass zwar zusätzlicher Speicherplatz benötigt wird, jedoch weniger als im gegenübergestellten Ansatz. Ein voller Kreis zeigt letztendlich, dass der zusätzlich benötigte Speicherplatz mehr oder genauso groß ist wie im gegenübergestellten Ansatz.

Dieser Vergleich zeigt, dass in den Fällen, in denen sich die beiden Ansätze unterscheiden (S3-S6), der zonenbasierte Data Lake mehr Speicherplatz benötigt als der zonenlose Data Lake. Dies kommt daher, dass Zonenmodelle nicht nur die Rohdaten und das finale Format speichern, sondern auch Zwischenschritte in den Transformationen verfügbar machen. Der benötigte Speicherplatz kann durch den Einsatz von Sichten reduziert werden, allerdings erhöht sich dadurch die Transformationskomplexität bei der Abfrage.

9.2.3.5 Wiederverwendbarkeit der Daten

Um die Wiederverwendbarkeit der Daten im zonenbasierten und zonenlosen Ansatz zu vergleichen, wird erneut auf die definierten Evaluationsszenarien zurückgegriffen. Wiederverwendbarkeit beschreibt, dass Daten, die in einem Anwendungsfall verwendet werden, ohne weitere Transformationen auch in einem anderen Anwendungsfall verwendet werden können. Die Ergebnisse dieses Vergleichs sind in Tabelle 9.5 und Tabelle 9.6 dargestellt.

Szenarios	Speicherplatzentwicklung			
	Zonenbasiert		Zonenlos	
S1: Existierenden Anwendungsfall ausführen	Kein zusätzlicher Speicherplatz	○	Kein zusätzlicher Speicherplatz	○
S2: Neuer Anwendungsfall mit Ähnlichkeiten zu existierendem Anwendungsfall	Zusätzlicher Speicherplatz für finales Format (Delivery Zone)	●	Zusätzlicher Speicherplatz für finales Format	●
S3: Neuer Anwendungsfall mit vorhandenen Rohdaten	Zusätzlicher Speicherplatz für alle Zonen außer Raw Zone	●	Zusätzlicher Speicherplatz für finales Format	◑
S4: Neuer Anwendungsfall mit neuer Datenquelle	Zusätzlicher Speicherplatz für alle Zonen	●	Zusätzlicher Speicherplatz für Rohdaten und finales Format	◑

Tabelle 9.3: Übersicht über die Entwicklung des benötigten Speicherplatzes in den Evaluationsszenarien. ○ -Kein zusätzlicher Speicherplatz, ◑ -Weniger zusätzlicher Speicherplatz als im anderen Ansatz, ● -Mehr oder gleich viel zusätzlicher Speicherplatz wie im anderen Ansatz

Szenarios	Speicherplatzentwicklung			
	Zonenbasiert		Zonenlos	
S5: Zusätzliche Daten für bestehenden Anwendungsfall	Zusätzlicher Speicherplatz für alle Zonen	●	Zusätzlicher Speicherplatz für Rohdaten und finales Format	◑
S6: Neue Anforderungen	Zusätzlicher Speicherplatz nur dann benötigt, wenn die neuen Anforderungen zu mehr Datenvolumen führen (Distilled Zone, Delivery Zone)	●	Zusätzlicher Speicherplatz nur dann benötigt, wenn die neuen Anforderungen zu mehr Datenvolumen führen	◑
S7: Explorative Analysen auf existierenden Daten	Zusätzlicher Speicherplatz für Sandbox	●	Zusätzlicher Speicherplatz für Sandbox	●
S8: Explorative Analysen auf neuen Daten	Zusätzlicher Speicherplatz für Rohdaten und Sandbox	●	Zusätzlicher Speicherplatz für Rohdaten und Sandbox	●

Tabelle 9.4: Übersicht über die Entwicklung des benötigten Speicherplatzes in den Evaluationsszenarien (cont.)

Szenarios	Wiederverwendbarkeit			
	Zonenbasiert		Zonenlos	
S1: Existierenden Anwendungsfall ausführen	Volle Wiederverwendbarkeit	●	Volle Wiederverwendbarkeit	●
S2: Neuer Anwendungsfall mit Ähnlichkeiten zu existierendem Anwendungsfall	Wiederverwendung von bereinigten und integrierten Daten möglich	●	Keine Wiederverwendung möglich	○
S3: Neuer Anwendungsfall mit vorhandenen Rohdaten	Keine Wiederverwendung möglich	○	Keine Wiederverwendung möglich	○
S4: Neuer Anwendungsfall mit neuer Datenquelle	Keine Wiederverwendung möglich	○	Keine Wiederverwendung möglich	○

Tabelle 9.5: Übersicht über die Wiederverwendbarkeit der Daten in den Evaluationsszenarien. ○ -Keine Wiederverwendung möglich, ◑ -Wiederverwendung von Ergebnissen möglich aber nicht von Transformationen, ● -Wiederverwendung möglich

Szenarios	Wiederverwendbarkeit			
	Zonenbasiert		Zonenlos	
S5: Zusätzliche Daten für bestehenden Anwendungsfall	Wiederverwendung von Zwischenschritten aus anderen Anwendungsfällen	●	Keine Wiederverwendung möglich	○
S6: Neue Anforderungen	Wiederverwendung von ETL-Prozessen möglich	●	Wiederverwendung von ETL-Prozessen möglich	●
S7: Explorative Analysen auf existierenden Daten	Wiederverwendung von bereinigten und integrierten Daten möglich	●	Keine Wiederverwendung möglich	○
S8: Explorative Analysen auf neuen Daten	Wiederverwendung bereits gelernter Modelle möglich, nicht aber von Daten selbst	◑	Keine Wiederverwendung möglich	○

Tabelle 9.6: Übersicht über die Wiederverwendbarkeit der Daten in den Evaluationsszenarien (cont.)

Lediglich Szenarien S1, S2, S3 und S5-S7 qualifizieren sich für die Wiederverwendbarkeit von Daten, da in den anderen Szenarien keine Daten für neue Implementierungen wiederverwendet werden können. In S1 (Existierenden Anwendungsfall ausführen) kann in beiden Varianten auf vollständig vorverarbeitete Daten zurückgegriffen werden. In S2 (neuer Anwendungsfall mit Ähnlichkeiten zu existierendem Anwendungsfall) können im zonenbasierten Data Lake Daten aus der Harmonized Zone oder Distilled Zone wiederverwendet werden. Für den zonenlosen Ansatz kann die Logik der ETL-Prozesse kopiert werden, aber Daten müssen erneut verarbeitet werden. In S3 (Neuer Anwendungsfall mit vorhandenen Rohdaten) müssen beide Ansätze die Daten von ihrer Rohform ab verarbeiten. In S5 (Zusätzliche Daten für bestehenden Anwendungsfall) müssen diese zusätzlichen Daten für jede Zone im zonenbasierten Data Lake aufbereitet werden. Es kann allerdings sein, dass sie bereits aus anderen Anwendungsfällen im Data Lake vorliegen. In diesem Fall müssen die Daten lediglich in der Harmonized Zone miteinander verbunden werden. Von hier aus müssen Transformationen erneut durchgeführt werden, da die neuen Daten die bisherigen Ergebnisse beeinflussen können. Im zonenlosen Ansatz müssen die zusätzlichen Daten komplett neu aufbereitet werden, auch wenn sie bereits in anderen Anwendungsfällen genutzt wurden. In S6 (Neue Anforderungen) müssen lediglich die ETL-Prozesse in beiden Ansätzen angepasst werden. Es kann sich hierbei um den ETL-Prozess von der Harmonized Zone zur Distilled Zone handeln oder um den ETL-Prozess von den Rohdaten ab im zonenlosen Ansatz. Schließlich in S7 (Explorative Analysen auf neuen Daten) können Data Scientists vortransformierte Daten im zonenbasierten Ansatz auswählen, wenn diese zu ihrem Anwendungsfall passen, z. B. bereinigte und integrierte Daten aus der Harmonized Zone. Im zonenlosen Ansatz können sie dagegen nur Rohdaten nutzen.

