Technology
Arts Sciences
TH Köln

Fachhochschule
Dortmund
University of Applied Sciences and Arts

Verbundstudiengang Wirtschaftsinformatik

Abschlussarbeit zur Erlangung Mastergrades

Master of Science

in der Fachrichtung Informatik

Machbarkeitsanalyse über den Aufbau eines Enterprise Data Warehouse auf Basis von Apache Hadoop

Erstprüfer:	Frau Prof. Dr. Faeskorn-Woyke
Zweitprüfer	Herr Dr. Markus Arns
vorgelegt am:	20. Mai 2016
von cand.	Matthias Kappelhoff
aus	Meineckestr. 22
	40474 Düsseldorf
Tel.:	+49 (0) 176 93 144 55 88
E-Mail:	kappelhoffm@gmail.com
Matr.-Nr. (DO):	7087239
Matr.-Nr. (Köln):	11094439

Abstract

Die vorliegende Masterthesis liefert eine Einführung in die Themen Data Warehouse, Big Data und Apache Hadoop. Sie präsentiert grundlegende Data-Warehouse-Konzepte und überprüft, inwieweit diese mit dem Apache Hadoop Software Framework zu realisieren sind. Hierbei wird sowohl eine technische Überprüfung vorgenommen als auch verschiedene Szenarien veranschaulicht, wie Hadoop inhaltlich sinnvoll in bestehende Systeme integriert werden kann. Inhaltlich wird über das Thema Big Data an die Notwendigkeit einer solchen Überprüfung herangeführt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs- u. Symbolverzeichnis

ETL	Extract, Transform, Load
IuK	Information und Kommunikation
AIS	Analytisches Informationssystem
DWH	Data Warehouse
OLTP	Online Transaction Processing
OLAP	Online Analytical Processing
YARN	Yet Another Resource Negotiator
HDFS	Hadoop Distributed File System
MR	MapReduce
UDF	User Defined Function
ROLAP	Relationales Online Transaction Processing
MOLAP	Multidimensionales Online Transaction Processing
IT	Informationstechnik

1 Einleitung

1.1 Problemstellung

Als ich seinerzeit ein Thema für meine Masterthesis suchte, war mir relativ schnell klar, dass ich ein Thema bearbeiten wollte, dass mich zum einen interessiert, mir aber zum anderen auch ein neues Wissensasset für meine tägliche Arbeit liefert. Als Business Intelligence Consultant beschäftige ich mich tagein, tagaus mit der Auswertung der Daten meiner Kunden. Die Tätigkeit umfasst auf der einen Seite stark betriebswirtschaftlich geprägte Aufgaben, wie das Analysieren von Geschäftsprozessen und das Definieren von Kennzahlen. Auf der anderen Seite finden sich jedoch auch viele Aufgaben, die stark technisch geprägt sind.

- Welche Daten werden benötigt, um die Kennzahlen zu berechnen?
- Woher stammen diese Daten? Wie können die Daten zugänglich gemacht werden?
- Wie müssen die Daten abgelegt werden, um sie möglichst effizient auswerten zu können?
- Wie können die Daten grafisch am wirkungsvollsten aufbereitet werden?

Die Beantwortung solcher und ähnlicher Fragen bestimmt meine tägliche Arbeit. Dabei wird das Ziel verfolgt, gemeinsam mit meinen Kunden Auswertungssysteme zu entwickeln, die bei der Steuerung und Lenkung ihres Unternehmens unterstützen.

Nun stellte sich also die Frage, welches Thema es für mich wert ist, viel Zeit und Arbeit zu investieren. Letztendlich war die Beantwortung dieser Frage relativ einfach und wurde durch das in der näheren Vergangenheit mit am stärksten gehypte Thema in der Informationstechnologie quasi von allein gelöst.

Gemeint ist der Begriff „Big Data“.

In den vergangenen Jahren ist wohl kaum ein Begriff derart überstrapaziert worden. Gestützt durch kontinuierlich günstiger werdenden Hauptspeicher und neue Mechanismen zur parallelen Datenverarbeitung haben sich Marketingabteilungen unterschiedlichster Softwarehersteller sowie Konferenz- und Schulungsanbieter auf die Techniken und Schlagwörter hinter Big Data gestürzt und einen klassischen Hype rund um das Thema ausgelöst. Dieser Hype hat auch die Entwicklung des Big-Data-Frameworks Apache Hadoop mit dem gesamten damit verbundenen technischen Ökosystem vorangetrieben. Für mich stand somit schnell fest, mich in meiner Masterthesis mit dem Begriff „Big Data“ näher beschäftigen zu wollen. Da das Apache

Hadoop Framework die wohl mit Abstand bekannteste Technologie in diesem Zusammenhang verkörpert, entschied ich mich dazu, mich mit dieser ebenfalls näher auseinanderzusetzen.

1.2 Zielsetzung

Mit der Festlegung, eine Arbeit über das Thema Big Data und Apache Hadoop verfassen zu wollen, endete dann jedoch auch die Flut der Erkenntnisse. Ursache hierfür war jedoch nicht das Fehlen einer schier unendlichen Zahl von Technologien, die hätten näher betrachtet werden können, sondern das Fehlen eines konkreten Bezugs, auf den diese hätten angewendet werden können. Das Verfassen dieser Abhandlung sollte schließlich auch einen konkreten Nutzen für meine tägliche Arbeit aufweisen.

Das Problem an einem klassischen Hype-Thema ist, dass gefühlt jede Person im beruflichen Umfeld – sei es Kollege oder Kunde – eine konkrete Vorstellung von dem hat, was Big Data eigentlich ist sowie welche Chancen und Möglichkeiten sich mit Big Data bieten. Betrachtet man dann allerdings die Realität, wird schnell klar, dass die allgemein geglaubte Verständlichkeit weit weniger vorherrscht als zunächst angenommen. In diesem Zusammenhang bin ich auf eine Grafik gestoßen, die das Problem, dass jeder eine Meinung zu Big Data hat, aber sich niemand genau sicher ist, was Big Data denn nun eigentlich ist, sehr anschaulich darstellt. Die Abbildung zeigt das Bildnis von blinden Männern und einem Elefanten.

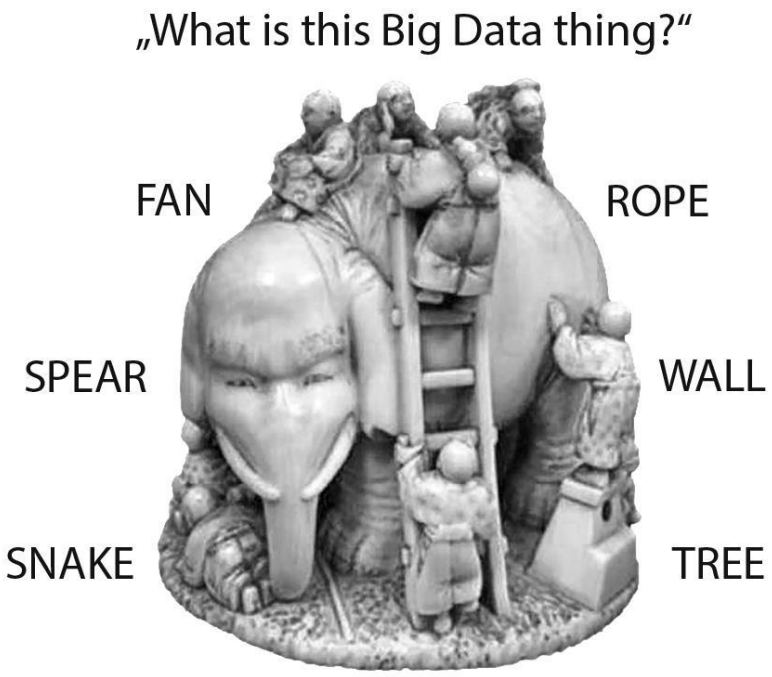

Abbildung 1: Bildnis von den blinden Männern und dem Elefanten (Phil Radley 2015)

In diesem Bildnis befühlen sechs blinde Männer, die noch nie zuvor von einem Elefanten gehört haben, einen solchen, und jeder der sechs Männer hat am Ende eine andere Vorstellung davon, was ein Elefant ist.

Diese Situation passt sehr gut zu den Erfahrungen, die ich im beruflichen Alltag, aber auch bei der Recherche für diese Arbeit sammeln konnte. Abhängig vom Background und den Erfahrungen des jeweiligen Gesprächspartners scheinen Big Data und die damit verbundenen Technologien stets etwas anderes zu sein. Natürlich gibt es bei allen Personen auch wiederkehrende Elemente, die unmittelbar mit dem Begriff „Big Data“ verknüpft werden, etwa die Annahme einer – für ihre jeweiligen Verhältnisse – besonders großen Datenmenge. Gleichwohl ist verwunderlich, wie unterschiedlich und facettenreich der Begriff „Big Data“ interpretiert wird. Da wäre zum Beispiel der Manager, der in Big Data die Möglichkeit sieht, aus bislang ungenutzten Informationen einen zukünftigen Mehrwert generieren zu können. Oder aber der klassische SAP-Berater und SAP-Kunde, die mit dem Begriff „Big Data“ häufig die SAP-In-Memory-Datenbank SAP Hana assoziieren. Sei aus, weil einerseits durch die verhältnismäßig neue Technologie rentable Beratungsaufträge in Aussicht stehen, oder andererseits wegen der Ambition, mittlerweile altbacken wirkende R3-Systeme wieder richtig auf Vordermann zu bringen. Diese Liste könnte noch einige Zeit so fortgeführt werden, indes reichen die beiden Beispiele völlig aus, um die Intention hinter dem Problem zu verdeutlichen.

In Anbetracht dessen und in Erinnerung an das Bildnis der blinden Männer und dem Elefanten wirkt es fast wie eine Ironie des Schicksals, dass das Logo des Apache-Hadoop-Software-Projekts von einem kleinen gelben Elefanten geziert wird.

Abbildung 2: Apache-Hadoop-Logo (The Apache Software Foundation 2015)

Nun wirft sich jedoch die Frage auf, welches Ziel ich mit dem Verfassen dieser Arbeit verfolge und weshalb ich mich zunächst maßgeblich darum bemühe, zu veranschaulichen, wie groß die Verwirrung rund um das Thema Big Data ist. Bedenkt man dazu auch noch, dass der Begriff „Big Data“ nicht mit einer Silbe im Titel dieser Studie erwähnt wird, ist die Verwirrung letztendlich perfekt. Die Antwort auf die Frage ist einfach. Mir ging es, als ich ein Thema für meine Masterthesis gesucht habe, nicht anders, als es den Personen geht, die ich im obigen Absatz zu skizzieren versucht habe. Auch ich glaubte seinerzeit, eine konkrete Vorstellung von dem zu haben, was Big Data verkörpert und welche Begriffe und Technologien damit verknüpft sind. Im Rahmen meines Masterstudiengangs habe ich die Vorlesung „E-Business und Internetdatenbanken“ besucht, innerhalb derer auch etliche Technologien behandelt wurden, die eng mit dem Begriff „Big Data“ verwandt sind. Letztendlich hat das Hadoop Framework bei mir den intensivsten Eindruck hinterlassen, sodass ich zu einer jener Personen wurde, die den Begriff „Big Data“ unmittelbar mit ebenjener speziellen Technologie verbanden.

Die Tatsache, dass ich mich in der Einleitung als SAP Business Intelligence Consultant vorgestellt habe, dessen Fokus auf der SAP-BI-Produktpalette liegt, ließe zu dem Schluss verleiten, dass der Begriff „Big Data“ für mich mit der SAP-In-Memory-Datenbank Hana Hand in Hand geht. Schließlich habe ich doch noch wenige Absätze zuvor beschrieben, dass exakt meine Berufsgruppe zu solchen Analogien neigt. Dass dies nicht der Wahrheit entspricht, hängt damit zusammen, dass die In-Memory-Datenbank-Technologie bei Weitem nicht so revolutionär ist, wie sie gerne von den Softwareherstellern dargestellt wird. Die Idee, eine Datenbank vollständig im Hauptspeicher eines Rechners abzulegen, ist fast so alt wie das Konstrukt der relationalen Datenbanken selbst. Nichtsdestotrotz scheiterte sie in der Umsetzung lange Zeit an den, verglichen mit konventionellen Speichersystemen, extrem hohen Preisen für Hauptspeicher, sodass solche Systeme nie rentabel betrieben werden konnten.

Von dem Irrglauben geleitet, eine genaue Vorstellung des Begriffs „Big Data“ zu haben, ließ sich somit schnell ein Thema für eine Masterthesis finden, welches letztendlich wie folgt lauten sollte: „Machbarkeitsanalyse über den Aufbau eines Enterprise Data Warehous auf Basis von Apache Hadoop“. Die initiale Idee hinter diesem Thema sollte sich grob formuliert mit den folgenden Fragen beschäftigen:

- Was ist Big Data?
- Was ist Apache Hadoop?
- Welche Hadoop-Distributionen sind im Markt verfügbar?
- Was ist ein Enterprise Data Warehouse?
- Inwieweit lässt sich Hadoop mit Standard-Business-Intelligence-Werkzeugen kombinieren?

Während ich mich jedoch in die Literatur und das Thema einarbeitete, wurde aus anfänglich geglaubter Sicherheit schnell große Unsicherheit. Mit jeder Seite Text, die ich gelesen habe, und jeder Google-Suchanfrage, die ich startete, wurde meine Erkenntnis über meine Unkenntnis bestärkt. Kurzum: Das Apache Hadoop Framework ist ein gigantischer Dschungel von verschiedenen Softwareprojekten, in dem man sich sehr schnell verlaufen kann. Ehe man sich‘s versieht, beschäftigt man sich auf tiefster Systemebene mit der Konfiguration von maximalen Dateiblockgrößen auf den einzelnen Hadoop-Knoten. Das eigentliche Ziel, die oben genannten Fragen zu beantworten, rückt dabei schnell in die Ferne. Darüber hinaus ließ sich zudem relativ schnell eine weitere Erkenntnis erlangen. War die initiale Hauptprämisse die Fragestellung, ob sich auf Basis von Apache Hadoop ein Enterprise Data Warehouse aufbauen lässt, so kann diese Frage nicht allumfassend bearbeitet werden. Dies liegt zum einen daran, dass auch der Begriff des Enterprise Data Warehouse sehr weit, wenn auch nicht so weit wie der Begriff „Big Data“, ausgelegt werden kann. Zum anderen ist die Frage, ob ein Data Warehouse auf Basis von Apache Hadoop erstellt werden kann, schnell beantwortet. Wie bei allen Hype-Themen wird ein Hype auch von jenen befeuert, die an diesem verdienen wollen, was im Falle von Big Data und Hadoop nicht selten Softwarehersteller sind, die ein breites Spektrum an Tools im Portfolio haben, die nahezu alle erdenklichen Konstellationen abdecken. Wie eine solche Implementierung dennoch genau erfolgen oder aussehen könnte, soll der Vollständigkeit halber ein Teil dieser Abhandlung sein.

Statt also die oben gestellten Fragen beantworten zu wollen, habe ich mich dazu entschieden, vielmehr die Arbeit zu schreiben, die ich selbst zu Beginn meiner Recherche gerne hätte lesen wollen. Auf der einen Seite will und kann ich nicht versuchen, den Begriff „Big Data“ zu definieren, und auf der anderen Seite ist das Hadoop Framework eben nicht nur das, für was es von vielen Leuten gehalten wird, sondern noch

viel mehr. Der Versuch, all dies in einer vom Umfang überschaubaren Arbeit behandeln zu wollen, käme einer Sisyphos-Aufgabe gleich und würde vor allen Dingen den Möglichkeiten des Hadoop Frameworks bei Weitem nicht gerecht werden.
Für die vorliegende Studie ergibt sich mithin das folgende Ziel: Sie soll als Einstieg in die Big-Data- und Hadoop-Welt für Berufsgruppen dienen, die bereits Erfahrungen mit den Begriffen „Data Warehousing“, „Business Intelligence“ und „Big Data“ besitzen. Klassischerweise sind dies IT-Berater, Controller oder auch Entscheider in mittelständischen oder größeren Unternehmen.

1.3 Aufbau

Die Arbeit ist wie folgt aufgebaut:
Im ersten Abschnitt müssen einige Grundlagen erörtert werden, die für das weitere Lesen und Verstehen dieser Arbeit vorausgesetzt werden. Der Grundlagenteil konzentriert sich auf die drei folgenden Bereiche und ist so formuliert, dass kenntnisreiche Leser diesen Abschnitt vollständig überspringen oder aber nur die für sie relevanten bzw. unbekannten Teile lesen können:

- Enterprise Data Warehouse (klassisch),
- Big Data,
- Apache Hadoop Framework.