Insgesamt ist die Wiederverwendbarkeit von Daten im zonenbasierten Data Lake hoch. Daten werden in Zwischenschritten gespeichert und verfügbar gemacht, mit Integration, Bereinigung und anderen zeitintensiven Schritten vorbereitet. Diese Datenaufbereitungen nehmen im allgemeinen zwischen 60-80% der Entwicklung analytischer Anwendungsfälle ein [Pre16]. Im Gegensatz dazu müssen Daten im zonenlosen Ansatz jedes Mal neu aufbereitet werden.

Schritt	Zonenbasiert	Zonenlos
Datenbeschaffung	0s	59,1s
Normalisierung	0s	32,6s
Aggregation	0s	97,6s
Datenanalyse	352,2s	497,1s
Gesamt	352,2s	686,4s

Tabelle 9.7: Laufzeitanalyse einer explorativen Analyse auf existierenden Daten

Tabelle 9.7 stellt den zeitlichen Vorteil dar, der sich durch die Wiederverwendung von Daten ergibt. Als Grundlage für diese Betrachtung wurde ML3 prototypisch umgesetzt und anschließend eine explorative Analyse auf den existierenden Daten ausgeführt (Evaluationsszenario S7). Die Gegenüberstellung der Verarbeitungszeiten zeigt, dass in einem solchen Fall eine zonenbasierte Datenorganisation erhebliche Zeiteinsparungen bieten kann.

9.2.3.6 Implementierungsunterstützung

Das finale Kriterium für die Evaluation ist die Implementierungsunterstützung. Dies bezieht sich darauf, wie gut die gewählten Ansätze (zonenbasiert vs. zonenlos) die Implementierung durch Struktur, Standardisierung und Regeln unterstützt haben. Eine hohe Implementierungsunterstützung ist hier gewünscht, da sie den Implementierungsprozess beschleunigt und dafür sorgt, dass alle Umsetzungen innerhalb des Data Lake unternehmensweiten Regelungen entspricht, wie Namenskonventionen und Privatheitsregeln. Dies ist von hoher Wichtigkeit für Unternehmen, da ihr Geschäft typischerweise sehr komplex ist, was in ebenso komplexen Datenökosystemen resultiert. Eine hohe Implementierungsunterstützung erlaubt daher die Optimierung der Umsetzung von Anwendungsfällen und ermöglicht die Industrialisierung der Entwicklung von Data Lakes.

Diese Implementierungsunterstützung ist sehr hoch im zonenbasierten Ansatz, da der gesamte Data Lake nach einem vordefinierten Zonenmodell im-

plementiert ist. Für jede Zone definiert dieses Modell exakt welche Daten miteinbezogen werden, wie sie verwaltet werden, welche Nutzer*innen darauf zugreifen können und so weiter. Die Zonen beinhalten auch verschiedene Werkzeuge und Systeme für die Implementierung, welches die Standardisierung im Data Lake weiter erhöht.

Im Gegensatz dazu gibt es eine solche Leitlinie im zonenlosen Ansatz typischerweise nicht. Da jeder Anwendungsfall unabhängig von allen anderen implementiert wird, sind Standards und Strukturen normalerweise nicht besonders streng. Auch sind Werkzeuge und Systeme nur selten standardisiert. Jede Leitlinie muss durch zusätzliche Regeln definiert werden im Gegensatz zum zonenbasierten Ansatz, wo solche Regeln automatisch enthalten sind.

Zusammenfassung der Evaluation. Insgesamt zeigt diese Evaluation, dass keiner der beiden Ansätze (zonenbasiert und zonenlos) klar überlegen ist. Stattdessen haben beide unterschiedliche Vor- und Nachteile. Während der zonenlose Ansatz geringere Transformationskomplexität, schnellere Verarbeitungszeiten und weniger benötigten Speicherplatz erlaubt, bietet der zonenbasierte Ansatz Wiederverwendbarkeit von Daten und Implementierungsunterstützung. Dies wiederum bedeutet, dass in Fällen, in denen Daten nicht für verschiedene Anwendungsfälle wiederverwendet werden sollen, der zonenbasierte Ansatz kaum bis keine Vorteile bietet. Wenn Daten allerdings wiederverwendet werden, kann ein zonenbasierter Data Lake signifikante Zeitersparnisse bieten, indem der Aufwand bei der Datenaufbereitung verringert wird (60-80% der Umsetzung von Anwendungsfällen [Pre16]). So können höhere Verarbeitungszeiten und Speicherplatzverbrauch ausgeglichen werden. In Fällen, in denen schnelle Verarbeitungszeiten benötigt werden, speziell in Echtzeitverarbeitung, kann ein zonenbasierter Data Lake nicht mit einem zonenlosen Ansatz mithalten. Dies liegt vor allem an den hohen Transferzeiten zwischen Zonen. Sind allerdings Verzögerungen von 1-2 Sekunden akzeptabel, kann auch ein zonenbasierter Data Lake für Datenströme verwendet werden, um von den Vorteilen zu profitieren.

9.2.4 Evaluation der Methodik

Für die Implementierung des zonenbasierten Data Lake wurde die in Abschnitt 7.5 vorgestellte Methodik verwendet. Dieser Abschnitt fasst kurz die gemachten Erfahrungen mit dieser Methodik zusammen.

Insgesamt ermöglichten die verschiedenen Schritte der Methodik ein strukturiertes und geführtes Vorgehen bei der Definition und Implementierung einer zonenbasierten Data-Lake-Implementierung. Durch die Vorgabe von spezifizierten Anwendungsfällen im ersten Schritt wird sichergestellt, dass der Aufwand der Datenaufbereitung nur für Daten erfolgt, die auch genutzt werden (bedarfsorientiert). Bei der Nutzung eines Zonenmodells ist es von enormer Bedeutung, zu definieren, in welchem Zustand die Daten in jeder Zone verfügbar sind. Die Detaillierung dieser Zwischenschritte erleichtert die Definition der Prozesse, die für den Transfer und die Verarbeitung von Daten zwischen den Zonen benötigt werden. Der Schritt der Identifizierung vorhandener Daten ist ebenfalls von großer Bedeutung, um die Wiederverwendung von Daten und deren Transformationen zu ermöglichen. Daher ist die Methodik ein geeigneter Ansatz für die Implementierung eines zonenbasierten Data Lake.

9.3 Zusammenfassung

Die Evaluation der in dieser Arbeit erstellten Konzepte zeigt deren Anwendbarkeit, sowie die Vorteile, die sich durch ihre Verwendung ergeben. So bietet das DLAF einen strukturierten Leitfaden zur Erstellung einer vollständigen Data-Lake-Architektur, welche auf ein gegebenes Anwendungsszenario angepasst ist. Bei der Evaluation des Zonenreferenzmodells zeigt sich, wie dieses die aus der Literatur und der Praxis abgeleiteten Anforderungen erfüllen kann. In einem Vergleich mit einem zonenlosen Data Lake lag ein Data Lake mit dem Zonenreferenzmodell zwar in den Bereichen Transformationskomplexität, Verarbeitungszeit und benötigter Speicherplatz zurück, dafür bietet der zonenbasierte Ansatz deutlich höhere Wiederverwendbarkeit von Daten sowie eine bessere Implementierungsunterstützung. Auch die in Abschnitt 7.5 vorgestellte Methodik stellte sich als vorteilhaft zur Implementierung eines zonenbasierten Data Lake heraus.

Kapitel 10

Zusammenfassung und Ausblick

Neue Arten der Datenerfassung, etwa durch Sensoren, generieren eine Vielzahl heterogener Daten, die traditionelle Systeme zur Datenverwaltung herausfordern. Der Data Lake verspricht hier Abhilfe, indem er große Mengen heterogener Daten verwaltet und für flexible Analysen zur Verfügung stellt. Zur Umsetzung des Data Lake fehlt es allerdings noch an methodischen Grundlagen, die in dieser Arbeit geschaffen wurden. Die Erkenntnisse und Beiträge dieser Arbeit sind in Abschnitt 10.1 zusammengefasst. Abschließend gibt Abschnitt 10.2 einen Ausblick auf mögliche zukünftige Arbeiten.

10.1 Zusammenfassung

Die leitende Forschungsfrage dieser Arbeit richtet sich nach dem Aufbau eines Data Lake, welcher die Verwaltung und Verarbeitung von Big Data in der Praxis unterstützen kann. Diese Forschungsfrage wurde in drei Forschungsziele unterteilt (Z1 - Identifikation der angestrebten Eigenschaften eines Data Lake, Z2 - Erstellung eines Leitfadens zur Definition einer vollständigen Data-Lake-Architektur, Z3 - Erarbeitung einer internen Datenorganisation für Data Lakes),

welche wiederum durch insgesamt acht Forschungsbeiträge adressiert wurden. Die erbrachten Forschungsbeiträge sind im Folgenden zusammengefasst:

B1.1: Konsolidierung verschiedener Data-Lake-Definitionen zu einer einheitlichen Definition [GGH+19a; GGH+20a]

Auf der Grundlage einer ausführlichen Literaturrecherche wurden die zentralen Eigenschaften von Data Lakes in Kapitel 2 herausgearbeitet und gegenübergestellt. Hierbei zeigte sich, dass existierende Definitionen unterschiedliche Sichten auf Data Lakes aufweisen und sich zum Teil sogar widersprechen. Daher wurde im Rahmen dieser Arbeit eine Definition entwickelt, welche sich aus unterschiedlichen Definitionen zusammensetzt und dabei die in der Literatur am häufigsten vertretenen Eigenschaften eines Data Lake umfasst.

B1.2: Identifikation der angestrebten Eigenschaften eines Data Lake [GGH+19a; GGH+20a]

Für diesen Beitrag wurden zunächst in Kapitel 3 verschiedene Anwendungsszenarien und deren Voraussetzungen diskutiert. Hierbei zeigte sich, dass die Einsatzgebiete für Data Lakes eine große Varianz aufweisen, abhängig von den Daten, die verwaltet werden sollen, der geplanten Nutzung und anderen Kriterien. Darum wurde ein konkretes Fallbeispiel als Basis dieser Arbeit untersucht, welches das Unternehmen eines großen, globalagierenden Herstellers betrachtet. Dieses Unternehmen umfasst nicht nur eine Vielzahl heterogener Daten, die in einem unternehmensweiten Data Lake verwaltet werden sollen, sondern auch verschiedenste Nutzergruppen und Anwendungsfälle. Aus diesem Fallbeispiel und der durchgeführten Literaturrecherche wurden in Kapitel 4 die praktischen Anforderungen an einen Data Lake herausgearbeitet.

B1.3: Identifikation von Lücken durch Abgleich zwischen angestrebten Eigenschaften aus der Praxis und Stand der Literatur [GGH+19a; GGH+20a]

Ebenfalls in Kapitel 4 wurden die so abgeleiteten Anforderungen mit dem aktuellen Stand zu Data Lakes abgeglichen. So konnten fünf Forschungslücken identifiziert werden: F1 unklare Datenmodellierungsmethoden, F2) fehlende

Data-Lake-Organisation, F3) unvollständiges Governance-Konzept, F4) unvollständiges Metadatenmanagementkonzept, F5) fehlende vollständige Data-Lake-Architektur. Im Rahmen dieser Arbeit wurde ein Fokus auf F2 sowie F5 gelegt, welche durch die folgenden Forschungsbeiträge adressiert wurden.

B2.1: Identifikation der Aspekte für die Umsetzung eines Data Lake, sowie deren Abhängigkeiten [GGH+21]

Die Identifikation der notwendigen Aspekte fand durch Clustering existierender Arbeiten zu Data Lakes in Kapitel 5 statt. Insgesamt neun solcher Aspekte wurden identifiziert. Ihre Abhängigkeiten wurden diskutiert und mithilfe eines Graphen visualisiert.

B2.2: Erstellung eines Leitfadens in Form eines Data-Lake-Architekturrahmenwerks, das die Definition einer vollständigen Data-Lake-Architektur ermöglicht und Entscheidungsunterstützung bei der Auswahl geeigneter Konzepte liefert [GGH+21]

Aus den unter B2.1 identifizierten Aspekten wurde ebenfalls in Kapitel 5 das Data Lake Architecture Framework (DLAF) mit zugehöriger Methodik erstellt. Das DLAF fasst alle notwendigen Aspekte für die Erstellung eines Data Lake zusammen und assoziiert jeden Aspekt mit einer Menge möglicher Konzepte zur Umsetzung. Die zum DLAF gehörige Methodik bietet eine schrittweise Leitlinie, welche die Abhängigkeiten der Aspekte einbezieht und Anwender bei der Auswahl geeigneter Konzepte für jeden Aspekt des DLAF unterstützt.

B3.1: Identifikation von Anforderungen an die Datenorganisation innerhalb des Data Lake [GGH+20b]

Für die Adressierung von Forschungsziel Z3 (Erarbeitung einer internen Datenorganisation für Data Lakes) wurden zunächst in Kapitel 7 praktische Anforderungen an eine solche Datenorganisation abgeleitet. Hierzu wurde das zuvor vorgestellte Fallbeispiel verwendet. Anschließend wurden existierende interne Datenorganisationskonzepte mit diesen Anforderungen evaluiert.

B3.2: Erstellung eines Metamodells für Zonen für den systematischen Vergleich existierender Zonenmodelle [GGH+20b]

Bei der Betrachtung existierender Zonenmodelle zeigte sich die hohe Varianz zwischen den verschiedenen Ansätzen. Um einen systematischen Vergleich der Modelle zu ermöglichen, wurde das Metamodell für Zonen in Kapitel 6 erarbeitet. Dieses Metamodell beschreibt, welche Eigenschaften eine Zone besitzt, und wie Zonen miteinander und mit der Außenwelt interagieren. Mithilfe des Metamodells wurden anschließend existierende Zonenmodelle systematisch gegenübergestellt.