Im darauffolgenden Abschnitt wird anhand verschiedener Kriterien evaluiert, ob Apache Hadoop sich als Grundlage für ein Enterprise Data Warehouse eignet bzw. inwieweit klassische Business-Intelligence-Anforderungen realisiert werden können. Auf diese technische Überprüfung folgt die Vorstellung verschiedener architektonischer Ansätze, wie Apache Hadoop sinnvoll in ein bestehendes Data-Warehouse-System integriert werden kann. Abschließend wird die Arbeit mit einem kurzen Fazit zu den gewonnenen Erkenntnissen abgeschlossen.

Bevor mit dem Studium dieser Abhandlung begonnen wird, soll noch ein kurzer Hinweis an den Leser erfolgen. Die Arbeit behandelt eine Vielzahl komplexer Themen. Aus diesem Grund können nicht alle Themen bis ins tiefste Detail betrachtet und hinsichtlich jeder Facette untersucht werden. Die vorliegende Studie ist primär darauf ausgerichtet, einen Überblick über die Möglichkeiten von Apache Hadoop im Bereich des Enterprise Data Warehousing zu geben. Aus diesem Grund kann es sein, dass wichtige Themen einzelner Teilbereiche nur kurz oder überhaupt nicht betrachtet werden. Eines dieser Teilgebiete sind der große Bereich des Datenschutzes und die damit verbundenen Diskussionen. Die gesamte Arbeit ist aus diesem Grund ohne Betrachtung dieses Themas formuliert worden. Eine zusätzliche Behandlung des Punktes Datenschutz wäre im Rahmen dieser Studie nicht zielführend gewesen und ist damit zu vernachlässigen.

2 Einführung in Data-Warehouse-Systeme

2.1 Begriffseinordnung

Im folgenden Abschnitt soll der Begriff des Data Warehouse vorgestellt werden. Hierzu sollen zunächst die Idee hinter dem Begriff des Data Warehouse und die damit verbundenen Ziele erläutert werden. In der Literatur findet sich ein weitgehender Konsens zur Klassifikation von Informations-und Kommunikationssystemen, deren Bestandteil auch die Idee des Data Warehouse ist.

Gabriel et al. teilen Informationssysteme zunächst in verschiedene Klassen von (IT-)Systemen ein. Als globale Kategorie betriebswirtschaftlicher IT-Systeme werden die Informations- und Kommunikationssysteme betrachtet. (Gabriel et al. 2009, S. 2)

In der nachstehenden Abbildung lässt sich diese globale Kategorie der betriebswirtschaftlichen Informations- und Kommunikationssysteme erkennen, die wiederum in verschiedene Stufen unterteilt werden kann. (Gabriel et al. 2009, S. 2)

Unterhalb dieser Einteilung sind nicht nur Subsysteme im Sinne zielgerichteter Informationsbereitstellung zu verstehen, sondern auch Systeme, die der Abbildung der Leistungs- und Austauschbeziehungen innerhalb einer Unternehmung, aber auch zwischen Unternehmen und der Umwelt dienen. (Gabriel et al. 2009)

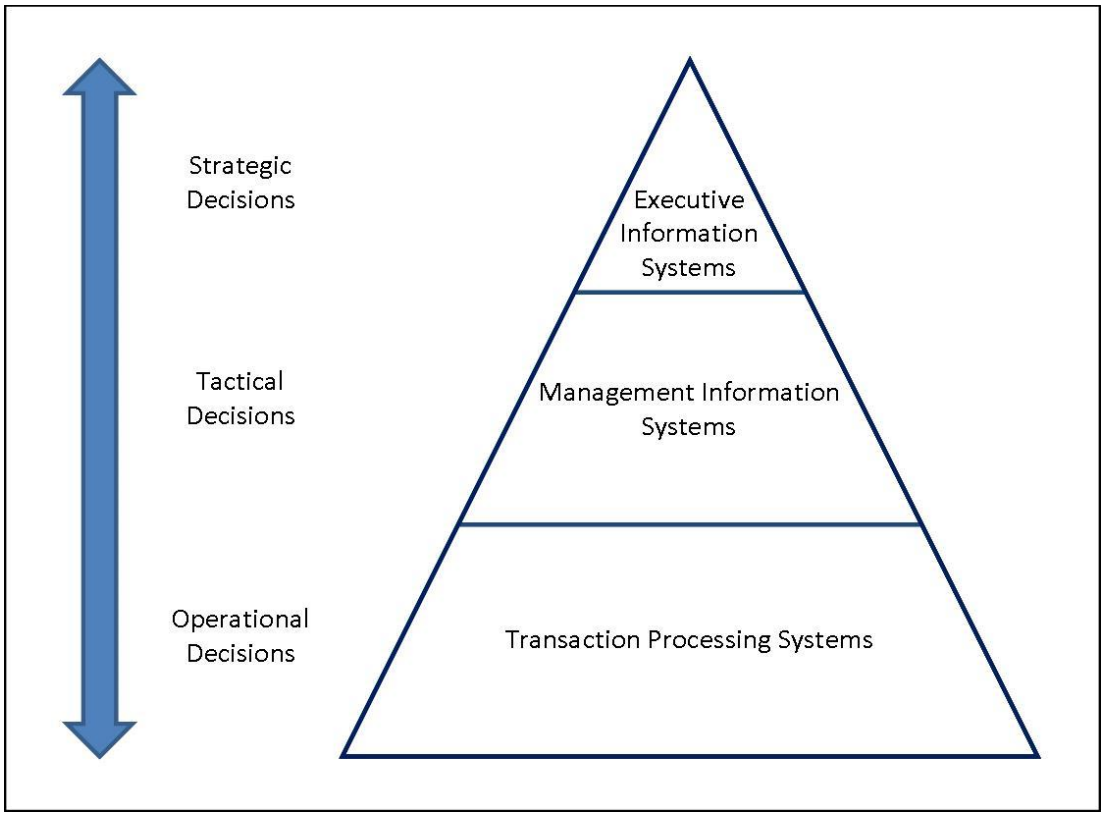

Abbildung 3: Dreistufige Informationssystem Pyramide (Kimble 2010)

Informations- und Kommunikationssysteme (IuK-Systeme) lassen sich in zwei Typen unterscheiden. Die operativen Informationssysteme stellen die Klasse der Administrations- und Dispositionssysteme dar, oft auch als *Online-Transactional-Processing-Systeme* (OLTP-Systeme) bezeichnet. Sie unterstützen bzw. begleiten Aufgaben auf operativer Ebene der Unternehmung. (Bauer und Günzel 2013, S. 9)

Diesen Systemen stehen die analyseorientierten Informationssysteme (AIS) gegenüber. Aufgabe dieser Systeme ist die Versorgung von Fach- und Führungskräften mit planungs- und entscheidungsrelevanten Informationen. (Gabriel et al. 2009, S. 3)

Bauer et al. stimmen der Differenzierung in operative respektive transaktionale und analytische Systeme zu, nehmen jedoch eine detailliertere Klassifikation derselben vor. Hierbei können drei wesentliche Kategorien und Merkmale bei operativen und analytischen Systemen unterschieden werden: (Bauer und Günzel 2013)

Anfragen

Aufbau und Charakteristika einer Abfrage haben einen entscheidenden Einfluss auf die Anfrageverarbeitung und Speicherung der Daten. Eine Anfrage auf operative Systeme kann alle möglichen Formen des Zugriffs umfassen (Lesen, Schreiben, Modifizieren und Löschen). Analytische Systeme beantworten typischerweise ausschließlich lesende Anfragen. Nur wenn neue Daten eingepflegt werden sollen, findet einmalig ein Schreibvorgang statt. In diesem Fall umfasst eine Transaktion gleich mehrere Datensätze. Anfragen an operative Systeme hingegen beschränken sich meist auf sehr wenige Datensätze, sind entsprechend kurz und einfach strukturiert. Analytische Systeme sind zur erweiterten Informationsgewinnung aus den Daten konzipiert. Sie stellen komplexe Anfragen und greifen dabei auf viele Millionen Datensätze zu. So ist es auch zu erklären, dass die Transaktionsdauer bei operativen Systemen nur sehr kurz ausfällt, bei analytischen Systemen indes deutlich länger dauert. (Bauer und Günzel 2013, S. 9)

Anfragen	Transaktional	Analytisch
Fokus	Lesen, Schreiben, Modifizieren, Löschen	Lesen, periodisches Hinzufügen
Transaktionsdauer und –typ	Kurze Lese-/Schreibtransaktionen	Lange Lesetransaktionen
Anfragestruktur	Einfach strukturiert	Komplex
Datenvolumen einer Anfrage	Wenige Datensätze	Viele Datensätze

Tabelle 1: Gegenüberstellung der Anfragecharakteristika von transaktionalen und analytischen Anwendungen (Bauer und Günzel 2013, S. 10)

Daten

Als Beispiel für einen transaktionalen Betrieb kann die Personalabteilung eines Unternehmens dienen. Hier werden Daten über Mitarbeiter in einer Datenbank gespeichert und verwaltet. Typischerweise ist hierfür eine Datenbank ausreichend. Mögliche

Transaktionen könnten das Hinzufügen neuer oder das Entfernen ausgeschiedener Mitarbeiter sein. Ein analytisches Informationssystem beinhaltet diese Daten (oder einen Auszug davon) neben den Daten vieler anderer Datenbanken operativer Systeme. Man spricht dabei von abgeleiteten Daten. Daten aus dem transaktionalen Betrieb sind meist zeitaktuell, nicht abgeleitet, autonom, dynamisch und stammen aus einer Datenquelle. Unter dynamisch ist die häufige Modifikation von Daten zu verstehen, anders als es bei der Analyseorientierung der Fall ist. Hier sind die einmal eingepflegten Daten als stabil zu betrachten. Darüber hinaus setzt sie konsolidierte, integrierte und oft auch aggregierte Daten voraus. Möchte man nun jeden Status eines Datensatzes über die Zeitachse erfassen, sodass man sämtliche Änderungen, die ihm zuteilwurden, zurückverfolgen kann, und darüber hinaus eine Vielzahl von operativen Informationssystemen in das AIS integrieren, so wächst das Datenvolumen schnell in einen hohen Giga- oder gar Terabyte-Bereich. (Bauer und Günzel 2013, S. 10)

Daten	Transaktional	Analytisch
Datenquelle	Meist eine	Mehrere
Eigenschaften	Nicht abgeleitet, zeitaktuell, autonom, dynamisch	Abgeleitet, konsolidiert, historisiert, integriert, stabil
Datenvolumen	Megabyte - Gigabyte	Gigabyte - Terabyte
Zugriffe	Einzeltupelzugriff	Bereichsanfragen

Tabelle 2: Gegenüberstellung der Datencharakteristika von transaktionalen und analytischen Anwendungen (Bauer und Günzel 2013, S. 10)

Anwender

Sind bei transaktionalen Systemen noch Sachbearbeiter mit der Pflege der Daten beschäftigt, ist es bei einem AIS meist ein Analyst, Manager oder Controller. In beiden Fällen wird jedoch eine schnelle Antwortzeit von den Systemen erwartet. Bei analytischen Systemen stellt dies aufgrund komplexer Anfragen und großer Datenmengen schnell eine Herausforderung dar. (Bauer und Günzel 2013)

Anwender	Transaktional	Analytisch
Anwendertyp	Ein-/Ausgabe durch Sachbearbeiter	Auswertungen durch Manager, Controller, Analyst
Anwenderzahl	Sehr viele	Wenige
Antwortzeit	Millisekunden - Sekunden	Sekunden - Minuten

Tabelle 3: Gegenüberstellung der Anwendercharakteristika von transaktionalen und analytischen Anwendungen (Bauer und Günzel 2013, S. 11)

Operative/transaktionale Informationssysteme

Operative Informationssysteme setzen sich aus mehreren Einzelsystemen zusammen. Sie zeichnen dafür verantwortlich, die operativen und Leistung generierenden Prozesse zu unterstützen. Mit diesen Systemen werden standardisierte Arbeitsabläufe automatisiert. Daten, die im Rahmen dieser Prozesse entstehen, werden von den operativen Informationssystemen effizient verarbeitet und in (operativen) Datenbanksystemen gespeichert sowie verwaltet. Beispiele für diese Systeme sind Anwendungen zur Personal-, Kunden-, Lieferanten- oder Produktverwaltung, Lagerhaltung oder Auftragserfassung. (Gabriel et al. 2009, S. 3)

Wie zuvor im obigen Abschnitt angedeutet, ließen sich die operativen Systeme weiter in Administrations- und Dispositionssysteme unterteilen. Für die Erstellung dieser Ausarbeitung ist eine weitere Verfeinerung der Begriffsdefinition jedoch nicht erforderlich. (Gabriel et al. 2009)

Analyseorientierte Informationssysteme

Analyseorientierte Systeme gewinnen ihre Berechtigung durch die hohe Bedeutung von Informationen für die Erhaltung der Wettbewerbsfähigkeit. Unternehmen, die in der Lage sind, durch den Einsatz innovativer Technologien schnell und flexibel auf sich ändernde Marktfaktoren und Kundenbedürfnisse zu reagieren, können sich so einen Vorsprung gegenüber Wettbewerbern sichern. In den operativen Datenbanken befinden sich viele Daten, die jedoch nur in einem geringen Umfang genutzt werden. Das begründet sich im Wesentlichen dadurch, dass die Daten nicht in der geeigneten Form oder zum richtigen Zeitpunkt zur Analyse und Entscheidungsfindung vorliegen. Die Anforderung besteht daher darin, den Zugriff auf diese Daten unternehmensweit zu ermöglichen, verbunden mit entsprechenden Analysemöglichkeiten, um die darin enthaltenen Informationen für die Entscheidungsfindung nutzbar zu machen. (Gabriel et al. 2009, S. 4)

Aus diesen Anforderungen folgt das Ziel, aus den Datenbeständen personen-, problem- und situationsgerechte Informationen bereitzustellen. Gabriel et al. führt aus diesem Grund neben dem Begriff des analyseorientierten Informationssystems den Begriff der Business-Intelligence-Systeme ein, die diese Aufgabe bewältigen sollen. (Gabriel et al. 2009, S. 4)

Bauer et al. beschreiben sie hingegen als *Data-Warehouse-Systeme*, welche die Integration, Datenhaltung und Analyse umfassen. (Bauer und Günzel 2013, S. 8)

2.2 Data Warehouse (System)

Der Begriff des Data Warehouse bildete sich Mitte der 1980er-Jahre maßgeblich zu dem heraus, als was er heute bekannt ist. (Devlin 1997, S. 7)

Dieser von Bauer et al. entnommenen, bereits sehr technisch formulierten Definition stehen in der Literatur alternative Ansätze gegenüber. Bauer et al. geben bereits in der Definition an, dass es sich bei einem Data-Warehouse-System stets um eine Datenbank handelt, die die Aufgabe der Integration, Datenhaltung und Analyse umfasst. Dem gegenüber steht z. B. die stärker betriebswirtschaftlich geprägte Sichtweise von H. Groffmann. Dessen Ansicht nach stellt das Data Warehouse ein Konzept dar, „[…] das beschreibt, wie eine logisch einheitliche, konsistente Datensammlung für die Managementunterstützung gestaltet und betrieben werden kann“. (Hans-Dieter Groffmann 1997)

Im Modell von Bauer et al. ist ein „[…] Data Warehouse in ein Data-Warehouse-System eingebettet“. Hieraus folgt, dass es das Data-Warehouse-System ist, welches „[…] alle für die Integration und Analyse notwendigen Komponenten [beinhaltet]“. (Bauer und Günzel 2013, S. 8)

Die Idee wurde dabei maßgebend von zwei Seiten beeinflusst: zum einen aus der Nutzerperspektive, die neue wissenschaftliche und betriebswirtschaftliche Anforderungen und Fragestellungen vorgegeben hat. Zum anderen aus der technischen Perspektive, die Datenbanksysteme und Datenintegrationsmöglichkeiten in den Vordergrund stellte. Von diesen beiden Seiten betrachtet kann ein Data Warehouse als eine Datenbank, „[…] die aus der technischen Sicht Daten aus verschiedenen Datenquellen integriert und aus der betriebswirtschaftlichen Sicht dem Anwender diese Daten zu Analysezwecken zur Verfügung stellt“, aufgefasst werden. (Bauer und Günzel 2013, S. 10)

Alternativ verkörpert ein Data Warehouse nach Gabriel keine Datenbank, beinhaltet jedoch „[…] eine von den operativen DV-Systemen physikalisch getrennte Datenbank […]“. (Gabriel et al. 2009, S. 7)

An dieser Stelle wird ersichtlich, dass zum einen die Begriffe des Data Warehouse und des Data-Warehouse-Systems inhaltlich abgegrenzt werden müssen. Auf Grundlage der verschiedenen Benennung und Sichtweisen ergeben sich auch in den folgenden Definitionen unterschiedliche Ansätze. Im folgenden Abschnitt werden einige Definitionen zum Data Warehouse behandelt. Auf Basis dieser Definitionen soll anschließend eine für diese Arbeit geltende Abgrenzung der Begrifflichkeiten vorgenommen werden.

<u>Definition nach William H. Inmon</u>

William H. Inmon ist einer der am meisten zitierten Koryphäen im Bereich des Data Warehousing.