B3.3: Erarbeitung des Zonenreferenzmodells zur internen Datenorganisation eines Data Lake für die Unterstützung unterschiedlicher Nutzer*innen und Anwendungsfälle für Stapel- und Datenstromverarbeitung [GGH+20b]

Da die Evaluation existierender Datenorganisationen zeigte, dass diese für praktische Anforderungen ungeeignet sind, wurde in Kapitel 7 das Zonenreferenzmodell vorgestellt. Dieses Modell bietet einen systematischen Ansatz zur Datenorganisation in Data Lakes. Durch die sechs enthaltenen Zonen können verschiedenste Anwendungsfälle und Nutzergruppen unterstützt werden.

Eine exemplarische Data-Lake-Architektur sowie das Zonenreferenzmodell wurden in Kapitel 8 prototypisch umgesetzt, um ihre Anwendbarkeit zu demonstrieren. Hierbei wurde erneut auf das vorgestellte Fallbeispiel zurückgegriffen. Anschließend wurde in Kapitel 9 eine Evaluation der erbrachten Beiträge durchgeführt. Diese Evaluation zeigt, dass das DLAF alle notwendigen Aspekte umfasst und darüber hinaus für die Bewertung existierender Data-Lake-Architekturen verwendet werden kann. Das Zonenreferenzmodell wurde zur Evaluation mit einem zonenlosen Ansatz verglichen. Hierbei zeigte sich, dass der zonenbasierte Ansatz mehr Speicherplatz und Verarbeitungszeit benötigt als der zonenlose Ansatz. Allerdings ist die Wiederverwendbarkeit von Daten und die Implementierungsunterstützung im zonenbasierten Ansatz deutlich höher. So kann zwischen 60-80% Implementierungsaufwand durch Datenaufbereitung eingespart werden.

Zusammenfassend zeigt sich somit, dass die in dieser Arbeit erbrachten Beiträge die Forschungsziele erfüllen und somit eine methodische Grundlage für den Aufbau von Data Lakes in der Praxis bieten.

10.2 Ausblick

Zwar wurden zwei der in Kapitel 4 vorgestellten Forschungslücken in dieser Arbeit geschlossen, jedoch verbleiben drei Lücken unadressiert. Diese Lücken betreffen die Datenmodellierung in Data Lakes, das Governance-Konzept und das Metadatenmanagement. Diese Lücken müssen in zukünftigen Arbeiten adressiert werden. Darüber hinaus verbleiben folgende Punkte für weiterführende Betrachtung:

Umsetzung einer vollständigen Data-Lake-Architektur

Während in der konzeptionellen Betrachtung eine vollständige Data-Lake-Architektur entworfen wurde, wurden in der prototypischen Implementierung nur ausgewählte Aspekte umgesetzt. In zukünftigen Arbeiten sollten darum die fehlenden Aspekte ebenfalls realisiert werden, um so eine vollständige Implementierung einer Data-Lake-Architektur zu erreichen.

Betrachtung unstrukturierter Daten im Zonenreferenzmodell

Aus Platzgründen fokussierte sich die Implementierung des Zonenreferenzmodells in dieser Arbeit auf strukturierte und semi-strukturierte Daten. Data Lakes umfassen jedoch auch unstrukturierte Daten, welche ebenfalls von den Vorteilen des Zonenreferenzmodells profitieren können. Hierbei gilt es zu erforschen, wie unstrukturierte Daten schrittweise aufbereitet und mit anderen Daten integriert werden können. Etwa können Videos in einzelne Frames unterteilt werden oder Bilder mit Tags versehen werden.

Verbesserung der Echtzeitnähe des Zonenreferenzmodells

Die Evaluation in Kapitel 9 hat gezeigt, dass es bei der Echtzeitverarbeitung in zonenbasierten Data-Lake-Implementierungen zu erheblichen Verzögerungen kommt. Allerdings bietet die Nutzung des Zonenreferenzmodells auch für

die Datenstromverarbeitung erhebliche Vorteile. Zukünftige Arbeiten sollten sich darum auf die Reduktion des Mehraufwands sowie der Verzögerung im zonenbasierten Ansatz fokussieren. Dies kann durch eine schnellere Datenübertragung zwischen den Zonen erreicht werden.

Eigene Veröffentlichungen

[1] C. Giebler und C. Stach. „Datenschutzmechanismen für Gesundheitsspiele am Beispiel von Secure Candy Castle“. In: *Proceedings der 17. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW'17)*. 2017.

[2] C. Giebler, C. Stach, H. Schwarz und B. Mitschang. „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 und Technology Publications, 2018, Seiten 294–301. ISBN: 978-989-758-318-6. DOI: 10.5220/0006861802940301. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006861802940301.

[3] C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „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. DOI: 10.1007/978-3-030-27520-4_13.

[4] C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned“. In: *Proceedings of the 38th Conference on Conceptual Modeling (ER 2019)*. 2019.

[5] C. Stach, C. Giebler und S. Schmidt. „Zuverlässige Verspätungsvorhersagen mithilfe von TAROT“. In: *Proceedings der 18. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW 2019)*. 2019.

[6] R. Eichler, C. Giebler, C. Gröger, H. Schwarz und B. Mitschang. „HANDLE - A Generic Metadata Model for Data Lakes“. In: *Proceedings of the 22nd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK2020)*. 2020.

[7] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz und B. Mitschang. „Data Lakes auf den Grund gegangen“. In: *Datenbank-Spektrum* 20.1 (2020), Seiten 57–69. ISSN: 1618-2162. DOI: 10.1007/s13222-020-00332-0. URL: http://link.springer.com/10.1007/s13222-020-00332-0.

[8] C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „A Zone Reference Model for Enterprise-Grade Data Lake Management“. In: *Proceedings of the 24th IEEE Enterprise Computing Conference (EDOC 2020)*. 2020.

[9] C. Stach, J. Bräcker, R. Eichler, C. Giebler und C. Gritti. „How to Provide High-Utility Time Series Data in a Privacy-Aware Manner: A VAULT to Manage Time Series Data“. In: *International Journal on Advances in Security* 13.3 & 4 (2020). Herausgegeben von H.-J. Hof und B. Gersbeck-Schierholz, Seiten 88–108. ISSN: 1942-2636. DOI: 10.1007/s11623-019-1201-8. URL: http://www.iariajournals.org/security/.

[10] C. Stach, C. Giebler, M. Wagner, C. Weber und B. Mitschang. „AMNESIA: A Technical Solution towards GDPR-compliant Machine Learning“. In: *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*. 2020, Seiten 21–32. ISBN: 978-989-758-399-5. DOI: 10.5220/0008916700210032. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0008916700210032.

[11] C. Stach, J. Bräcker, R. Eichler, C. Giebler und B. Mitschang. „Demand-Driven Data Provisioning in Data Lakes: BARENTS — A Tailorable Data Preparation Zone“. In: *Proceedings of the 23rd International Conference on Information Integration and Web Intelligence*. Herausgegeben von M. Indrawan-Santiago, E. Pardede, I. L. Salvadori, M. Steinbauer, I. Khalil und G. Kotsis. iiWAS '21. **iiWAS 2021 Best Paper Award**. Linz: ACM, Nov. 2021, Seiten 191–202. ISBN: 978-1-4503-9556-4. DOI: 10.29007/1sbn.

[12] R. Eichler, C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „Enterprise-Wide Metadata Management: An Industry Case on the Cur-

rent State and Challenges“. In: *Proceedings of the 24th International Conference on Business Information Systems (BIS)*. 2021.

[13] R. Eichler, C. Giebler, C. Gröger, H. Schwarz und B. Mitschang. „Modeling metadata in data lakes—A generic model“. In: *Data & Knowledge Engineering* (2021).

[14] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz und B. Mitschang. „The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture“. In: *Proceedings der 19. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW 2021)*. 2021.

Betreute Arbeiten

[1] N. Keppler. „Zugriffskontrollmodel für Data Lakes“. Masterarbeit. Universität Stuttgart, Nov. 2018.

[2] S. Schmidt. „Realisierung von umfassenden Analysetechniken in einer hybriden Datenverarbeitungsarchitektur“. Bachelorarbeit. Universität Stuttgart, Nov. 2018.

[3] F. Ebinger. „Personenbezogene Daten im Data Lake“. Masterarbeit. Universität Stuttgart, Dez. 2018.

[4] G. Nataraj. „Integration of Heterogeneous Data in the Data Vault Model“. Masterarbeit. Universität Stuttgart, Jan. 2019.

[5] R. K. Eichler. „Metadata Management in the Data Lake Architecture“. Masterarbeit. Universität Stuttgart, Mai 2019.

[6] I. Kutger. „Visualisierung der Auswirkungen von Privacy Techniken auf Machine Learning Ergebnisse“. Prozessanalyse. Universität Stuttgart, Jan. 2020.

[7] M. Winckler. „Umsetzung anwendungsspezifischer ETL-Prozesse im Data Lake“. Bachelorarbeit. Universität Stuttgart, Feb. 2020.

[8] F. Geiger. „Realisierung des Zonenreferenzmodells auf Datenströmen“. Bachelorarbeit. Universität Stuttgart, Nov. 2020.

[9] H. Düsseldorf. „Entwicklung einer Datenbereitstellungsplattform für datenintensive IoT-Anwendungen“. Bachelorarbeit. Universität Stuttgart, Dez. 2020.

[10] M. Altvater. „Anforderungen von Data-Science-Anwendungsfällen im Zonenreferenzmodell“. Bachelorarbeit. Universität Stuttgart, Apr. 2021.

Literaturverzeichnis

[AC15] A. A. Avery und K. Cheek. „Analytics Governance: Towards a Definition and Framework“. In: *Proceedings of the 21st Americas Conference on Information Systems (AMCIS 2015)*. 2015 (Zitiert auf S. 30).

[AJO+13] P. Atzeni, C. S. Jensen, G. Orsi, S. Ram, L. Tanca und R. Torlone. „The relational model is dead, SQL is dead, and I don't feel so good myself“. In: *ACM SIGMOD Record* 42.1 (Mai 2013), Seite 64. ISSN: 01635808. DOI: 10.1145/2503792.2503808. URL: http://dl.acm.org/citation.cfm?doid=2503792.2503808 (Zitiert auf S. 33).

[BC13] F. Bugiotti und L. Cabibbo. „A Comparison of Data Models and APIs of NoSQL Datastores“. In: *Proceedings of the 21st Italian Symposium on Advanced Database Systems (SEBD 2013)*. 2013, Seiten 63–74. URL: http://www.bugiotti.it/downloads/publications/noamSEBD13.pdf (Zitiert auf S. 33).

[BCJ+14] C. Ballard, C. Compert, T. Jesionowski, I. Milman, B. Plants, B. Rosen und H. Smith. *Information Governance Principles and Practices for a Big Data Landscape*. Herausgegeben von W. Wallace. IBM, 2014 (Zitiert auf S. 63).

[Bos09] R. Bose. „Advanced analytics: opportunities and challenges“. In: *Industrial Management & Data Systems (IDMS)* 109.2 (März 2009), Seiten 155–172. ISSN: 0263-5577. DOI: 10.1108/02635570910930073. URL: https://www.emeraldinsight.com/doi/10.1108/02635570910930073 (Zitiert auf S. 18).

[Cao17] L. Cao. „Data Science“. In: *ACM Computing Surveys* 50.3 (Juni 2017), Seiten 1–42. ISSN: 03600300. DOI: 10.1145/3076253. URL: http://dl.acm.org/citation.cfm?doid=3101309.3076253 (Zitiert auf S. 18).

[Cat11] R. Cattell. „Scalable SQL and NoSQL data stores“. In: *ACM SIGMOD Record* 39.4 (Mai 2011), Seite 12. ISSN: 01635808. DOI: 10.1145/1978915.1978919. URL: http://portal.acm.org/citation.cfm?doid=1978915.1978919 (Zitiert auf S. 33).

[CCS12] H. Chen, R. H. L. Chiang und V. C. Storey. „Business Intelligence and Analytics: From Big Data to Big Impact“. In: *MIS Quarterly* 36.4 (2012), Seiten 1165–1188 (Zitiert auf S. 46).

[CJL+15] M. Chessell, N. L. Jones, J. Limburn, D. Radley und K. Shan. *Designing and Operating a Data Reservoir*. Herausgegeben von IBM. IBM, 2015 (Zitiert auf S. 51, 62, 63, 72).

[CMB+16] L. H. Childs, S. Mamlouk, J. Brandt, C. Sers und U. Leser. „SoFIA: a data integration framework for annotating high-throughput datasets“. In: *Bioinformatics* 32.17 (Sep. 2016), Seiten 2590–2597. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw302. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw302 (Zitiert auf S. 71).

[CSN+14] M. Chessell, F. Scheepers, N. Nguyen, R. van Kessel und R. van der Starre. *Governing and Managing Big Data for Analytics and Decision Makers*. Herausgegeben von IBM. IBM, 2014 (Zitiert auf S. 30, 31, 63, 134, 156).

[CY15] R. Casado und M. Younas. „Emerging trends and technologies in big data processing“. In: *Concurrency and Computation: Practice and Experience* 27.8 (Juni 2015), Seiten 2078–2091. ISSN: 15320626. DOI: 10.1002/cpe.3398. URL: http://doi.wiley.com/10.1002/cpe.3398 (Zitiert auf S. 35, 36, 61).

[DAM17] DAMA. *DAMA-DMBOK: Data Management Body of Knowledge*. 2nd. Technics Publications, 2017 (Zitiert auf S. 62, 134).

[Deh19] Z. Dehghani. *How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh*. 2019. URL: https://martinfowler.com/articles/data-monolith-to-mesh.html (besucht am 27. 05. 2019) (Zitiert auf S. 62, 67, 68).

[DG08] J. Dean und S. Ghemawat. „MapReduce“. In: *Communications of the ACM* 51.1 (Jan. 2008), Seite 107. ISSN: 00010782. DOI: 10.1145/1327452.1327492. URL: http://portal.acm.org/citation.cfm?doid=1327452.1327492 (Zitiert auf S. 35, 42).

[Dix10] J. Dixon. *Pentaho, Hadoop, and Data Lakes*. 2010. URL: https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/ (besucht am 01. 03. 2018) (Zitiert auf S. 17, 18, 26, 29, 107, 111).

[Dix14] J. Dixon. *Data Lakes Revisited*. 2014. URL: https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/ (besucht am 01. 03. 2018) (Zitiert auf S. 28, 29).

[EGG+20] R. Eichler, C. Giebler, C. Gröger, H. Schwarz und B. Mitschang. „HANDLE - A Generic Metadata Model for Data Lakes". In: *Proceedings of the 22nd International Conference on Big Data Analytics and Knowledge Discovery (DaWaK2020)*. 2020 (Zitiert auf S. 58, 63, 71, 110, 123, 134, 145, 156).