„A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management‘s decisions.“ (Inmon 1996, S. 33)

Analyse:

- Fach- bzw. Themenorientierung (subject orientation): Die Datenbasis zielt nicht mehr auf die Erfüllung einer einzelnen konkreten Aufgabe ab, wie z. B. eine Materialstammdatenverwaltung, sondern auf die Modellierung eines gesamtheitlichen Anwendungsmodells.
- Vereinheitlichte Datenbasis (integration): Die Analyse der Daten erfolgt nicht mehr auf einer isolierten Datenquelle, sondern auf den integrierten Daten verschiedener Quell- und Datenbanksysteme.
- Permanente Datenbasis (non-volatile): Einmal im Data Warehouse vorhandene Daten werden nicht mehr verändert und/oder entfernt. Der Datenbestand ist als stabil zu betrachten. Analyse-Ergebnisse sind zu jedem späteren Zeitpunkt aufgrund der nicht vorhandenen Volatilität reproduzierbar.
- Historisierte Daten (time variance): Die Daten sind so abgelegt, dass sie vor allen Dingen für den Vergleich über verschiedene Zeiträume geeignet sind. Das dauerhafte Speichern und Historisieren von Daten ist damit eine notwendige Voraussetzung.
- Entscheidungsunterstützung (management‘s decisions): Die Datenhaltung dient in erster Linie zur Unterstützung der betriebswirtschaftlichen Entscheidungsfindung.

(Inmon 1996, S. 33)

Kritiken

Auch wenn der Ansatz von William H. Inmon in der Literatur oft verwendet wird, so ist er jedoch nicht frei von Kritik. (Inmon 1996, S. 33)

Bauer et al. bemängeln so z. B., dass es der Definition an Aussagekraft fehle. Zudem sei sie so stark eingeschränkt, dass viele Ansätze und Anwendungsgebiete bei strikter Einhaltung von vornherein herausfielen. (Bauer und Günzel 2013, S. 8)

Ein Problem an der Kritik von Bauer et al. besteht darin, dass die Autoren die Nennung konkreter Anwendungsgebiete und Ansätze schuldig bleiben. Auch stellen sie in ihrer Analyse der Definition von Inmon keine Widersprüchlichkeiten zu heutigen Anforderungen heraus, wodurch es schwerfällt, ihrer Argumentation in der Kritik zu folgen. (Bauer und Günzel 2013, S. 8)

Wie Bauer et al. gelangt T. Zeh in seinem Kurzaufsatz „Eine kritische Betrachtung der Data-Warehouse-Definition von Inmon“ ebenfalls zu dem Entschluss, dass der Ansatz von Inmon zu stark eingeschränkt sei, setzt sich jedoch in seinem Aufsatz intensiver mit diesem auseinander. Hierdurch wird deutlich, woraus sich die einschränkenden Merkmale ableiten lassen. Diese sollen im Folgenden kurz vorgestellt werden,

um die Unterschiede zu heutigen Einsatzgebieten von Data Warehousing und Date-Warehouse-System aufzuzeigen. (Zeh 2003)

Inmon führt in seiner Definition eine Themenfokussierung (subject orientation) an. Ein Data Warehouse soll seiner Auffassung nach einer solchen unterliegen. Hierzu liefert er Beispiele für Themen wie Kunden, Lieferanten, Produkte oder Aktivitäten. Ein solcher Themenschwerpunkt ist für Inmon ein wesentliches Merkmal eines Data Warehouse. Im Gegensatz zur Themenfokussierung sieht Inmon die Prozess- oder Funktionsorientierung, für die er Beispiele aus dem Bankensektor anführt. Kredite, Ersparnisse oder Bankkarten (z. B. EC- oder Kreditkarten) stellen aus seiner Sicht keine Themenfokussierung dar und würden somit kein Data Warehouse rechtfertigen. Zeh kritisiert an dieser Stelle sowohl eine willkürliche Abgrenzung der Begrifflichkeiten als auch die reine Themenabgrenzung per se. Er sieht keinen Grund, weshalb der Datenbestand eines Data Warehouse einmalig festgelegt, dann jedoch nicht mehr angepasst werden sollte, ohne im Vorfeld die Anforderungen zu kennen. Vielmehr sollte sich der Datenbestand eines Data Warehouse am Warenbestand eines physischen Warenlagers orientieren, der sich ständig anhand der Nachfrage der Verbraucher ändert. (Zeh 2003, S. 33)

Hinsichtlich des Grundgedankens der Historisierung geht Zeh mit der Meinung von Inmon konform, führt jedoch nachvollziehbare Argumente an, weshalb die ursprüngliche Auffassung Inmons heute nicht mehr in jedem Fall zutreffend sein muss. Inmon fordert in seinem Ansatz eine spezielle Zeitbetrachtung in Form von Momentaufnahmen zu vorgegebenen Zeitpunkten, die wiederum mit einer schubweisen Beladung des Data Warehouse eingeht. Das Problem an dieser Art der Datenbeladung ist eine umso stärker abnehmende Datenaktualität im Data Warehouse, desto größer die Zeitintervalle zwischen den einzelnen Beladungen ausfallen. Laut Zeh existieren durchaus Anwendungsszenarien, bei denen eine sehr kurze Reaktionszeit gefordert wird, sodass eine schubweise Beladung diese Anforderung nicht erfüllen könne. Als Beispiel nennt er hier die Analyse von Aktienkursen. Die Intervalle zwischen der Beladung müssten so kurz gewählt werden, dass sich eine schubweise Datenbeladung zu einem Datenstrom verschieben würde.

Darüber hinaus betrachtet Zeh das Konstrukt der Momentaufnahmen (Snapshot) kritisch. Er stellt hier die Frage, weshalb zeitlich stabile Datenobjekte, wie etwa physikalische Eigenschaften eines Produkts, in eine für sie unnatürliche Zeitstruktur hineingepresst werden sollten. Neben der Unnatürlichkeit dieses Vorgehens betrachtet er zudem die hierdurch auftretende, zum Teil erhebliche Redundanz der Daten als kritisch. Zeh diskutiert den Begriff time variant, den man auch als Integrationsaspekt (integrierte Datenbasis) auffassen könnte, würde dieser nur als Zeit begriffen, womit gemeint ist, den Zeitpunkt der Integration eines jedes Datensatzes festzuhalten, nicht

jedoch mit Momentaufnahmen (Schnappschussverfahren) zu historisieren. Die Forderung nach dem Zeitaspekt wäre somit bereits hinreichend durch die Integrationsforderung gedeckt, sodass eine Momentaufnahme kein Muss darstellen würde, sondern nur eine von mehreren Optionen, zeitabhängige Daten zu halten. (Zeh 2003, S. 34)

Inmon vertritt die Ansicht, dass Daten aus einer Momentaufnahme, nachdem sie in das Data Warehouse geladen wurden – mit Ausnahme von Korrekturen – nicht mehr verändert werden dürfen (non-volatil). Als Grund hierfür sieht er die Gefahr von Änderungsanomalien sowie technische Gründe: Backup und Recovery seien nicht notwendig, ebenso wenig Vorkehrungen zur Transaktions- und Datenintegrität. Zeh kann den inhaltlichen Grund für dieses Vorgehen widerlegen, indem er angibt, dass Änderungsanomalien hinfällig werden, sobald der Inhalt des Data Warehouse normalisiert strukturiert wird und mithin frei von Redundanz ist. Die von Inmon angeführten technischen Gründe griffen laut Zeh ohnehin nur bei Massendaten, um möglichen Performanzproblemen vorzugreifen. Dieses Problem ließe sich jedoch durch den Einsatz einer geschickten Datenorganisation, wie der Beschränkung auf die Aktualisierung geänderter Daten in normalisiert strukturierten Daten, sowie die Verwendung moderner, leistungsfähiger Datenverwaltungssysteme umgehen. Dieses Problem der Datenvolatilität wird in der folgenden Grafik schematisch dargestellt. (Zeh 2003, S. 34)

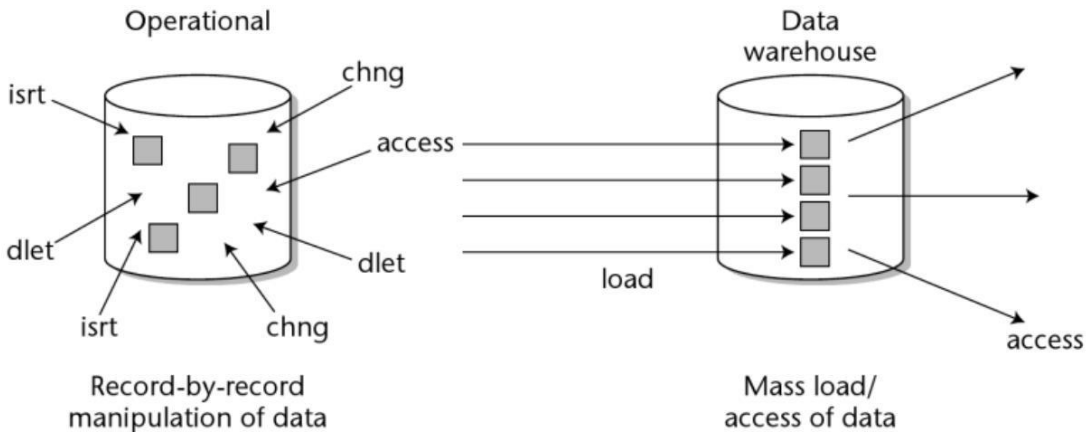

Abbildung 4: Volatilität der Daten (Inmon 2005, S. 32)

Die von Zeh geforderte Notwendigkeit von Änderungen an Daten, die über Korrekturen hinausgehen, führt er an dieser Stelle nicht weiter aus. Wird indes bedacht, dass er Daten vorsieht, die Zeit ihres Lebens keine Veränderung erfahren und somit nicht Bestandteil einer Momentaufnahme sind, ist die Möglichkeit der nachträglichen Änderung die einzige Möglichkeit, diese Daten dennoch zu ändern. Die Auslegung der time variance von Zeh setzt die Möglichkeit der nachträglichen Änderung somit sogar zwingend voraus. (Zeh 2003, S. 34)

<u>Definition nach Bauer et al.</u>

Von Bauer et al. wird die folgende Definition eines Data Warehouse angeführt:
„Ein Data Warehouse ist eine physische Datenbank, die eine integrierte Sicht auf beliebige Daten zu Analysezwecken ermöglicht.“ (Bauer und Günzel 2013)

Im gleichen Maße wie die Kritik von Bauer et al. an der Data-Warehouse-Definition von Inmon lässt sich auch aus dessen eigener Definition kein Aufschluss darüber gewinnen, welche Aspekte er exakt kritisiert. Überraschend ist hingegen, dass sich relativ viele Parallelen zwischen der Definition von Bauer et al. und Inmon erkennen lassen. Inmons *collection of data* findet sich bei Bauer in der physischen Datenbank wieder, und die integrierte Datenbasis spiegelt sich als integrierte Sicht wider. Auch der von Inmon beschriebene Zweck des *support of management decisions* lässt sich bei Bauer erkennen, wird allerdings als Analysezweck weiter gefasst und ermöglicht so größeren Spielraum bei der Interpretation. Wird die Definition von Bauer et al. mit der von Inmon verglichen und dabei jegliche Wertung seitens Bauer außen vor gelassen, erscheint die Definition von Inmon durch Nennung konkreter Vorgaben aussagekräftiger als die von Bauer, ist dadurch aber auch gleichermaßen einschränkend. (Bauer und Günzel 2013)

<u>Definition nach Gabriel et al.</u>
Die Definition von R. Gabriel ist ausführlicher als die von Bauer und/oder Inmon. Eine Besonderheit bildet die ausführliche Trennung der operativen (transaktionalen) Informationssysteme.
„Unter einem Data Warehouse ist eine Systemlösung zu verstehen, die die unternehmungsweite Versorgung der Front-End-Systeme zur Managementunterstützung mit den benötigten Informationen gewährleistet. Zweckmäßigerweise wird das Data-Warehouse getrennt von den operativen Vorsystemen aufgebaut und betrieben. Nur so lässt sich eine konsistente unternehmensweite Datenbasis etablieren, in die selektierte und verdichtete Informationen anwendungsgerecht aufbereitet einfließen und auf die interaktiv und intuitiv zugegriffen werden kann. Für die gespeicherten Dateninhalte ist deren thematische Ausrichtung sowie Vereinheitlichung, Dauerhaftigkeit und Zeitorientierung charakteristisch.“ (Gabriel et al. 2009, S. 43)

Gabriel et al. stellen in ihrer Definition deutlich heraus, dass unter einem Data Warehouse mehr als nur eine physische Datenbank zu verstehen ist.

Aufgrund der verschiedenen vorgestellten Definitionen und der damit verbundenen Begrifflichkeiten wird in dieser Arbeit zwischen den beiden folgenden Begriffen unterschieden:

- Data Warehouse,
- Data-Warehouse-System

Die folgende Abgrenzung soll dabei das Verständnis der beiden Begriffe im Rahmen dieser Ausarbeitung repräsentieren. (Gabriel et al. 2009, S. 43)

„Das Data Warehouse umfasst einen gegenüber operativen Systemen redundant gehaltenen integrierten Datenpool in Form einer oder mehrerer Datenbanken. Das Data-Warehouse-System beinhaltet darüber hinaus Werkzeuge und Programme zur Erzeugung, Aktualisierung und Nutzung des Data Warehouse. Es unterstützt die Integration der Datenbestände aus unterschiedlichen Quellen sowie deren flexible, interaktive Analyse.“ (Goeken 2006, S. 16)

Diese Definition soll noch um ein weiteres Synonym ergänzt werden. Das Konstrukt, welches hier als Data Warehouse betitelt wird, findet sich an anderer Stelle in der Literatur auch unter dem Begriff Enterprise Data Warehouse (EDW) wieder. (Gabriel et al. 2009, S. 47)

Zusammenfassend ergeben sich mithin die folgenden Punkte als primäre Aufgaben für ein Data-Warehouse-System:

- Datenbewirtschaftung,
- Datenhaltung (physisch),
- Datenauswertung.

Der Prozess des Data Warehousing beschreibt dabei den Fluss der Daten über alle Komponenten das Data-Warehouse-Systems hinweg – beginnend mit dem Datenbeschaffungsprozess über die Datenhaltung bis hin zur Datenauswertung. (Bauer und Günzel 2013, S. 9)

Diese Vorgänge rund um das Data Warehousing sollen in den folgenden Abschnitten näher betrachtet werden.

2.3 Data-Warehouse-Architektur

2.3.1 Problemstellung

Im folgenden Abschnitt sollen verschieden Ansätze von Data-Warehouse-Architekturen vorgestellt werden. Wie bereits bei den Definitionen gibt es auch bei den Architekturen unterschiedliche Ansätze. Um diese Ansätze miteinander vergleichen zu können, ist es zunächst notwendig, die einzelnen Komponenten eines Data Warehouse zu veranschaulichen. Anschließend können die vorgestellten Musterarchitekturen auf die Berücksichtigung dieser Komponenten hin untersucht und Differenzen zwischen den einzelnen Architekturen dargestellt werden.

Die untere Abbildung visualisiert ein vollständiges Data-Warehouse-System (exkl. Analysewerkzeuge) inklusive Abbildungen der einzelnen Datenintegrationsstufen und -schritte.