[Fan15] H. Fang. „Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem". In: *Proceedings of the 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER 2015)*. IEEE, Juni 2015, Seiten 820–824. ISBN: 978-1-4799-8728-3. DOI: 10.1109/CYBER.2015.7288049. URL: http://ieeexplore.ieee.org/document/7288049/ (Zitiert auf S. 27, 28, 30).

[FBT+12] L. J. Fülöp, Á. Beszédes, G. Tóth, H. Demeter, L. Vidács und L. Farkas. „Predictive complex event processing". In: *Proceedings of the Fifth Balkan Conference in Informatics (BCI '12)*. New York, New York, USA: ACM Press, 2012, Seite 26. ISBN: 9781450312400. DOI: 10.1145/2371316.2371323. URL: http://dl.acm.org/citation.cfm?doid=2371316.2371323 (Zitiert auf S. 128).

[FHM05] M. Franklin, A. Halevy und D. Maier. „From databases to dataspaces". In: *ACM SIGMOD Record* 34.4 (Dez. 2005), Seiten 27–33. ISSN: 01635808. DOI: 10.1145/1107499.1107502. URL: http://portal.acm.org/citation.cfm?doid=1107499.1107502 (Zitiert auf S. 26).

[Gar14] Gartner Inc. *Gartner Says Beware of the Data Lake Fallacy*. Online. 2014. URL: https://www.gartner.com/en/newsroom/press-releases/2014-07-28-gartner-says-beware-of-the-data-lake-fallacy (Zitiert auf S. 30).

[GGH+19a] C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „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. DOI: 10.1007/978-3-030-27520-4_13 (Zitiert auf S. 25, 176).

[GGH+19b] C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned“. In: *Proceedings of the 38th Conference on Conceptual Modeling (ER 2019)*. 2019 (Zitiert auf S. 70, 79, 133).

[GGH+20a] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz und B. Mitschang. „Data Lakes auf den Grund gegangen“. In: *Datenbank-Spektrum* 20.1 (März 2020), Seiten 57–69. ISSN: 1618-2162. DOI: 10.1007/s13222-020-00332-0. URL: http://link.springer.com/10.1007/s13222-020-00332-0 (Zitiert auf S. 25, 32, 41, 44, 49, 52, 176).

[GGH+20b] C. Giebler, C. Gröger, E. Hoos, H. Schwarz und B. Mitschang. „A Zone Reference Model for Enterprise-Grade Data Lake Management“. In: *Proceedings of the 24th IEEE Enterprise Computing Conference (EDOC 2020)*. 2020 (Zitiert auf S. 62, 63, 67, 69, 72, 76, 78, 80, 88, 91, 92, 94, 95, 98, 100, 101, 124, 152, 177, 178).

[GGH+21] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz und B. Mitschang. „The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture“. In: *Proceedings der 19. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW 2021)*. 2021 (Zitiert auf S. 55, 57, 59, 64, 66, 68, 124, 129, 131, 132, 152, 153, 177).

[GH19] C. Gröger und E. Hoos. „Ganzheitliches Metadatenmanagement im Data Lake: Anforderungen, IT-Werkzeuge und Herausforderungen in der Praxis“. In: *Proceedings der 18. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*. 2019 (Zitiert auf S. 29).

[Gor16] A. Gorelik. *The Enterprise Big Data Lake*. Herausgegeben von T. McGovern. O’Reilly Media, Inc., 2016 (Zitiert auf S. 19, 69, 75, 77, 78, 81, 82, 87, 92–96, 119).

[Grö18] C. Gröger. „Building an Industry 4.0 Analytics Platform“. In: *Datenbank-Spektrum* 18.1 (März 2018), Seiten 5–14. ISSN: 1618-2162. DOI: 10.1007/s13222-018-0273-1. URL: https://doi.org/10.1007/s13222-018-0273-1%20http://link.springer.com/10.1007/s13222-018-0273-1 (Zitiert auf S. 44).

[GSM14] C. Gröger, H. Schwarz und B. Mitschang. „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, Sep. 2014, Seiten 210–217. ISBN: 978-1-4799-5470-4. DOI: 10.1109/EDOC.2014.36. URL: http://ieeexplore.ieee.org/document/6972069/ (Zitiert auf S. 70, 103, 133).

[GSSM18] C. Giebler, C. Stach, H. Schwarz und B. Mitschang. „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 und Technology Publications, 2018, Seiten 294–301. ISBN: 978-989-758-318-6. DOI: 10.5220/0006861802940301. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006861802940301 (Zitiert auf S. 38, 39, 61, 67, 130).

[GWFR17] F. Gessert, W. Wingerath, S. Friedrich und N. Ritter. „NoSQL database systems: a survey and decision guidance“. In: *Computer Science - Research and Development* 32.3-4 (Juli 2017), Seiten 353–365. ISSN: 1865-2034. DOI: 10.1007/s00450-016-0334-3. URL: http://link.springer.com/10.1007/s00450-016-0334-3 (Zitiert auf S. 69).

[Hag16] J. Hagerty. *2017 Planning Guide for Data and Analytics*. Technischer Bericht. 2016 (Zitiert auf S. 41, 47, 125).

[HAR16] V. Herrero, A. Abelló und O. Romero. „NOSQL Design for Analytical Workloads: Variability Matters“. In: *Proceedings of the 35th International Conference on Conceptual Modeling (ER 2016)*. 2016, Seiten 50–64. DOI: 10.1007/978-3-319-46397-1_4. URL: http://link.springer.com/10.1007/978-3-319-46397-1_4 (Zitiert auf S. 33).

[HFM06] A. Halevy, M. Franklin und D. Maier. „Principles of dataspace systems“. In: *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '06)*. New York, New York, USA: ACM Press, 2006, Seiten 1–9. ISBN: 1595933182. DOI: 10.1145/1142351.1142352. URL: http://portal.acm.org/citation.cfm?doid=1142351.1142352 (Zitiert auf S. 26).

[HGQ16] R. Hai, S. Geisler und C. Quix. „Constance: An Intelligent Data Lake System“. In: *Proceedings of the 2016 International Conference on Management of Data (SIGMOD'16)*. 2016, Seiten 2097–2100 (Zitiert auf S. 58, 63, 123).

[HKN+16a] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy und S. E. Whang. „Goods: Organizing Google's Datasets". In: *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. ACM Press, 2016, Seiten 795–806. ISBN: 9781450335317. DOI: 10.1145/2882903.2903730. URL: http://dl.acm.org/citation.cfm?doid=2882903.2903730 (Zitiert auf S. 42).

[HKN+16b] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy und S. E. Whang. „Managing Google's data lake: an overview of the Goods system". In: *IEEE Data Engineering Bulletin* 39 (2016), Seiten 5–14 (Zitiert auf S. 42).

[Hou17] P. Houle. *Data Lakes, Data Ponds, and Data Droplets*. Online. 2017. URL: http://ontology2.com/the-book/data-lakes-ponds-and-droplets.html (Zitiert auf S. 58, 61, 70).

[Inm16] B. Inmon. *Data Lake Architecture - Designing the Data Lake and avoiding the Garbage Dump*. Herausgegeben von R. A. Peters. Technics Publications, 2016 (Zitiert auf S. 19, 57, 58, 62, 67, 68, 72, 75, 76, 92).

[JQ17] M. Jarke und C. Quix. „On Warehouses, Lakes, and Spaces: The Changing Role of Conceptual Modeling for Data Integration". In: *Conceptual Modeling Perspectives*. Herausgegeben von J. Cabot, C. Gómez, O. Pastor, M. R. Sancho und E. Teniente. Springer International Publishing AG, 2017. Kapitel 16, Seiten 231–245 (Zitiert auf S. 26, 57, 58, 62).

[KBLK10] M. T. Koch, H. Baars, H. Lasi und H.-G. Kemper. „Manufacturing Execution Systems and Business Intelligence for Production Environments". In: *Proceedings of the 16th Americas Conference on Information Systems (AMCIS 2010)*. 2010 (Zitiert auf S. 89).

[KR13] K. Kaur und R. Rani. „Modeling and querying data in NoSQL databases". In: *Proceedings of the 2013 IEEE International Conference on Big Data*. IEEE, Okt. 2013, Seiten 1–7. ISBN: 978-1-4799-1293-3. DOI: 10.1109/BigData.2013.6691765. URL: http://ieeexplore.ieee.org/document/6691765/ (Zitiert auf S. 33).

[Kre14] J. Kreps. *Questioning the Lambda Architecture*. Online. 2014. URL: https://www.oreilly.com/ideas/questioning-the-lambda-architecture (Zitiert auf S. 37, 38).

[LC16] D. Larson und V. Chang. „A review and future direction of agile, business intelligence, analytics and data science“. In: *International Journal of Information Management* 36.5 (Okt. 2016), Seiten 700–710. ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2016.04.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S026840121630233X (Zitiert auf S. 46).

[Lin12] D. Linstedt. *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*. Herausgegeben von K. Graziano. 1. Auflage. 2012 (Zitiert auf S. 57, 61, 70, 103).

[LKY14] J. Lee, H.-A. Kao und S. Yang. „Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment“. In: *Procedia CIRP* 16 (2014), Seiten 3–8. ISSN: 22128271. DOI: 10.1016/j.procir.2014.02.001. URL: http://linkinghub.elsevier.com/retrieve/pii/S2212827114000857%20https://linkinghub.elsevier.com/retrieve/pii/S2212827114000857 (Zitiert auf S. 18).

[LO15] D. Linstedt und M. Olschimke. *Building a Scalable Data Warehouse with Data Vault 2.0*. Elsevier LTD, 2015 (Zitiert auf S. 133).

[Loc16] M. Lock. *Maximizing your Data Lake with a Cloud or Hybrid Approach*. Technischer Bericht. 2016 (Zitiert auf S. 29, 61).

[Los09] D. Loshin. *Master Data Management*. Elsevier Inc., 2009 (Zitiert auf S. 45).

[Mad15] M. Madsen. *How to Build an Enterprise Data Lake: Important Considerations before Jumping In*. Technischer Bericht. Third Nature Inc., 2015 (Zitiert auf S. 17, 19, 27–29, 75, 77, 79, 81, 83, 87, 92–96).

[Mat17] C. Mathis. „Data Lakes“. In: *Datenbank-Spektrum* 17.3 (Nov. 2017), Seiten 289–293. ISSN: 1618-2162. DOI: 10.1007/s13222-017-0272-7. URL: http://link.springer.com/10.1007/s13222-017-0272-7 (Zitiert auf S. 18, 29, 31).

[MB17] T. Marschall und H. Baars. „Pi-Architektur“. In: *BI-Spektrum: Online Special Self-Service Data Preperation* (2017). URL: http://www.sigs.de/publications/bi/2017/SelfService/marschall_baars_BIS_OTS_SelfService_2017.pdf (Zitiert auf S. 28, 39).

[MBG+17] M. A. Martínez-Prieto, A. Bregon, I. García-Miranda, P. C. Álvarez-Esteban, F. Díaz und D. Scarlatti. „Integrating Flight-related Information into a (Big) data lake“. In: *Proceedings of the 36th IEEE/AIAA Digital Avionics Systems Conference (DASC)*. IEEE, 2017. ISBN: 978-1-5386-0365-9. DOI: 10.1109/DASC.2017.8102023. URL: http://ieeexplore.ieee.org/document/8102023/ (Zitiert auf S. 42, 56, 58, 61, 107, 109, 111, 119, 123, 152–154).

[ML16] C. Madera und A. Laurent. „The Next Information Architecture Evolution: The Data Lake Wave“. In: *Proceedings of the 8th International Conference on Management of Digital EcoSystems (MEDES)*. New York, New York, USA: ACM Press, 2016, Seiten 174–180. ISBN: 9781450342674. DOI: 10.1145/3012071.3012077. URL: http://dl.acm.org/citation.cfm?doid=3012071.3012077 (Zitiert auf S. 27, 29, 30, 32).

[MM18] A. A. Munshi und Y. A.-R. I. Mohamed. „Data Lake Lambda Architecture for Smart Grids Big Data Analytics“. In: *IEEE Access* 6 (2018), Seiten 40463–40471. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2858256. URL: https://ieeexplore.ieee.org/document/8417407/ (Zitiert auf S. 43, 56, 58, 111, 123, 152, 153, 155).

[MSLV13] A. Muschalle, F. Stahl, A. Löser und G. Vossen. „Pricing Approaches for Data Markets“. In: *International Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE 2012)*. 2013, Seiten 129–144. DOI: 10.1007/978-3-642-39872-8_10. URL: http://link.springer.com/10.1007/978-3-642-39872-8_10 (Zitiert auf S. 72).

[MW15] N. Marz und J. Warren. *Big Data - Principles and best practices of scalable real-time data systems*. Herausgegeben von Manning Publications Co. Manning Publications Co., 2015 (Zitiert auf S. 36, 37, 43, 61, 67, 155).

[NRD18] I. Nogueira, M. Romdhane und J. Darmont. „Modeling Data Lake Metadata with a Data Vault“. In: *Proceedings of the 22nd International Database Engineering Applications Symposium (IDEAS 2018)*. 2018 (Zitiert auf S. 58).

[OLe14] D. E. O'Leary. „Embedding AI and Crowdsourcing in the Big Data Lake“. In: *IEEE Intelligent Systems* 29.5 (Sep. 2014), Seiten 70–73. ISSN: 1541-1672. DOI: 10.1109/MIS.2014.82. URL: http://ieeexplore.ieee.org/document/6949519/ (Zitiert auf S. 27, 28, 44).

[Pre16] G. Press. *Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says*. 2016. URL: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/%7B%5C#%7Ddd2893a6f637 (besucht am 12. 10. 2020) (Zitiert auf S. 75, 171, 173).

[PWD17] P. Patel, G. Wood und A. Diaz. „Data Lake Governance Best Practices". In: *The DZone Guide to Big Data - Data Science & Advanced Analytics* 4 (2017), Seiten 6–7 (Zitiert auf S. 19).

[Rus17] P. Russom. „Data Lakes - Purposes, Practices, Patterns, and Platforms". In: *TDWI* Q1 (2017) (Zitiert auf S. 50).

[RWE15] I. Robinson, J. Webber und E. Eifrem. *Graph Databases - New Opportunites for Connected Data*. 2nd. O'Reilly Media, Inc., 2015. ISBN: 9781491930892 (Zitiert auf S. 34).