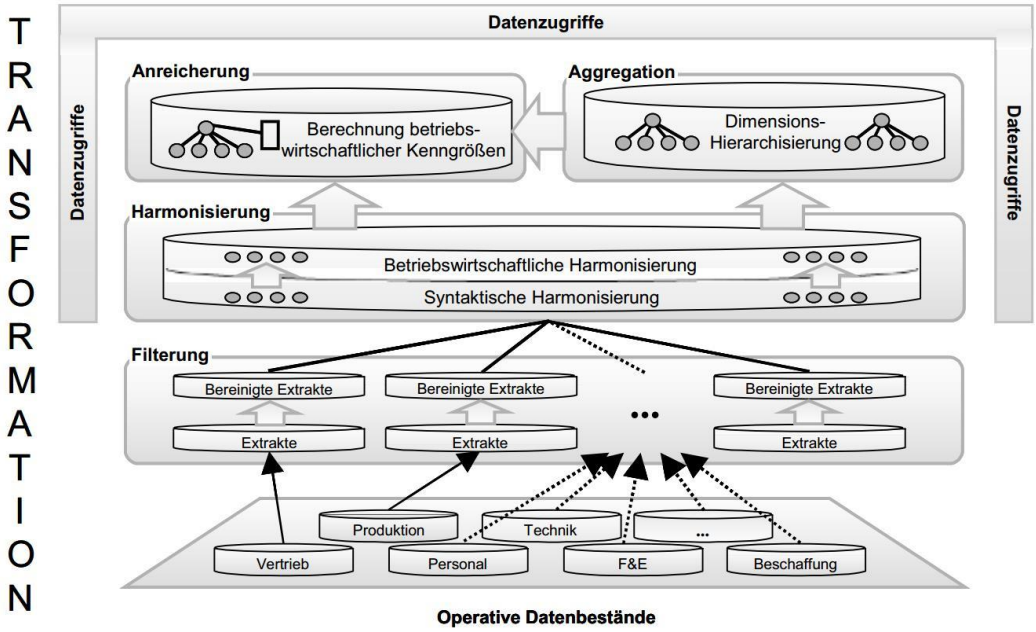

Abbildung 5: Data Warehouse – Gesamtschaubild inkl. ETL-Prozesse (H. Kemper: und R. Finger 2006, S. 116)

2.3.2 Datenquellen (Ebene)

Datenquellen dienen als Ausgangspunkt eines jeden Data-Warehouse-Projekts. Mit ihnen beginnt der Prozess des Data Warehousing, und ihre Beschaffenheit sowie Qualität haben einen unmittelbaren Einfluss auf das Analyseergebnis, welches Ziel des Prozesses ist. Auswahl und Sicherung der Datenqualität sind damit für ein erfolgreiches Data-Warehouse-Projekt von entscheidender Bedeutung. Klassischerweise wurden in der Historie ausschließlich betriebsinterne Daten als Datenquelle verwendet. Diese entstammen in der Regel den operativen, transaktionalen Systemen einer Unternehmung. (Bauer und Günzel 2013, S. 40)

Bei der Auswahl der Datenquellen gilt es, verschiedene Faktoren zu berücksichtigen, die, selbst bei physischer Verfügbarkeit der Daten, das Vorhaben, sie als Quelle zu verwenden, scheitern lassen können:

- Zweck des Data-Warehouse-Systems,
- Qualität der Quelldaten,
- Verfügbarkeit (rechtlich, sozial, organisatorisch, technisch),
- Kosten, um Quelldaten nutzbar zu machen.

(Bauer und Günzel 2013, S. 41)

<u>Zweck</u>

Eine Datenquelle muss für die angedachte Verwendung geeignet sein. Ein Projekt, für welches die kontinuierliche Anlieferung neuer Daten vorausgesetzt wird, kann nicht auf einer Datenquelle aufgebaut werden, deren Daten nach initialer Anlage nicht

weiter gepflegt werden. Eine Produktdatenbank eines Lieferanten, die nur in unregelmäßigen Abständen gepflegt wird, eignet sich beispielsweise nicht für das Vorhaben, Preisverläufe einzelner Waren tagesgenau zu erfassen. (Bauer und Günzel 2013, S. 41)

Datenqualität

Datenqualität bzw. mangelnde Datenqualität kann in vielen Fällen Ursache für das Scheitern eines Data-Warehouse-Projekts sein. Dies liegt zum einen daran, dass durch die fehlende Datenqualität erhebliche Zusatzkosten verursacht werden können, sodass ein Projekt jegliche Budgetplanungen sprengt, oder aber die nicht vorhandene Qualität aus fragwürdigen Analyseergebnissen resultiert, sodass die Zielvorgaben nicht erreicht werden können. Für ein erfolgreiches Data-Warehouse-Projekt müssen somit konkrete Anforderungen an die Datenqualität gestellt werden, die durch entsprechende Validierungen gesichert werden müssen. (Bauer und Günzel 2013, S. 42)

Liste typischer Qualitätsmängel:

- inkorrekte Daten, verursacht durch Eingabe-, Mess- oder Verarbeitungsfehler,
- logisch widersprüchliche Daten,
- unvollständige, ungenaue bzw. zu grobe Daten,
- Duplikate im Datenbestand,
- uneinheitlich repräsentierte Daten,
- veraltete Daten,
- für den Verwendungszweck irrelevante Daten,
- unverständliche Daten, bedingt durch qualitativ mangelhafte Metadaten.

(Bauer und Günzel 2013, S. 42)

Das Thema Datenqualität und Datenqualitätssicherung ließe sich noch sehr stark vertiefen, soll an dieser Stelle jedoch nicht weiter ausgearbeitet werden. Es ist indes durchaus gängig, dass Data Warehouses bzw. Data-Warehouse-Systeme als Werkzeug im Data Quality Management eingesetzt werden, Data Warehouses also aus keinem anderen Grund aufgebaut werden, als die Datenqualität in den (operativen) Vorsystem zu gewährleisten.

Verfügbarkeit

Das alleinige physische Vorhandensein von Daten lässt keine Aussage über die Nutzbarkeit derselben zu. Es muss zunächst aus rechtlicher Perspektive geklärt werden, ob es zulässig ist, diese Daten zu verwenden. Dies kann insbesondere der Fall bei der Integration externer Daten (wie z. B. Adresse und Telefonnummern) der Fall sein. Darüber hinaus ist zu klären, ob der Besitzer der Daten (Data Owner, keine einzelne physische Person), z. B. ein Unternehmensbereich, gewillt ist, seine Daten für die Integration in das Data Warehouse zur Verfügung zu stellen. Bei personenbedingten Daten ist zudem häufig die Einholung einer Genehmigung seitens des Betriebsrats

erforderlich. Darüber hinaus besteht die Möglichkeit, dass besonders sensible bzw. schützenswerte Daten benötigt werden, für deren Integration schon im Vorfeld ein Zugriffs- und Berechtigungskonzept erstellt werden muss. Im Gesamtkontext des Data-Warehouse-Projekts muss aus diesen Gründen geklärt werden, ob ein Data Warehouse überhaupt die an es gestellten Anforderungen erfüllen kann, wenn eine dieser Datenquellen nicht oder nur unzureichend vorhanden ist. Neben diesen meist organisatorischen Fragestellungen gilt es zu klären, ob alle notwendigen Daten überhaupt technisch integriert werden können. So kann es z. B. vorkommen, dass ein operatives Quellsystem ein proprietäres Speicherformat verwendet, das von anderen Applikationen nicht gelesen bzw. verarbeitet werden kann. (Bauer und Günzel 2013, S. 43)

Kosten

Neben der Verfügbarkeit von Daten ist der Kostenfaktor für ihre Integration unbedingt zu beachten. Externe Daten, wie z. B. Analysen von Marktforschungsinstituten oder Nachrichtendiensten, müssen häufig teuer eingekauft werden, sodass durch diese ein für das Data-Warehouse-Projekt nicht zu vernachlässigender Kostenblock entstehen kann. Darüber hinaus sind noch weitere Kostenfaktoren vorstellbar, die möglicherweise mit neuen Datenquellen einhergehen. So gilt es, etwaig anfallende Kosten auch dann zu berücksichtigen, wenn die Quelldaten an sich kostenfrei zur Verfügung stehen, ihre Einbindung in das Data-Warehouse-System jedoch mit weiteren Aufwendungen verbunden ist. Das kann dann der Fall sein, wenn die Daten in einer Form vorliegen, die es erforderlich macht, das Data-Warehouse-System anzupassen. Denkbar wäre sowohl ein hardwaretechnischer Ausbau als auch eine softwaretechnische Erweiterung, um die neuen Daten zu integrieren. (Bauer und Günzel 2013, S. 44)

Neben den internen Daten stellen die externen Daten in einem Data Warehouse einen entscheidenden Beitrag dar. Häufig sind sie es, die in Kombination mit den internen Daten, die Möglichkeit bieten, diese kontextsensitiv auszuwerten. Die externen Daten können dabei aus einer Vielzahl heterogener Datenquellen entstammen, wie etwa Markt-, Meinungs- und Trendforschungsinstituten, politischen Informationsdiensten oder Nachrichtendiensten wie Reuters. In der letzten Dekade ist zudem eine Vielzahl von Quellen über die Verbreitung des Internets hinzugekommen, etwa soziale Netzwerke wie Facebook oder der Kurznachrichtendienst Twitter. Hierbei müssen die Daten nicht immer zwangsweise aus einer strukturierten Datenbank stammen. (H. Mucksch 2006, S. 134)

2.3.3 Extraktion (Prozess)

Um den Datenbestand eines Data Warehouse aktuell zu halten, müssen Aktualisierung der Quellsysteme inkrementell in den bestehenden Datenbestand integriert werden. Ein vollständiges Löschen inklusive anschließender gesamtheitlicher Neubeladung ist aus zweierlei Gründen nicht praktikabel. Zum einen sind die zu verarbeitenden Datenvolumina der Vorsysteme typischerweise so groß, dass dies nicht in einem akzeptablen Zeitraum zu bewerkstelligen wäre. Zum anderen würde die Historisierung der Daten im Data Warehouse auf diese Weise verloren gehen. (Bauer und Günzel 2013, S. 49)

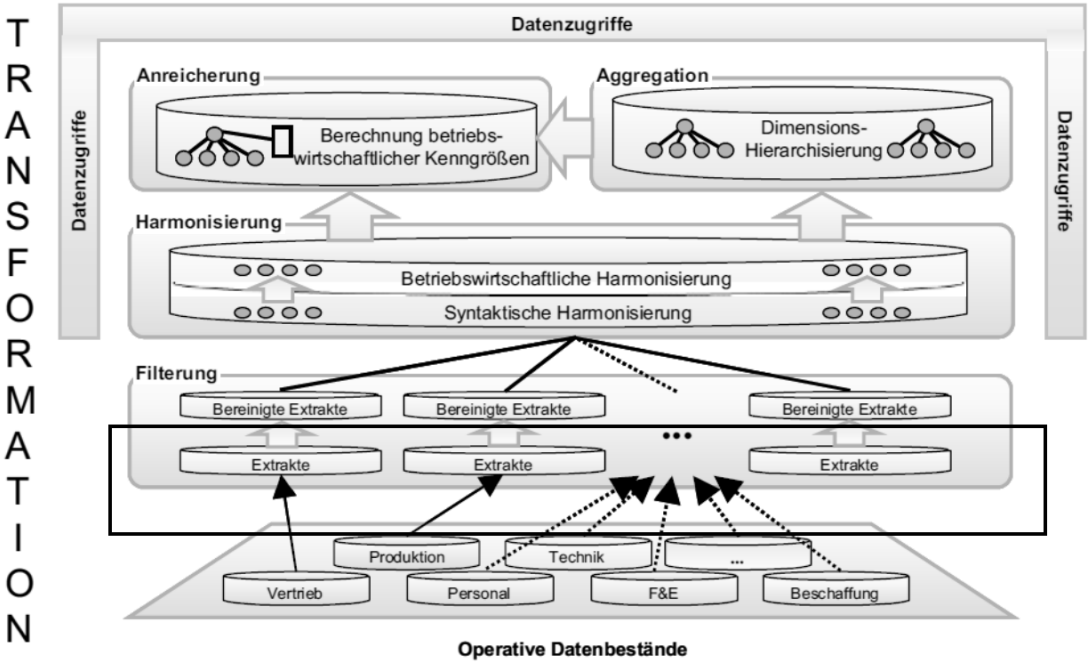

Abbildung 6: Data Warehouse – Gesamtschaubild inkl. ETL-Prozesse – Fokus auf Extraktion und Staging Area (H. Kemper: und R. Finger 2006, S. 116)

Der zweite Aspekt stellte, auch wenn nicht unmittelbar ersichtlich, das größere Hindernis in dieser Vorgehensweise dar. Wesentliche Teilaufgabe eines Data Warehouse ist die Analyse von Informationen im zeitlichen Verlauf über mehrere Jahre. Darüber hinaus werden viele operative Quellsysteme zur Performancesteigerung in regelmäßigen Abständen archiviert, sodass Altdaten gar nicht mehr im Zugriff wären. Das Data Warehouse dient hierbei als langfristiger Datenspeicher. (Bauer und Günzel 2013)

R. Gabriel weist daraufhin, dass das Verfahren der Löschung und vollständigen Neubeladung durchaus in einigen Situationen sinnhaft sein kann. Er knüpft dies zum einen an das zu übernehmende Datenvolumen und zum anderen daran, ob Datenarchivierung, -löschung oder -veränderung keine Berücksichtigung im Data Warehouse finden müssen. (Gabriel et al. 2009, S. 50)

Einen Teilbereich eines Data Warehouse in regelmäßigen Abständen vollständig zu löschen und neu zu beladen, kann dann sinnhaft sein, wenn der Implementierungsaufwand einer inkrementellen Beladung um ein Vielfaches höher ist als der einer vollständigen Beladung oder aber eine vollständige Neubeladung mit weniger Ressourcen ausgeführt werden kann. Als Beispiel kann hier eine Stammdatentabelle mit wenigen Datensätzen dienen, die bereits historisiert im Quellsystem vorliegt. Die Implementierung inkrementeller Beladungen bereits historisiert vorliegender Tabellen kann in vielen Fällen um ein Vielfaches aufwendiger sein als die Option, diese vollständig zu löschen und anschließend neu zu beladen. (Gabriel et al. 2009, S. 50)

Zur Erkennung geänderter Datensätze bestehen verschiedene Verfahren:

- Triggerverfahren: Ein Datenbankmanagementsystem, das Trigger unterstützt, protokolliert Änderungen an seinen Daten unmittelbar nach diesen in einer dedizierten Datei oder einer anderen Datenstruktur. Das verwendete Extraktionswerkzeug liest diese im Nachgang aus und verarbeitet die Daten in das Data Warehouse. Nach erfolgreicher Verarbeitung werden die entsprechenden Datensätze als solche markiert oder aus der Protokolldatei entfernt, um nicht ein weiteres Mal verarbeitet zu werden.
- Erkennung manipulierter Datensätze mittels Replikationsdiensten: Das Verfahren funktioniert ähnlich wie das Triggerverfahren. Geänderte Datentupel werden in eine spezielle Zieltabelle geschrieben. Das Extraktionswerkzeug kann anschließend die geänderten Datensätze über eine Schnittstelle wie etwa SQL aus den Replikationstabellen abfragen.
- Zeitstempel-basiertes Vorgehen: Bei dieser Variante sind Datenmanipulationen anhand eines Zeitstempels erkennbar, der im Falle einer Änderung auf den aktuellen Zeitpunkt gesetzt wird. Die Extraktionskomponente selektiert nun anhand der Zeitstempel die relevanten Datensätze.
- Log-basiertes Vorgehen: In diesem Verfahren werden die Standard-Logdateien des jeweiligen Datenbankmanagementsystems verwendet. Hierzu müssen die Logdateien im Zugriff des jeweiligen ETL-Werkzeugs sein, welches diese analysieren und interpretieren muss, um auf Basis dieser Änderungen an den Daten zu ermitteln, die in das Data Warehouse verarbeitet werden sollen. Bei diesem Verfahren findet in der Regel keine standardisierte Schnittstelle zwischen Datenquelle und Extraktionskomponente Verwendung.
- Snapshot-Variante: Das Snapshot-Vorgehen ist ähnlich wie das Log-basierte Vorgehen. In periodischen Abständen werden Snapshots der Quelldatenbank erstellt, welche den Datenbestand einer bestimmen Quelle zu einem bestimmten Zeitpunkt umfassen. Durch einen Vergleich zweier Snapshots (Delta-Berechnung) können anschließend Änderungen identifiziert werden.

(Gabriel et al. 2009, S. 50)

Welches Deltaverfahren Anwendungen finden soll, muss von Fall zu Fall entschieden werden. Beispielsweise ist das Trigger-basierte System der Erfahrung nach mit vergleichsweise wenig Implementierungsaufwand verbunden, versucht jedoch eine erhöhte Last im Quellsystem, da sämtliche Änderungen an einer weiteren Stelle protokolliert werden müssen. Im Gegensatz hierzu steht etwa das Log- oder Snapshot-basierte Verfahren, das keine erhöhte Last auf dem Quellsystem erzeugt, aber in der Anbindung an ein Data Warehouse sehr aufwendig ist. Das in der Realität wohl am häufigsten verwendete System ist das Zeitstempel-basierte System, das jedoch in der Regel den Nachteil birgt, dass gelöschte Datensätze im Quellsystem nicht als gelöscht identifiziert werden können. (Gabriel et al. 2009, S. 50)

Der Zeitpunkt, an dem die Daten übertragen werden, hängt maßgeblich von der Semantik der Daten bzw. von den durchzuführenden Auswertungen ab. (Bauer und Günzel 2013, S. 51)

Häufig werden Tages-, Wochen- oder Monatsgrößen gefordert. Um die operativen Systeme nicht zu beeinträchtigen, findet die Übertragung oftmals nachts oder am Wochenende statt. (Gabriel et al. 2009, S. 51)

Folgende Strategien kommen laut Bauer et al. bei der Extraktion prinzipiell infrage:

- periodische Extraktion, wobei die Periodendauer von der geforderten Mindestaktualität (monatlich, täglich, stündlich usw.) der Daten abhängt;
- Extraktionen auf Anfrage (Daten werden bei erstmaliger Anforderung aus dem Quellsystem extrahiert/gelesen);
- Ereignis-gesteuerte Extraktionen, z. B. bei Erreichen einer festgelegten Anzahl von Änderungen;
- sofortige Extraktion bei Änderungen.

Teilweise wird bei den Strategien auch zwischen synchronen und asynchronen Vorgehen unterschieden. Als synchron ist die sofortige Extraktion zu verstehen. Alle anderen Strategien werden als asynchron bezeichnet, da Änderungen mit einem größeren Zeitversatz übertragen werden. Mittels Extraktion gelangen die Quelldaten in den Arbeitsbereich des Data Warehouse, auch Stageing Area genannt. (Bauer und Günzel 2013, S. 51)

Im Rahmen der Planung eines Data-Warehouse-Projekts stellt die Auswahl der dauerhaft zu integrierenden Daten eine nicht triviale Frage dar. Hierfür sind mehrere Strategien denkbar.

Wesentlicher Bestandteil eines fast jeden Data-Warehouse-Projekt ist der Datenintegrationsprozess (Datenbewirtschaftung), auch ETL genannt. ETL steht für Extraktion, Transformation und Laden. Damit wird die Überführung von Daten zwischen den verschiedenen transaktionalen Informationssystemen und den Data-Warehouse-Datenhaltungskomponenten organisiert. Das Werkzeug, anhand dessen der ETL-Prozess implementiert wird, kann sowohl eine völlig eigenständige Anwendung darstellen, die autark von anderen Data-Warehouse-Komponenten agiert, als auch Teil einer Mehr-Komponenten-Lösung sein. C. Bange unterscheidet hier nach drei Zugehörigkeitsgruppen: Spezialwerkzeuge zur Datenintegration, die ausschließlich Komponenten zu Extraktion, Transformation, Qualitätssicherung und Überführung von Daten enthalten; Modulen in Business-Intelligence-Suiten als Teil einer Produktfamilie für diverse Business-Intelligence-Aufgaben; Datenbankkomponenten, die Teil des Produktangebots von Standard-Datenbanken sind. (C. Bange 2006, S. 92)

2.3.4 Staging Area (Ebene)

In der Staging Area werden die zuvor extrahierten Quelldaten temporär zwischengespeichert. Bei der Übertragung findet meist noch keine Transformation der Daten statt, sodass sich in der Persistent Staging Area (PSA) zunächst eine 1:1-Kopie der Quelldaten befindet. Durch die Entkoppelung von den Quellsystemen besteht im weiteren Verlauf des ETL-Prozesses nicht mehr das Risiko, die Quellen in ihrem laufenden Betrieb zu beeinflussen. Die gewünschten Transformationen können nun auf der Datenbasis der Staging Area durchgeführt werden. (Bauer und Günzel 2013)

Die Staging Area dient häufig als Bereich zur Zwischenspeicherung von bspw. tagesaktuellen Daten. Die extrahierten Daten werden über einen entsprechenden Zeitraum gesammelt (bspw. bis eine Woche abgeschlossen ist), um sie anschließend auf der gewünschten Aggregationsebene zu verdichten. Beispielhaft wären so keine tagesaktuellen, sondern lediglich wochenaktuelle Daten im Data Warehouse enthalten. Die Funktion dieser Komponente besteht folglich darin, einen bestimmten Zeitpunkt bzw. Stand der Quelldaten bis zu ihrer weiteren Verarbeitung festzuhalten. Für diesen Zweck könnte jedoch auch das Operational Data Storage (ODS) infrage kommen. (Bauer und Günzel 2013)

2.3.5 Transformation (Prozess)

Operative respektive transaktionale Daten sind auf die Anforderungen der mengen- und wertorientierten Abrechnungs-, Administrations- und Dispositionssysteme ausgerichtet. Sie gewährleisten einen reibungslosen Ablauf des Tagesgeschäfts. Daher stehen bei der Konzeption operativer Datenhaltungssysteme keine Anforderungen hinsichtlich der Analysefähigkeit der Systeme im Fokus. Aufgrund dessen müssen die operativen Daten, die in der Staging Area vorgehalten werden, im nächsten Schritt

derart transformiert werden, dass aus ihnen management- und entscheidungsrelevante Informationen gewonnen werden können. Diese hierfür notwendigen Prozesse werden unter dem Obergriff der Transformation zusammengefasst. Dieser setzt sich nach H. Kemper aus den Sub-Prozessen der Filterung, Harmonisierung, Aggregation und Anreicherung zusammen. (H. Kemper: und R. Finger 2006, S. 115)

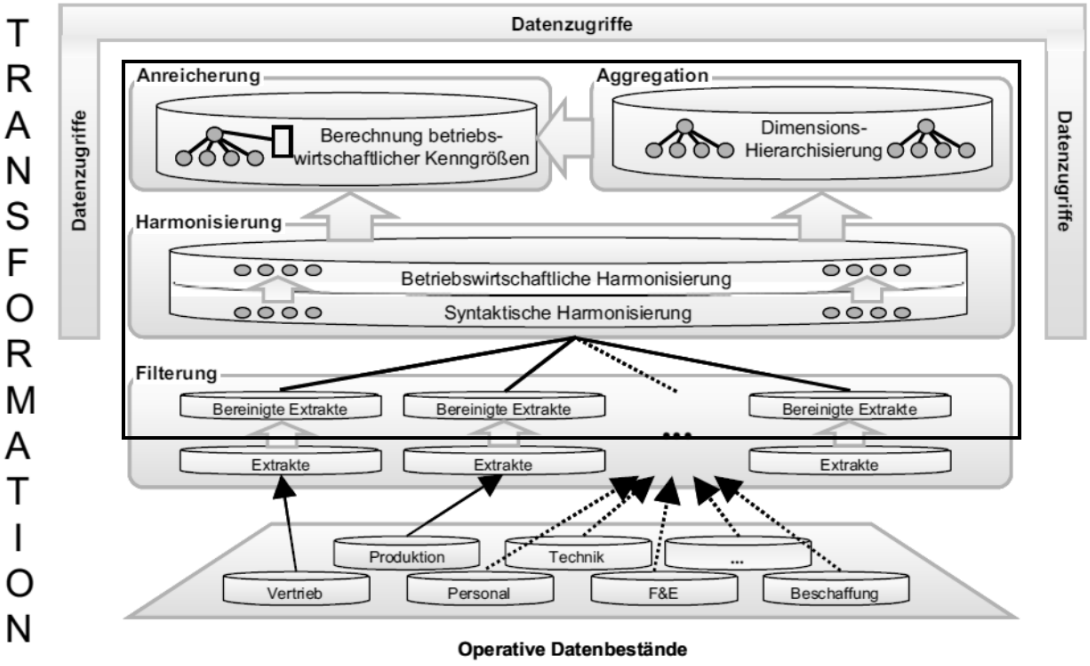

Abbildung 7: Data Warehouse – Gesamtschaubild inkl. ETL-Prozesse – Fokus auf Transformation (H. Kemper: und R. Finger 2006, S. 116)

Filterung und Bereinigung

Operative Datenbestände sind in vielen Fällen nicht frei von Defekten. Darunter fallen das Fehlen von Datenwerten oder die nicht korrekte Belegung von Datentypen. Ursache hierfür kann die syntaktische oder inhaltliche Fehleingabe eines Nutzers der operativen Systeme sein. Teile dieser Fehler können bereits bei der Extraktion behoben werden, wenn im Vorfeld Überlegungen zur erwartenden Datenqualität inklusive potenzieller Fehlerquellen durchgeführt wurden. Die folgende Tabelle veranschaulicht nach Kemper klassifizierte Fehlerarten. (H. Kemper: und R. Finger 2006)

Bereinigung	1. Klasse	2. Klasse	3. Klasse
	autom. Erkennung und autom. Korrektur, vor Extraktion	autom. Erkennung und man. Korrektur, nach Extraktion	man. Erkennung und man. Korrektur, nach Extraktion
Syntaktische Mängel	Bekannte Formatanpassungen	erkennbare Formatinkompatibilitäten	Lesen, periodisches hinzufügen
Semantische Mängel	Fehlende Datenwerte	Ausreißerwerte / unstimmige Wertekonstellationen	unerkannte semantische Fehler in opera tiven Quellen

Tabelle 4: Mängelklassifikation im Rahmen der Bereinigung (H. Kemper und R. Finger 2006, S. 120)

- 1. Klasse: Mängel, die automatisiert durch implementierte Routinen noch während der Extraktion behoben werden können. Unter einem semantischen Mangel ist eine Formatanpassung zu verstehen. Werden besondere Format-, Steuer- oder Sonderzeichen in den operativen Systemen verwendet, um bestimmte Eigenschaften kenntlich zu machen, können sie an dieser Stelle erkannt und bereinigt werden. Beispielsweise könnte aus dem Aufbau einer Bestellnummer eine Auftragsart abgeleitet werden, da je Art feste Nummernkreise definiert sind.
- 2. Klasse: Automatisch identifizierbare Mängel, deren Korrektur jedoch ein manuelles Eingreifen erfordert. Syntaktische Mängel können beispielsweise in Form bisher unberücksichtigter Syntaxvarianten zutage treten. Über Plausibilitätskontrollen sind auch semantische Mängel automatisiert erkennbar. Korrekturen der Mängel sollten stets im Quellsystem erfolgen, um diese langfristig zu beheben.
- 3. Klasse: Die manuelle Mängelerkennung beschränkt sich stets auf semantische Mängel, da syntaktische Vorgaben stets vollständig beschrieben und damit automatisch erkannt werden können. Fehler dieser Art können in der Regel nur von Fachspezialisten erkannt und behoben werden. Eine Korrektur im Quellsystem sollte schnellstmöglich erfolgen bzw. übergangsweise in der Staging Area durchzuführen.

Kemper weist darauf hin, dass die Bereinigung semantischer Mängel der Stufen 1 und 2 kurz- bis mittelfristig zu einer Behebung der Fehlerursache in den operativen Quellsystemen führt. (H. Kemper: und R. Finger 2006, S. 120)

Die Filterung/Bereinigung ist in der Literatur auch unter dem Begriff Data Cleansing bzw. Datenbereinigung bekannt. (Bauer und Günzel 2013, S. 52)

Das abgebildete Data-Warehouse-Gesamtschaubild zeichnet für die bereinigen Extrakte separate Datenspeicher aus, die in der Staging Area getrennt von den Quellsystemextrakten liegen. Diese separate Datenhaltungsschicht wird, in Anlehnung an die Staging Area, häufig als Cleansing Area bezeichnet. In vielen Data-Warehouse-Umgebungen wird jedoch auf diese separate Schicht verzichtet – entweder, weil der Bereinigungsprozess Teil des nachgelagerten Migrationsprozesses ist, oder aber,

weil die Bereinigung unmittelbar auf den Daten in der Staging Area erfolgt, die dann weiterverarbeitet werden. (Bauer und Günzel 2013, S. 52)

Datenmigration/Harmonisierung

Die Bereinigung der Daten ist nicht ausreichend, um auf dem hierdurch erzeugten Datenbestand informationsgenerierende Analysen durchzuführen. Die vorliegenden Daten kommen aus verschiedenen Quellsystemen, liegen in unterschiedlichen Strukturen vor und weisen somit eine starke Heterogenität auf. Infolge dessen muss zuvor eine themenbezogene Gruppierung der bereinigten Extrakte stattfinden. (H. Kemper: und R. Finger 2006, S. 121)

Bauer et al. sprechen hierbei von einer Standardisierung, die vorgenommen wird, um unterschiedliche Quelldaten in ein einheitliches Format zu bringen und sie so miteinander vergleichbar zu machen. In der Literatur wird dieser Prozess auch häufig als Datenmigration (engl. data migration) beschrieben.
Der Prozess der Datenmigration setzt sich dabei aus den folgenden Subprozessen zusammen:

- Anpassungen von Datentypen,
- Konvertierung von Kodierungen,
- Vereinheitlichung von Zeichenketten,
- Vereinheitlichung von Datumsangaben,
- Umrechnung von Maßeinheiten,
- Kombination bzw. Separierung von Attributwerten.

(Bauer und Günzel 2013, S. 52)

Diese recht einfache Auflistung der Teilprozesse nach Bauer et al. benötigt keine weitreichende Erklärung. Anders sieht es jedoch in der Definition der Subprozesse nach Kemper und Finger aus, welche die Datenmigration in die folgenden drei Subprozesse unterteilen: (Bauer und Günzel 2013, S. 52)

- Abstimmung von Kodierungen, Synonymen und Homonymen,
- Lösung des Problems der Schlüsseldisharmonien,
- Vereinheitlichung betriebswirtschaftlicher Begriffsabgrenzungen.

(H. Kemper: und R. Finger 2006, S. 122)

Die folgende Tabelle veranschaulicht den Prozess der Abstimmung.

	Charakteristika	Datenquelle 1	Datenquelle 2	Aktivität
Unterschiedliche Kodierung	Gleiche Attributnamen; gleiche Bedeutung; unterschiedliche Domänen	Attribut: GESCHLECHT Domäne: (0,1)	Attribut: GESCHLECHT Domäne: (M,W)	Wahl einer Domäne
Synonyme	Unterschiedliche Attributnamen; gleiche Bedeutung; gleiche Domänen	Attribut: PERSONAL Inhalt: Name der Betriebsangehörigen	Attribut: MITARBEITER Inhalt: Name der Betriebsangehörigen	Wahl eines Attributnamens
Homonyme	Gleiche Attributnamen; unterschiedliche Bedeutung; gleiche oder ungleiche Domänen	Attribut: PARTNER Inhalt: Name der Kunden	Attribut: PARTNER Inhalt: Name der Lieferanten	Wahl unterschiedlicher Attributnamen

Tabelle 5: Unterschiedliche Kodierung, Synonyme und Homonyme (H. Kemper und R. Finger 2006, S. 122)

Schlüsseldisharmonie beschreibt das Problem, dem eine uneinheitliche Schlüsselstruktur zugrunde liegt. Sollen systemübergreifende Entitäten zusammengeführt werden, ist ein gemeinsamer Primärschlüssel erforderlich. Kunden, die beispielsweise in zwei unterschiedlichen Systemen (z. B. CRM- und Abrechnungssystem) unter verschiedenen Primärschlüsseln existieren, müssen erkannt und in einem zweiten Schritt zusammengeführt werden. Da der hierfür notwendige technisch eindeutige Primärschlüssel jedoch nicht vorhanden ist, wäre die nächstliegende Option die Anpassung zugrundeliegender operativer Systeme. Die einzelnen Verfahren, die Kemper und Finger zur Auflösung dieser Disharmonien anführen, sollen an dieser Stelle nicht weiter erläutert werden, da lediglich die Sensibilisierung für die Notwendigkeit einer Harmonisierung im Fokus steht. (H. Kemper: und R. Finger 2006, S. 122)

Die Vereinheitlichung betriebswirtschaftlicher Begriffe hat zur Aufgabe, ein einheitliches, organisationsweites Verständnis des Datenbestandes herbeizuführen. Die Herausforderung dieses Schrittes ist somit nicht technischer, sondern organisatorischer Natur, da dieses in den meisten Fällen eine fachbereichsübergreifende Diskussion nach sich ziehen wird. (H. Kemper: und R. Finger 2006, S. 123)

Die beiden vorgestellten Auffassungen der notwendigen Subprozesse zur Datenharmonisierung unterscheiden sich dahin gehend, dass Prozesse nach Bauer et al. stark technischer, abstrakter Natur sind. Demgegenüber stehen die stärker prozessorientierten Tätigkeiten nach Kemper und Finger. Es ist jedoch davon auszugehen, dass die Prozesse nach Kemper und Finger die Tätigkeiten nach Bauer et al. implizieren. (H. Kemper: und R. Finger 2006, S. 123)

Nach erfolgter Filterung und Harmonisierung steht der Datenbestand des Data Warehouse für die anschließende Nutzung durch analytische Informationssysteme zur Verfügung. (H. Kemper: und R. Finger 2006, S. 123)

Aggregation/Verdichtung

In der Stufe der Aggregation kann der zuvor bereinigte Datenbestand ausgerichtet auf unternehmerische Fragestellungen vorberechnet bzw. aggregiert werden. Hierdurch wird bereits im Vorfeld der Anwendungslogik in den Daten implementiert. Die Notwendigkeit dieses Schrittes ergibt sich aus zwei Umständen. Für die Beantwortung vieler analytischer Fragestellungen wird nicht die unterste Granularität der Daten des Data Warehouse benötigt. Die Fragestellung „Wie verlief der Umsatz mit dem Kunden A in der Produktgruppe B über die Monate des vergangenen Halbjahres?“ beinhaltet die Berichtsdimensionen Kunde, Produkt sowie Zeit und verlangt deren Aggregation. Aus Gründen der Performanceoptimierung kann es sinnvoll sein, diese Werte vorab zu berechnen. Die Dimensionen Kunde, Produkt und Zeit werden daraufhin in sogenannten Dimensionstabellen gespeichert, wohingegen die aufsummierten Umsatzwerte in sogenannten Faktentabellen abgelegt werden. Der andere Umstand ergibt sich daraus, dass gewisse betriebswirtschaftliche Kennzahlen überhaupt erst dann berechenbar sind, wenn diese im Vorfeld aufsummiert wurden. Planwerte für Verkaufszahlen werden beispielsweise in Unternehmen häufig nur auf Monatsbasis erfasst. Um also eine Ist-Plan-Abweichung abbilden zu können, müssen die tagesbasierten Istwerte zunächst je Monat aufsummiert werden. (H. Kemper und R. Finger 2006, S. 124)

Oftmals sind die Berichtsdimensionen hierarchisch angeordnet. Dabei muss eine Dimension nicht zwangsweise genau einer Hierarchie zugeordnet sein. Meist existieren, in Abhängigkeit von der Sichtweise, mehrere Hierarchien. So ist ein Kundenmanager eher an Summen auf der Basis einer Kundengliederung interessiert, während ein Produktmanager Summenstrukturen nach einer Produktgliederung benötigt. Infolge dessen wird auf Basis unterschiedlicher Sichtweisen und Hierarchien eine Vielzahl von Aggregationen gebildet. (H. Kemper und R. Finger 2006, S. 124)

Anreicherung

Die Anreicherung der Daten beabsichtigt die Erweiterung der Daten um funktionale Aspekte erweitert. Ziel ist es, bewusst betriebswirtschaftliche Kennzahlen vorauszuberechnen und einer Vielzahl von Nachfrager zur Verfügung zu stellen. Dies beabsichtigt zum einen schnellere Antwortzeiten von Abfragen, da Kennzahlen nicht erst bei den Anfragen berechnet werden müssen, sondern bereits berechnet vorliegen. Zum anderen kann für die Vorberechnung ein vordefiniertes betriebswirtschaftliches Instrumentarium verwendet werden, wodurch eine einheitliche, organisationsweite Konsistenz der Daten garantiert werden kann. (H. Kemper und R. Finger 2006)

2.3.6 Laden

Nachdem die Daten transformiert wurden, müssen diese in den bestehenden Datenbestand integriert werden. Dieser Schritt erfolgt typischerweise zu einem oder mehreren festgelegten Zeitpunkten oder Ereignissen. In der Regel werden die Aktionen Transformation und Laden in einem Schritt durchgeführt, sodass die transformierten Daten nicht zunächst zwischengespeichert werden müssen, sondern direkt in die Zieltabelle geschrieben werden können. Sofern bereits bei der Extraktion berücksichtigt, werden in diesem Schritt nur neue oder geänderte Daten übertragen, sodass die benötigten Rechenressourcen für die Integration möglichst gering ausfallen. Bei der Integration der Daten muss das Historisierungskonzept des Data Warehouse berücksichtigt werden. Geänderte Datensätze dürfen nicht einfach überschrieben, sondern müssen als zusätzliche Datensätze unter Verwendung eines Änderungszeitstempels abgelegt werden. Da der Transformations- und Beladungsvorgang häufig mit einer zusätzlichen Belastung der beteiligten Systeme einhergeht, erfolgen diese in der Regel zu Zeiten außerhalb der Arbeitszeiten, wie z. B. nachts oder am Wochenende. (Bauer und Günzel 2013, S. 98)

2.3.7 ODS (Operational Data Storage)

Um die nachfolgenden Begriffe im Zusammenhang mit einer Data-Warehouse-Architektur erläutern zu können, bietet es sich an, ein anderes Schaubild einzuführen. Das bislang verwendete Schaubild nach Kemper und Finger konzentrierte sich vornehmlich auf den ETL-Prozess. Das unterhalb dargestellte Schaubild visualisiert nun hingegen ein gesamtes Data Warehouse mit dem Fokus auf die physischen Komponenten.

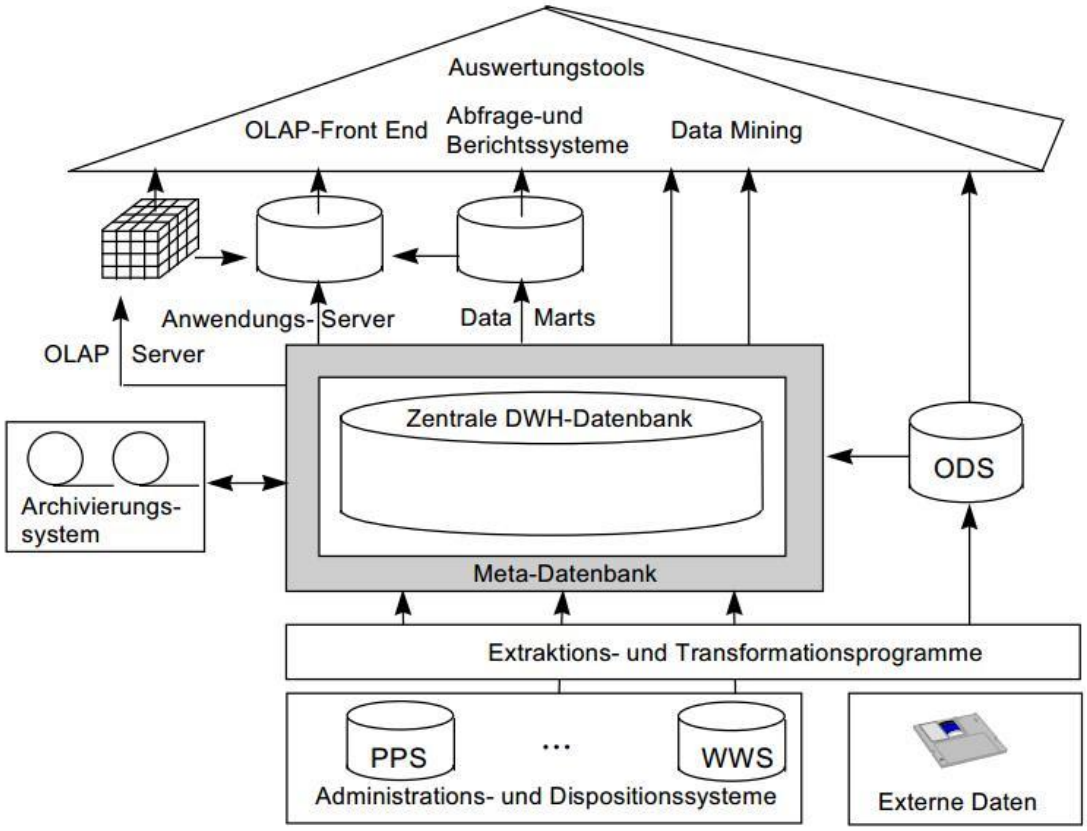

Abbildung 8: Idealtypische Data-Warehouse-Architektur (Mucksch, H., Behme, W. 2000, S. 14)

Neben der zentralen Data-Warehouse-Datenbank findet sich im obigen Schaubild ein Datenbankobjekt, das mit dem Titel ODS beschrieben ist. ODS steht für Operational Data Storage, wozu sich in der Literatur unterschiedlich Ansätze finden. H. Mucksch beschreibt das ODS als eine Datenbank, mit der die Möglichkeit geschaffen werden soll, Datenanalysen zwischen zwei ETL-Prozessen durchführen zu können. Findet beispielsweise die Datenübernahme mittels ETL in das zentrale Data Warehouse nur wöchentlich statt, so können im ODS tagesaktuelle Daten gespeichert werden, die zu Analysezwecken der operativen Unternehmensführung zur Verfügung gestellt werden können. (H. Mucksch 2006)

Es stellt sich bei dieser Verwendung eines ODS indes die Frage, weshalb diesem Problem nicht entgegengewirkt werden kann, indem die Zeitspanne zwischen zwei

ETL-Prozessen verkürzt wird. Somit muss es für den Einsatz eines ODS weitere Gründe geben als die einer verkürzten Periodendauer.

Einen alternativen Grund für den Einsatz eines ODS liefern Bauer et. al. So kann es durchaus der Fall sein, dass im Data Warehouse lediglich stark aggregiert Daten liegen, für gewisse Auswertungsanforderungen jedoch der Zugriff auf die Quelldaten benötigt wird. Diesen als operationales Reporting bezeichneten Auswertungen ließe sich folglich lediglich unter Verwendung des Data Warehouse nicht nachkommen. Die Alternative im direkten Zugriff auf die Quelldaten stellt sich in einem solchen Fall als problematisch dar, da hierbei möglicherweise mit Beeinträchtigungen der operativen Systeme zu rechnen wäre oder weil die Daten in transformierter Form benötigt werden. Hier bietet sich das ODS als Lösung an, das nur einen zeitpunktaktuellen Stand der operativen Systeme enthält und so auch bei mehrmaligem Zugriff durch das operative Reporting die Quellsysteme nicht über Gebühr belastet. (Bauer und Günzel 2013, S. 57)

Neben den Gründen für einen ODS sind sich die einzelnen Literaten dahin gehend uneins, in welcher Form die Daten im ODS abgelegt werden.

H. Mucksch plädiert zum einen dafür, die Daten im ODS bereits transformiert und verdichtet abzulegen, sodass diese den Anforderungen der Auswertungswerkzeuge entsprechen. Das ODS dient für ihn somit nicht „[…] als Ersatz-Datenbasis für operationale DV-Systeme“. Auf der anderen Seite widerspricht er sich jedoch selbst, indem er das ODS als eine Entlastung für die operativen Systeme ansieht, weil nun „[…] der größte Teil der zur Abwicklung des Tagesgeschäftes benötigen Auswertungen nicht mehr auf die operationalen Datenbestände, sondern nur auf die verdichteten Daten des ODS zugreift“. (H. Mucksch 2006, S. 136)

Bauer et al. bzw. auch Inmon gehen sehr detailliert auf den Aufbau und die Struktur eines ODS ein und gelangen so einheitlich zu dem Schluss, dass einem ODS durchaus die Rolle eines Ersatzes für operationale Datenbanken zukommen kann. Hierzu übernimmt Bauer et al. die Auffassung von Inmon, der das ODS in fünf Klassen (Klasse 0-Klasse 4) unterteilt.

- Klasse 0: Komplette Tabellen werden aus der operativen Umgebung in den ODS repliziert – es findet lediglich eine physische, aber keine logische Integration, geschweige denn eine Bereinigung statt. Ein separates Data Warehouse kann von diesem ODS mit Daten beliefert werden. Es gibt aber keine direkte Schnittstelle oder Integration zwischen ODS und Data Warehouse.
- Klasse 1: Daten der Transaktionen werden nach deren Ausführung in den operativen Systemen an das ODS übertragen und dort analog zu den Tabellen der

Klasse 0 lediglich physisch integriert. Analog zur Klasse 0 können Daten aus diesem ODS an das Data Warehouse geliefert werden. Auch hier gibt es keine direkte Schnittstelle.

- Klasse 2: Datenänderungen werden gesammelt, integriert, transformiert und in gewissen Zeitabständen in das ODS übertragen, um im Data Warehouse ausgewertet werden zu können. Die im ODS integrierten Daten werden in das Data Warehouse übertragen.
- Klasse 3: In diesem Falle wird das ODS [*ausschließlich*] durch das Data Warehouse beliefert. Die fachlichen Anforderungen entscheiden über die Relevanz der Daten, die in das ODS übertragen werden sollen.
- Klasse 4: Das ODS ist eine Kombination aus integrierten Daten aus den operativen Quellen (siehe Klasse 2) und aggregierten Daten aus dem Data Warehouse. Es findet ein bidirektionaler Austausch statt.

(Bauer und Günzel 2013, S. 56)

Als prägnantester Unterschied zwischen den beiden Auffassungen von Inmon und Mucksch fällt auf, dass nach Inmon ein ODS auch nicht transformierte Daten speichern kann. Die Daten müssen bei Inmon somit, im Gegensatz zum Ansatz von Muksch, nicht den Anforderungen der nachgelagerten Auswertungswerkzeuge entsprechen.

2.3.8 Zentrale Datenbereitstellung/Data Warehouse

Bei Betrachtung der bereits vorgestellten Abbildung der idealtypischen Data-Warehouse-Architektur wird ersichtlich, dass neben dem ODS eine zentrale Datenhaltungskomponente, die mit „Zentrale DWH-Datenbank“ beschrieben ist, vorhanden ist. In den vergangenen Abschnitten ist der Prozess des Data Warehousing über die verschiedenen Stationen beschrieben worden. Zum einen ist der Begriff ETL (Extrakt, Transform, Load) charakterisiert und erläutert worden, zum anderen sind jedoch auch Datenhaltungs- bzw. Persistierungskomponenten genannt worden. Zu den bislang vorgestellten Datenhaltungskomponenten gehören zum einen die Quellsysteme, die Staging Area sowie das ODS. Sowohl in den Prozessschritten der Transformation als auch der Beladung ist des Öfteren davon gesprochen worden, dass die prozessierten Daten in den bestehenden Datenbestand integriert werden. Bei diesem bestehenden Datenbestand kann es sich, abhängig vom Verarbeitungsschritt, um die PSA oder auch um das ODS handeln. In den meisten Fällen ist allerdings gemeint, dass die Daten in eine zentrale Datenhaltungskomponente innerhalb des Data-Warehouse-Systems geschrieben werden. Diese zentrale Stelle der Datenbereitstellung wird in der Regel als Data Warehouse bezeichnet, jedoch finden auch die Bezeichnungen Core Data Warehouse oder Enterprise Data Warehouse häufig Verwendung. Im folgenden Abschnitt soll diese zentrale Datenhaltungskomponente näher veranschaulicht werden.

Bevor mit der näheren Betrachtung begonnen werden soll, muss auf einen zentralen Aspekt hingewiesen werden. Finden sich die bislang vorgestellten DWH-Komponenten noch in fast allen populären Modellen, so unterscheiden sich die Auffassungen der einzelnen Literaten über die Funktion der zentralen Datenhaltungskomponenten sowie der diesen nachgelagerten teilweise erheblich voneinander. Ein Leser dieser Arbeit, der gut mit der Materie des Data Warehousing vertraut ist, mag somit möglicherweise über den folgenden Absatz irritiert sein, da das von ihm präferierte Modell nicht betrachtet wird.

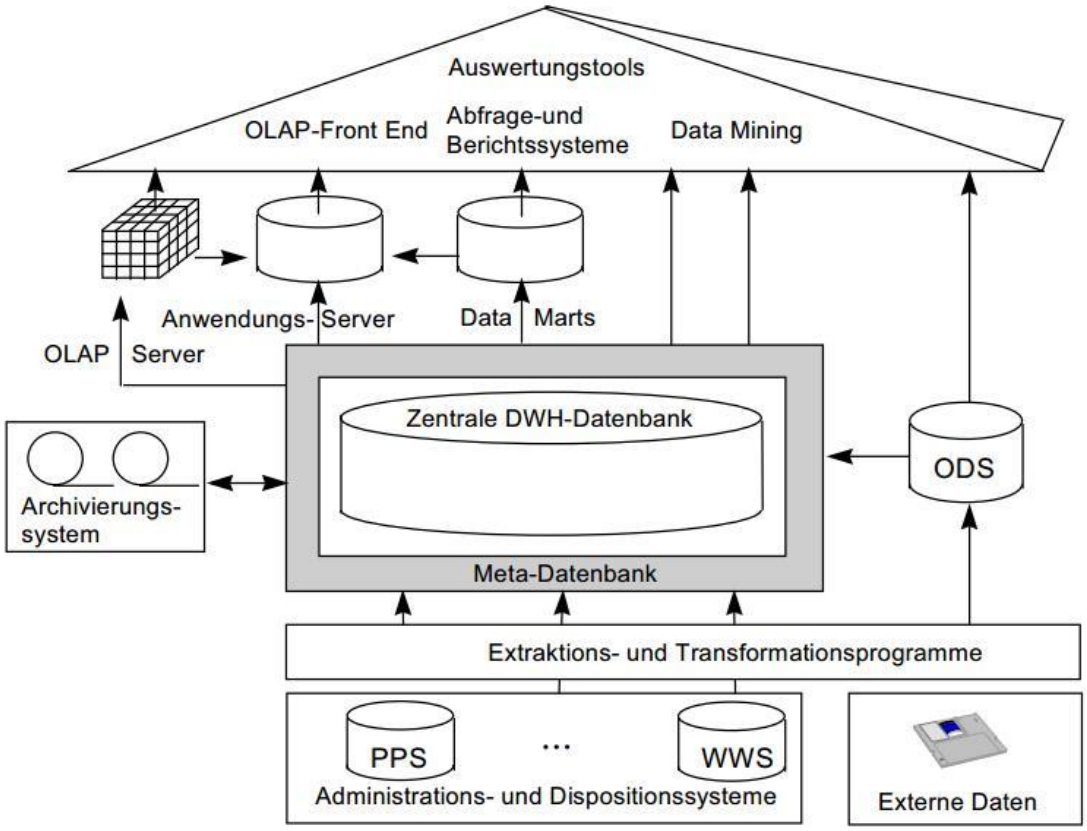

Abbildung 9: Idealtypische Data-Warehouse-Architektur (Mucksch, H., Behme, W. 2000, S. 14)

Im Modell von Bauer et al. stellt das Data Warehouse die zugrunde liegende Datenbank dar, die für Analysezwecke aufgebaut wird. Aufgabe des Data Warehouse sind die dauerhafte Verwaltung und Bereitstellung der Analysedaten. Hierbei wird von Bauer et al. besonders betont, dass die Datenmodellierung, also die Art und Weise, wie die Daten abgelegt sind, konsequent auf die Analysebedürfnisse der Anwender ausgerichtet ist. So sehen Bauer et al. im Gegensatz zu vielen anderen Modellen einen direkten Zugriff der Anwender durch Analysewerkzeuge auf das Data Warehouse vor. Diese Sichtweise basiert bei Bauer et al. auf dem Einsatz einer sogenannten Basis-Datenbank. Diese Basis-Datenbank sammelt und integriert alle notwendi-

gen Analysedaten, ohne indes auf die Modellierung und Optimierung auf eine spezifische Analyse fokussiert zu sein – sie ist vielmehr durch eine Anwendungsneutralität geprägt, sodass auch keinerlei Aggregate mit Blick auf typische OLAP-Anfragen anzutreffen sind. Im Umkehrschluss bedeutet dies, dass in der zentralen Data-Warehouse-Komponente nach dem Verständnis von Bauer et al. bereits eine Themenfokussierung und Aggregatbildung stattgefunden haben. Ob die Modellierung unter Verwendung relationaler oder multidimensionaler Strukturen umzusetzen ist, wird von Bauer et al. nicht weiter spezifiziert, den Ausführungen ist jedoch zu entnehmen, dass beide Verfahren als sinnvoll erachtet werden. (Bauer und Günzel 2013, S. 54)

Im Gegensatz zu Bauer et al. enthält das Modell von Bill Inmon keine solche Basis-Datenbank. Wie bereits in dem vorgestellten Modell sieht Inmon eine strenge Einhaltung der dritten Normalform im zentralen Data Warehouse vor. Hierdurch nimmt das zentrale Data Warehouse in dem Modell von Inmon die Aufgabe der Basis-Datenbank ein, wie Bauer et al. sie in ihrem Modell vorsehen. Hierdurch ergeben sich zwei Punkte. Zum einen findet laut Inmon keine direkte Auswertung der Data-Warehouse-Daten statt, da diese nicht Analyse-optimiert vorliegen. Zum anderen bedeutet dies, dass in dem Modell von Inmon eine weitere Datenhaltungsschicht existieren muss, die diese Aufgabe erfüllt. Inmon sieht für diese Aufgabe den Einsatz von Data Marts vor. Wie der obigen Abbildung einer idealtypischen Data-Warehouse-Architektur zu entnehmen ist, fungieren diese Data Marts als dedizierte Datenhaltungskomponenten oberhalb des zentralen Data Warehouse. Innerhalb dieser werden die Analysedaten multidimensional, aggregiert, analyseoptimiert und themenorientiert abgelegt, sodass die Anwender mit Analysewerkzeugen auf diese zugreifen können. (Inmon 2005, S. 36)

Das Modell, welches demjenigen von Inmon am häufigsten gegenübergestellt wird, ist das von Ralph Kimball. Kimball propagiert in seinem Data Warehouse Toolkit, Data-Warehouse-Architekturen stets nach einem multidimensionalen Modell aufzubauen, und verzichtet sogar auf eine Trennung von Datenhaltung und zentraler Datenbereitstellung, wie Bauer oder Inmon es in ihren Modellen handhaben. (Kimball und Ross 2013)

In den vorhergehenden Absätzen sind wiederholt die Begriffe relationales und multidimensionales Modell genannt worden, ohne näher auf diese Begriffe einzugehen. Da die Modelle von Inmon und Kimball im weiteren Verlauf der Arbeit noch weiter vorgestellt werden, sollen diese beiden Modellierungsansätze im Folgenden kurz veranschaulicht werden.

<u>Das relationale Modell</u>

Der bekannteste Vertreter des relationalen Modellierungsansatzes ist Bill Inmon, wenngleich sich die von ihm verwendeten technischen Methoden und Ansätze in diversen anderen populären Modellen wiederfinden. Beispielsweise kommt die in dem

Modell von Bauer et al. vorhandene Basis-Datenbank dem Ansatz eines Data Warehouse nach Bill Inmon sehr nahe. Wichtig ist allerdings, dass Inmon in seinem Modell keinen direkten analytischen Zugriff auf das Data Warehouse vorsieht, sondern diese Funktion für die nachgelagerten Data Marts vorsieht.

Der relationale Modellierungsansatz ist hierbei vergleichsweise einfach zu erklären, indem darauf verwiesen wird, dass er sich nach den Regeln der Datenbanknormalisierung bzw. nach Einhaltung der dritten Normalform richtet.

Laut Inmon sprechen die folgenden Punkte gegen die Verwendung eines relationalen Modells im Rahmen der Datenbereitstellung, aber für eine Verwendung als Modell für die der Datenbereitstellung vorhergehende Datenhaltungskomponente. Im Sprachgebrauch von Inmon wären Ersteres Data Marts und Letzteres das zentrale Data Warehouse.

- Das relationale Modell ist nicht auf eine bestimmte Prozessanforderung hin optimiert. Im Rahmen der themenorientierten Datenanalyse lieferte es somit keine besonders hohe Performanz im Rahmen der Datenbereitstellung. Es ist folglich nicht für den direkten Zugriff mittels Analysewerkzeugen geschaffen.
- Die relationale Struktur ermöglicht es, einen hohen Detailgrad der Daten zu erhalten, wodurch ein hohes Abstraktionslevel zugrunde liegt. Dieses kann die Grundlage für viele verschiedene prozedurale Anforderungen bilden, wodurch flexibel viele unterschiedliche Sichten auf Basis eines einheitlichen Datenpools abgeleitet werden können.
- Für eine performante Auswertung bzw. Analyse müssen die Daten zunächst aus dem relationalen Modell extrahiert und in eine neue, themenorientierte Struktur überführt werden. (Inmon 2005, S. 362)

Im Gegensatz zu dem, was die Kapitelüberschrift vermuten ließe, ist die zentrale Datenbereitstellungskomponente im relationalen Modell nach Inmon nicht für eine direkte Datenbereitstellung geeignet. Da in der Literatur jedoch häufig eine direkte Gegenüberstellung der Modelle von Inmon und Kimball stattfindet, der auf dem häufig verwendeten multidimensionalen Ansatz beruht, ist es notwendig, diesem den relationalen Ansatz gegenüberzustellen. Bei oberflächlicher Betrachtung der beiden Modelle zeigt sich indes schnell, dass eine solche Gegenüberstellung nicht sinnvoll ist, da beide Modelle grundlegend andere Ansätze verfolgen. Zum besseren Verständnis folgt aus diesem Grund im nachstehenden Absatz eine kurze Einführung in die multidimensionale Datenmodellierung.

<u>Das multidimensionale Modell</u>

Der multidimensionale Modellierungsansatz wird häufig auch als der Kimball-Ansatz nach Ralph Kimball bezeichnet. Der Ansatz ist in Bezug auf die zentrale Datenbereitstellungskomponente eines Data-Warehouse-Systems auf die optimale Bereitstellungs- und Analysefähigkeit der Daten ausgerichtet und bildet somit auf dieser Ebene

einer Data-Warehouse-Architektur den genauen Gegenpart zum relationalen Ansatz nach Inmon. (Inmon 2005, S. 357)

Die Stärken dieses Ansatzes im Rahmen der Datenbereitstellung basieren auf den folgenden Punkten:

– Das Modell ist auf die speziellen Anforderungen einiger bestimmter Fachbereiche bzw. Nutzer ausgerichtet. Es kann somit exakt auf diesen Anforderungen hin optimiert werden.

– Bedingt durch die themenspezifische Optimierung, bietet es eine hohe Analyseperformanz und einen schnellen sowie effizienten Zugriff. Hierdurch wird die Möglichkeit erzeugt, Analysen direkt auf diesen Strukturen ausführen zu können. Eine weitere Transformation der Daten in eine nachgelagerte Komponente kann damit hinfällig sein.

Aus dem Ansatz resultieren jedoch auch Nachteile, die betrachtet werden müssen:

– Mit jeder weiteren konträren Anforderung, die in das Modell einfließt, verringert sich der Grad der Optimierung.
– Eine flexible Anpassung des Modells an divergierende Anforderungen ist somit nur schwer und unter Einschränkungen durchführbar. Eine dieser Einschränkungen resultiert z. B. daraus, Daten teilweise mehrfach redundant in verschiedenen Strukturen vorhalten zu müssen. Es liegt mithin nur eine sehr einseitige Optimierung vor. (Lehner 2003, S. 362)

Multidimensionale Datenmodellierung spiegelt sich in der Verwendung eines wiederkehrenden Schemas wider. Dieses Schema ist als das sogenannte Star-Schema bzw. in einer leicht modifizierten Version als Snowflake-Schema bekannt. Wesentlicher Bestandteil beider Schemata sind sogenannte Dimensions- und Faktentabellen. Faktentabellen enthalten hierbei klassischerweise Kennzahlen oder Ergebniswerte. In transaktionalen Systemen bilden die Bewegungsdatensätze den Gegenspieler zu den Faktendatensätzen im Bereich des Data Warehousing. Faktendatensätze sind somit Werte, die beispielsweise im laufenden Geschäftsbetrieb einer Unternehmung entstehen. Typische Beispiele hierfür wären Einnahmen, Ausgaben, Umsatz, Gewinn, aber auch Kosten und Verbräuche. Da diese Kennzahlwerte jedoch erst in einem weiter gefassten Kontext sinnvolle Analysen zulassen, müssen diese mit weiteren Daten angereichert werden. Bei der Auswertung des erzielten Umsatzes ist es so beispielsweise von Interesse, mit welchen Produkten und in welchem zeitlichen Horizont der Umsatz generiert wurde. Diese weiteren Informationen werden über die Dimensionstabellen in dem Modell abgebildet. Schlüsselbeziehungen zwischen Fakten- und Dimensionstabellen ermöglichen die Verknüpfung von Kennzahlen und Stammdaten. Der Begriff Star-Modell geht auf die Struktur zurück, nach der die Dimensionstabellen um die zentrale Faktentabelle angeordnet sind. (Inmon 2005, S. 357)

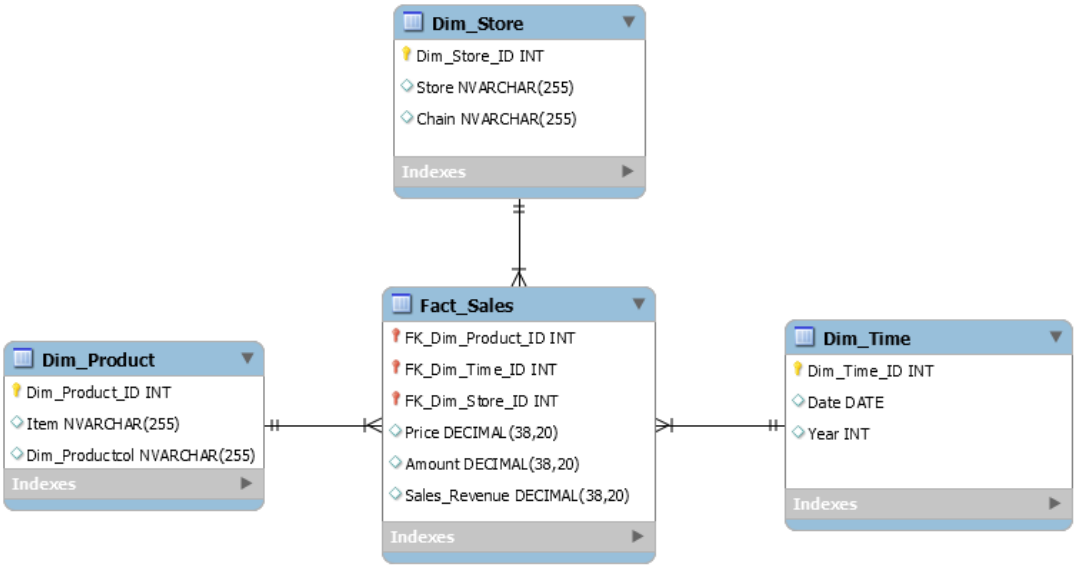

Abbildung 10: Multidimensionales Star-Schema nach Kimball

Eine Variante des Star-Schemas stellt das Snowflake-Schema dar. Kernkomponente des Snowflake-Schemas ist wie beim Star-Schema die Anordnung mehrerer Dimensionstabellen um eine Faktentabelle. Im Gegensatz zum Star-Schema liegen diese Dimensionstabellen jedoch normalisiert vor. Durch die weitere Verzweigung der Dimensionstabellen sieht es so aus, als bilde ein solches Modell eine Schneeflocke. Durch die feinere Strukturierung der Daten sind die Dimensionsdaten zwar weniger redundant als in einem Star-Schema, allerdings kann die zusätzliche Komplexität zulasten der Abfrageperformance gehen, da für die gleiche Abfrage mehr Join-Operatoren notwendig sein können. (IBM Analytics)

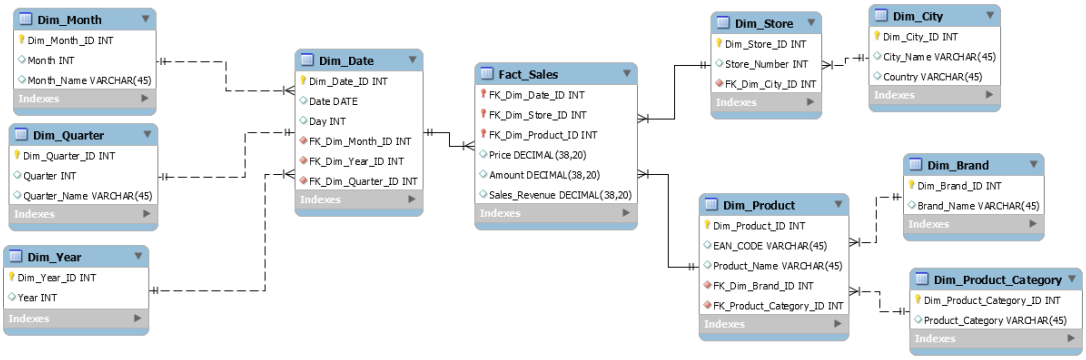

Abbildung 11: Multidimensionales Snowflake-Schema nach Kimball

Da Geschäftsprozesse in Unternehmen in der Regel sehr komplex sind, werden für die Abbildung derselben mehr als eine Kenngröße benötigt. Zur Abbildung solcher komplexer Geschäftsprozesse in einem Data Warehouse sind somit mehrere Faktentabellen mit jeweils unterschiedlichen Dimensionstabellen vonnöten. Hieraus entstehen mehrere Stern-Schemata, die über mehrfach verwendete Dimensionstabellen zusammengeführt werden können. Als Beispiel können Einkäufe und Verkäufe in einer

Filiale dienen. Diese beiden getrennten Faktentabellen können beispielsweise über die Dimensionen Filiale, Artikel und auch Zeit zu einem Datenmodell zusammengeführt werden. Ein solch konsolidiertes Modell mehrerer Stern- bzw. Snowflake-Schemata wird als Galaxy-Schema bezeichnet. Die nachfolgende Abbildung visualisiert ein solches Galaxy-Schema anhand des beschriebenen Beispiels. (Antje Höll 2005, S. 6)

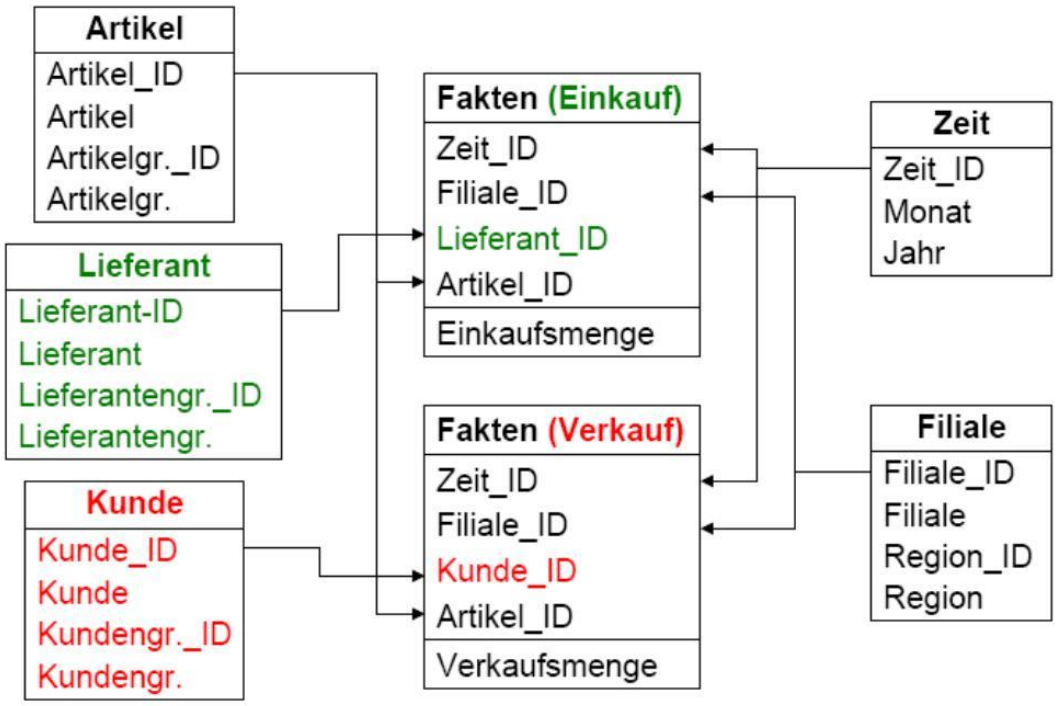

Abbildung 12: Beispiel für ein Galaxy-Schema (Antje Höll 2005, S. 6)

Die bislang vorgestellten multidimensionalen Strukturen können neben klassischen relationalen Datenbankmanagementsystemen auch in hierfür spezialisierten Systemen abgebildet werden. Diese Systeme werden als OLAP-Systeme bezeichnet und zeichnen sich dadurch aus, die für das Reporting geforderte Multidimensionalität über sogenannte Würfel (eng. Cubes) abbilden zu können. Sie sind der spezialisierten Datenbereitstellung zuzuordnen und werden daher kurz im folgenden Abschnitt erwähnt, ihr genauer Aufbau und Funktionsweise jedoch nicht weiter erläutert.

Für Bill Inmon ist Multidimensionalität innerhalb eines Data-Warehouse-Systems unabdingbar. Relationale und multidimensionale Strukturen können auf sinnvolle Weise miteinander koexistieren, indem aus den feingliedrigen Daten der Datenhaltungskomponente, abgebildet mit einem RDBMS, ein oder mehrere multidimensionale Strukturen beliefert werden. Inmon weist an dieser Stelle nochmals darauf hin, dass die Daten in seinem DWH-Modell in einem sehr hohen Detailgrad in dritter Normalform abgelegt werden. Auf Basis dieser detaillierten Daten können anschließend Aggregate

gebildet werden, die einem multidimensionalen DBMS bereitgestellt werden. In diesen multidimensionalen Strukturen, auf die anschließend mit Reportingwerkzeugen zugegriffen wird, finden sich somit laut Inmon im Normalfall keine detaillierten Daten. (Inmon 2005, S. 176)

Dem gegenüber steht wiederum die Architektur nach Kimball. Dessen zentrale Datenbereitstellung erfolgt über eine Presentation-Area. Da er eine dedizierte Datenhaltungskomponente nicht für zwingend notwendig erachtet, schreibt er seiner Presentation-Area alle wesentlichen Aufgaben zu. So werden die Daten dort aus seiner Sicht organisiert, gespeichert/gehalten und gleichzeitig für den direkten Zugriff durch Nutzer als auch für den indirekten Zugriff, z. B. durch Analyseanwendungen, bereitgestellt. Als technische Basis hierfür fordert Kimball ausschließlich multidimensionale Strukturen, wahlweise das Star-Schema oder den Einsatz von OLAP-Cubes. Parallel dazu setzt er voraus, dass die Daten in einer möglichst detaillierten Form zur Verfügung stehen. Dem Nutzer muss aus Kimballs Sicht bei seiner Auswertung Zugriff auf die unterste, detaillierteste Ebene ermöglicht werden. Für Kimball ist es nicht akzeptabel, nur hoch aggregierte Daten in multidimensionalen Strukturen abzulegen und Detaildaten nur über ein darunterliegendes normalisiertes Modell bereitzustellen, welches jedoch nicht für den Zugriff mit Analysewerkzeugen optimiert ist. (Kimball und Ross 2013, S. 21)
In den Standwerken von Inmon und Kimball ließe sich noch eine Vielzahl von Argumenten finden, die für den Einsatz und für die Ablehnung des anderen Modells sprechen, jedoch würde dies das Ziel der Arbeit verfehlen. Dennoch soll der nach Inmon gewichtigste Grund, nach dem ein multidimensionales DBMS allein ungeeignet ist, um die Anforderungen der Datenhaltung sowie die der Datenbereitstellung abzudecken, zuletzt genannt werden. Dieser liegt in den unterschiedlichen Anforderungen verschiedener Nutzer bzw. Fachbereiche. Um diesen gerecht zu werden, müsste in einem Unternehmen jede involvierte Abteilung ein eigenes multidimensionales DBMS zugewiesen bekommen. Für all diese Systeme gäbe es keine gemeinsame Datenbasis. Jedes der multidimensionalen DBMS würde somit direkt auf die von ihm benötigten Quellsysteme zugreifen, was mit vielerlei Nachteilen verbunden wäre. Eine zentrale Datenbasis würde hier als Versorgungsinstanz dienen, die in der Lage ist, viele einzelne Übertragungswege zu bündeln.

Die folgende Grafik visualisiert den Aufbau eines solchen Modells, bei dem auf eine dedizierte Datenbereitstellungskomponente verzichtet wird. (Inmon 2005, S. 178)

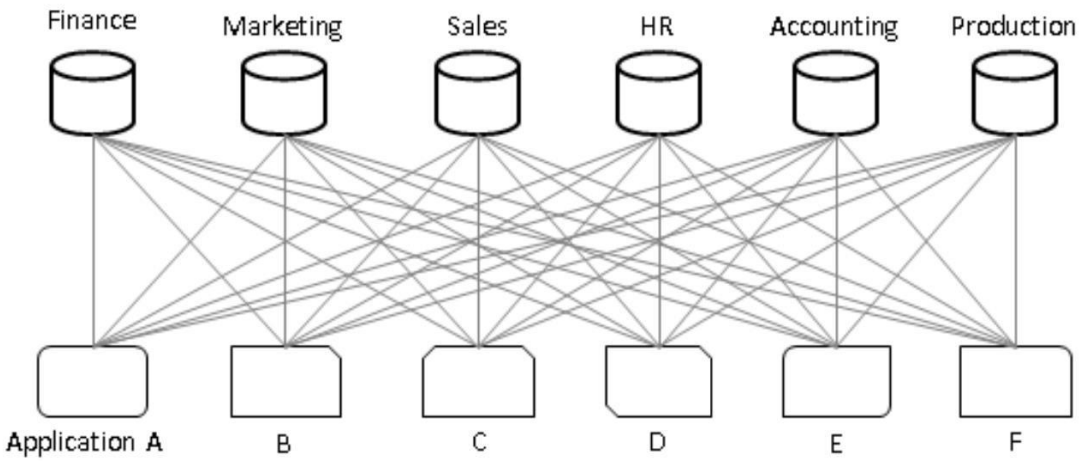

Abbildung 13: Problematik eines multidimensionalen Modells ohne dedizierte Datenbereitstellung (Inmon 2005, S. 180)

Beim Vergleich der Modelle nach Kimball und Inmon scheint dies wie so oft eine Schwarz-Weiß-Betrachtung zu sein. Ein Blick in die Praxis zeigt jedoch, dass die Realität eben nicht nur schwarz oder weiß, sondern häufig grau ist. Bei Betrachtung von Ansätzen anderer Literaten, die häufig auf Kimball und oder Inmon referenzieren, wird ersichtlich, dass zwischen diesen beiden Ansätzen Platz für Alternativen oder Kompromisse ist. Aus diesem Grund wurde einige Absätze zuvor auch kurz das Galaxy-Schema angesprochen, das die Wiederverwendung von Dimensionen bezogen auf unterschiedliche Fakten vorsieht. Auffällig ist indes, dass mit Ausnahme gänzlich alle bislang genannten Literaten wie Mucksch, Gabriel und Gluchowski, Lehner sowie Bauer et al. den Einsatz einer relationalen Struktur für eine zentrale Datenbereitstellung vorsehen. Bei Betrachtung der zu Anfang des Abschnitts abgebildeten idealtypischen Data-Warehouse-Architektur wird ersichtlich, dass oberhalb der zentralen DWH-Datenbank weitere spezialisierte Datenbereitstellungskomponenten vorgesehen sind. Diese finden sich unter anderem auch bei Muksch, Lehner, Bauer et al. sowie Inmon, die in ihren Referenzarchitekturen als Data-Mart bezeichnete Komponenten einsetzen. Auf gleicher Ebene sind bei einigen Architekturen darüber hinaus noch weitere Komponenten zu finden, wie beispielsweise OLAP-Server oder Anwendungsserver. Diese Komponenten unterstützen die Datenanalyse, indem sie die Daten in spezialisierter Form bereitstellen. Diese spezialisierte Datenbereitstellung wird im folgenden Abschnitt kurz vorgestellt.

2.3.9 Spezialisierte Datenbereitstellung/Data Mart

Neben der zentralen Datenbereitstellungskomponente sehen viele Architekturen weitere spezialisierte Komponenten zum Zugriff zwecks Analyse auf die Daten vor.

Lehner positioniert nach seinem Bereich zur Datenbereitstellung einen Bereich zur Datenanalyse. Dieser umfasst „[…] alle Systeme und spezielle Datenbasen, die zur konkreten Interaktion mit dem Benutzer und Exploration des Datenmaterials dienen“.

Diese Speichersysteme werden häufig auch unter dem Terminus der Data Marts zusammengefasst. Ziel solcher Data Marts ist die vollständige Ausrichtung auf die jeweiligen Anwendungsgebiete und Zielsysteme. (Lehner 2003)

Inmon setzt ebenfalls eine Komponente in seiner Architektur ein, die er als Data Mart bezeichnet. Bis dahin findet in seinem Modell keine Datenbereitstellung statt. Inmon nennt an dieser Stelle zwei Typen von multidimensionalen Systemen. Neben den bereits häufiger genannten klassischen relationalen Systemen führt er ebenfalls die sogenannten OLAP-Cubes an, die bereits weiter oben kurz Erwähnung fanden. Sie können über einen OLAP-Server implementiert werden, der typischerweise mit der Abfragesprache MDX angesprochen. Inmon stellt für den Vergleich dieser beiden Systeme die folgende Pro- und Kontra-Liste auf. (Inmon 2005, S. 175)

	Relationale basis for multidimensional structure	Cube
Strengths	Can support a lot of data Can support dynamic joining of data Has proven technology Is capable of supporting general-purpose update processing If there is no knowns pattern of usage of data, then the relational structure is as good as any other	Performance that is optimal for DSS processing Can be optimized for very fast access of data If pattern of access of data is known, then the structure of data can be optimized Can easily be sliced an diced Can be examined in many ways
Weaknesses	Has performance that is less than optimal Cannot be purley optimized for access processing	Cannot handle nearly as much data as a standard relational format Does not support general-purpose update processing May take a long time to load If access is desired on a path not supported by the design of the data, the structure is not flexible Questionable support for dynamic joins of data

Tabelle 6: Stärken und Schwächen multidimensionaler Systeme nach Inmon (Inmon 2005, S. 177)

Auch Bauer et al. sehen den Einsatz von Data Marts vor. Er beschreibt das Data-Mart-Konzept als eine Idee, einen „[…] weiteren inhaltlich beschränkten Fokus des Unternehmens oder einer Abteilung als Teilsicht eines Data Warehouse abzubilden“, und führt dafür die folgenden Gründe an: (Bauer und Günzel 2013, S. 62)

– Eigenständigkeit,
– bessere Zugriffskontrolle durch Teilsicht auf Daten,
– organisatorische Aspekte (z. B. Unabhängigkeit von Abteilungen),
– Verringerung des Datenvolumens,
– Performanzgewinn durch Aggregation,
– Lastverteilung auf unterschiedliche Systeme,
– Unabhängigkeit von den Aktualisierungszyklen des Data Warehouse.

(Bauer und Günzel 2013, S. 62)

3 Big Data

3.1 Begriffsherkunft

In der Einleitung der vorliegenden Arbeit wurde bereits auf den Begriff Big Data und die damit verbundenen Probleme eingegangen. Im Kontext dieser Einleitung wurde darauf hingewiesen, dass nicht versucht werden soll, den Begriff Big Data im Rahmen dieser Abhandlung zu definieren. Gleichwohl soll im folgenden Absatz nochmals der Begriff Big Data aufgegriffen werden, jedoch nicht mit dem Ziel, diesen zu definieren, sondern dessen Herkunft zu erläutern. Hierzu wird zunächst kurz auf die historische Entwicklung von IT-Systemen und Datenmengen in den vergangenen Dekaden eingegangen. Anschließend werden die vier Facetten des Begriffs beschrieben, die mit ein Grund dafür sein könnten, weshalb es so viele verschiedene Auffassungen und Interpretationen des Begriffs gibt.

Die folgenden Absätze sind in Teilen aus einer bereits zur Vorbereitung dieser Arbeit eingereichten Projektarbeit mit dem Titel „Prototypische Implementierung eines Apache-Hadoop-Clusters auf Basis von ARM Kleinstrechnern“ entnommen, die sich verstärkt mit der technischen Installation einer Apache-Umgebung befasst. Die vollständige Arbeit kann dem Anhang entnommen werden.

Über die Entwicklung der Rechenkapazitäten hat Gordon Moore, ein Mitbegründer von Intel, in den 1970er-Jahren die Aussage getroffen, dass sich diese alle 18 Monate verdoppelt. Diese Aussage ist in der Informationstechnologie als das Moorsche Gesetz bekannt geworden. (ITWissen 2007)

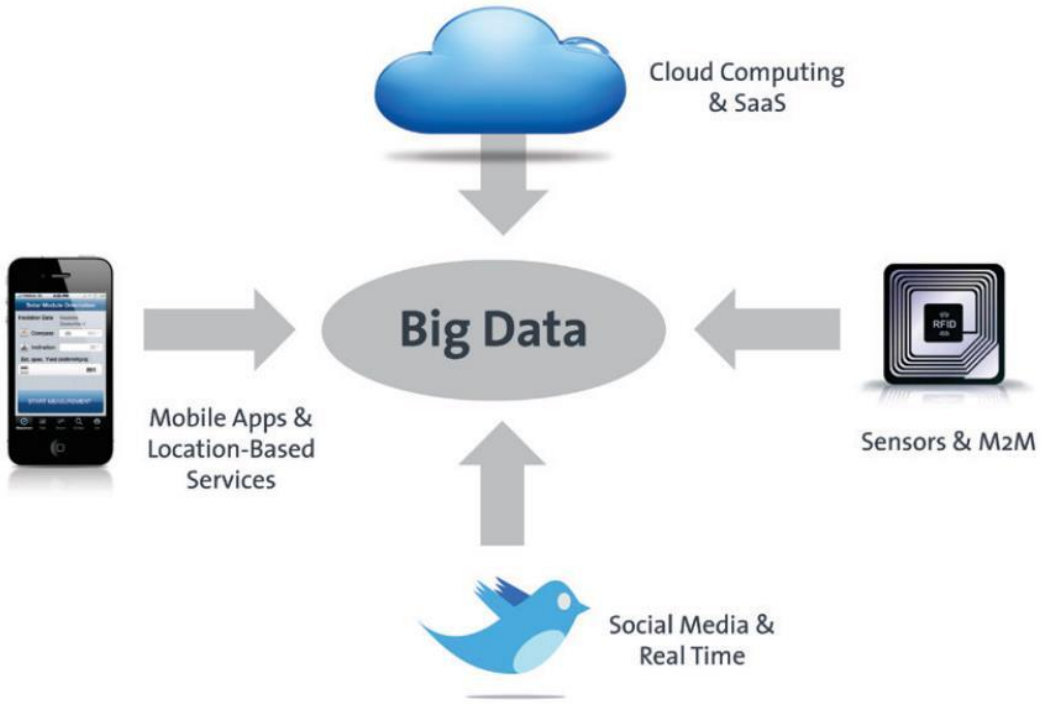

Abbildung 14: Welche Informationstechnologien das Big-Data-Phänomen entstehen lassen (BITKOM, S. 11)

Das digitale Informationszeitalter ist mit der Entwicklung und der Verbreitung von Fernsehen, Computern und dem Internet in der Mitte des vergangenen Jahrhunderts gestartet. Seitdem werden kontinuierlich digitale Daten erzeugt, verarbeitet und gespeichert. Wird diese Datenmenge jedoch mit der Datenmenge vergleichen, die in den vergangenen 15 Jahren entstanden ist, erscheint diese sehr gering und in der ganzheitlichen Betrachtung vernachlässigbar. Der Grund hierfür ist die zunehmende Verbreitung von Technologien wie RFID, Ambient Intelligence, Smartphones und der verstärkten und zunehmenden Nutzung von Social-Media-Diensten wie Blogs, Foren, Facebook und Twitter. (BITKOM, S. 12)

Beim Vergleich der Wachstumsraten, der Datenbestände und der Verarbeitung dieser verfügbaren Kapazitäten wird ersichtlich, dass sich Erstere deutlich schneller entwickeln, sodass im Jahr 2020 das weltweite Datenvolumen auf über 100 Zettabytes angestiegen sein wird. (BITKOM, S. 12)