[RZ19] F. Ravat und Y. Zhao. „Data Lakes: Trends and Perspectives". In: *Proceedings of the 30th International Conference on Database and Expert Systems Applications (DEXA 2019)*. 2019, Seiten 304–313. DOI: 10.1007/978-3-030-27615-7_23. URL: http://link.springer.com/10.1007/978-3-030-27615-7%7B%5C_%7D23 (Zitiert auf S. 19, 22, 30, 31, 57, 58, 75, 77, 81, 84, 87, 92–96, 107, 118, 119).

[SD21] P. Sawadogo und J. Darmont. „On data lake architectures and metadata management". In: *Journal of Intelligent Information Systems* 56.1 (Feb. 2021), Seiten 97–120. ISSN: 0925-9902. DOI: 10.1007/s10844-020-00608-7. URL: http://link.springer.com/10.1007/s10844-020-00608-7 (Zitiert auf S. 19, 58).

[SF13] P. J. Sadalage und M. Fowler. *NoSQL Distilled - A Brief Guide to the Emerging World of Polyglot Persistence*. Pearson Education, Inc., 2013. ISBN: 978-0-321-82662-6 (Zitiert auf S. 33, 34).

[SGS19] C. Stach, C. Giebler und S. Schmidt. „Zuverlässige Verspätungsvorhersagen mithilfe von TAROT". In: *Proceedings der 18. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW 2019)*. 2019 (Zitiert auf S. 39).

[SGW+20] C. Stach, C. Giebler, M. Wagner, C. Weber und B. Mitschang. „AMNESIA: A Technical Solution towards GDPR-compliant Machine Learning". In: *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*. 2020, Seiten 21–32. ISBN:

978-989-758-399-5. DOI: 10.5220/0008916700210032. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0008916700210032 (Zitiert auf S. 41, 63).

[Sha18] B. Sharma. *Architecting Data Lakes - Data Management Architectures for Advanced Business Use Cases*. 2. Auflage. O'Reilly Media, Inc., 2018 (Zitiert auf S. 19, 22, 30, 56–58, 62, 75, 77, 78, 81, 85–87, 92–96, 98, 119, 121).

[SHJ17] K. Shin, C. Hwang und H. Jung. „NoSQL Database Design Using UML Conceptual Data Model Based on Peter Chen's Framework". In: *International Journal of Applied Engineering Research* 12.5 (2017), Seiten 632–636 (Zitiert auf S. 34).

[SJWW16] D. Schnider, C. Jordan, P. Welker und J. Wehner. *Data Warehouse Blueprints - Business Intelligence in der Praxis*. 1. Auflage. Carl Hanser Verlag, 2016. ISBN: 978-3-446-45075-2 (Zitiert auf S. 77).

[SKW17] J. Stefanowski, K. Krawiec und R. Wrembel. „Exploring complex and big data". In: *International Journal of Applied Mathematics and Computer Science* 27.4 (Dez. 2017), Seiten 669–679. ISSN: 2083-8492. DOI: 10.1515/amcs-2017-0046. URL: http://www.degruyter.com/view/j/amcs.2017.27.issue-4/amcs-2017-0046/amcs-2017-0046.xml (Zitiert auf S. 47).

[SM14] B. Stein und A. Morrison. „The enterprise data lake: Better integration and deeper analytics". In: *Technology Forecast: Rethinking integration* 1 (2014) (Zitiert auf S. 28).

[SME14] D. Schnider, A. Martino und M. Eschermann. „Comparison of Data Modeling Methods for a Core Data Warehouse". In: *Trivadis* (2014) (Zitiert auf S. 45).

[SSPW14] K. Sandkuhl, J. Stirna, A. Persson und M. Wißotzki. „Frameworks and Reference Architectures". In: *Enterprise Modeling - Tackling Business Challenges with the 4EM Method*. 2014, Seiten 273–289. DOI: 10.1007/978-3-662-43725-4_15. URL: http://link.springer.com/10.1007/978-3-662-43725-4%7B%5C_%7D15 (Zitiert auf S. 40).

[Sti14] P. Stiglich. „Data Modeling in the Age of Big Data". In: *Business Intelligence Journal* 19.4 (2014), Seiten 17–22 (Zitiert auf S. 33, 79).

[SZ92] J. F. Sowa und J. A. Zachman. „Extending and formalizing the framework for information systems architecture“. In: *IBM Systems Journal* 31.3 (1992), Seiten 590–616. ISSN: 0018-8670. DOI: 10.1147/sj.313.0590. URL: http://ieeexplore.ieee.org/document/5387433/ (Zitiert auf S. 40).

[TC05] J. J. Thomas und K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization und Analytics Center, 2005 (Zitiert auf S. 127).

[TD16] P. Tyagi und H. Demirkan. „Data Lakes: The biggest big data challenges“. In: *Analytics* 9.6 (Nov. 2016), Seiten 56–63 (Zitiert auf S. 27–29).

[Top16] A. Topchyan. „Enabling Data Driven Projects for a Modern Enterprise“. In: *Proceedings of the Institute for System Programming of the RAS* 28.3 (2016), Seiten 209–230. ISSN: 20798156. DOI: 10.15514/ISPRAS-2016-28(3)-13. URL: http://www.ispras.ru/en/proceedings/isp_28_2016_3/isp_28_2016_3_209/ (Zitiert auf S. 31).

[TSRC15] I. Terrizzano, P. Schwarz, M. Roth und J. E. Colino. „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 (Zitiert auf S. 18, 29, 41, 43, 47, 63, 72, 152).

[VBH+15] H. Vera, W. Boaventura, M. Holanda, V. Guimarães und F. Hondo. „Data Modeling for NoSQL Document-Oriented Databases“. In: *Proceedings of the 2nd Annual International Symposium on Information Management and Big Data (SIMBig 2015)*. 2015 (Zitiert auf S. 33).

[WDR06] E. Wu, Y. Diao und S. Rizvi. „High-performance complex event processing over streams“. In: *Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD '06)*. New York, New York, USA: ACM Press, 2006, Seite 407. ISBN: 1595934340. DOI: 10.1145/1142473.1142520. URL: http://portal.acm.org/citation.cfm?doid=1142473.1142520 (Zitiert auf S. 127).

[Wel20] D. Wells. „An Architect's View of the Data Lakehouse: Perplexity and Perspective“. In: *Eckerson Group* (2020). URL: https://www.eckerson.com/articles/an-architect-s-view-of-the-data-lakehouse-perplexity-and-perspective (Zitiert auf S. 28).

[WSL18] S. Wandelt, X. Sun und U. Leser. „Column-wise compression of open relational data“. In: *Information Sciences* 457-458 (Aug. 2018), Seiten 48–61. ISSN: 00200255. DOI: 10.1016/j.ins.2018.04.074. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020025516315456 (Zitiert auf S. 103).

[Zac87] J. A. Zachman. „A framework for information systems architecture“. In: *IBM Systems Journal* 26.3 (1987), Seiten 276–292. ISSN: 0018-8670. DOI: 10.1147/sj.263.0276. URL: http://ieeexplore.ieee.org/document/5387671/ (Zitiert auf S. 39, 40, 56).

[ZDB+15] P. Zikopoulos, D. DeRoos, C. Bienko, R. Buglio und M. Andrews. *Big Data Beyond the Hype*. 1. Auflage. McGraw-Hill Education, 2015. ISBN: 978-0-07-184466-6 (Zitiert auf S. 19, 28, 43, 61, 75, 77, 81, 87, 92–95, 97, 107–109, 111, 116, 119, 123).

Alle URLs wurden zuletzt am 02. 08. 2021 geprüft.

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis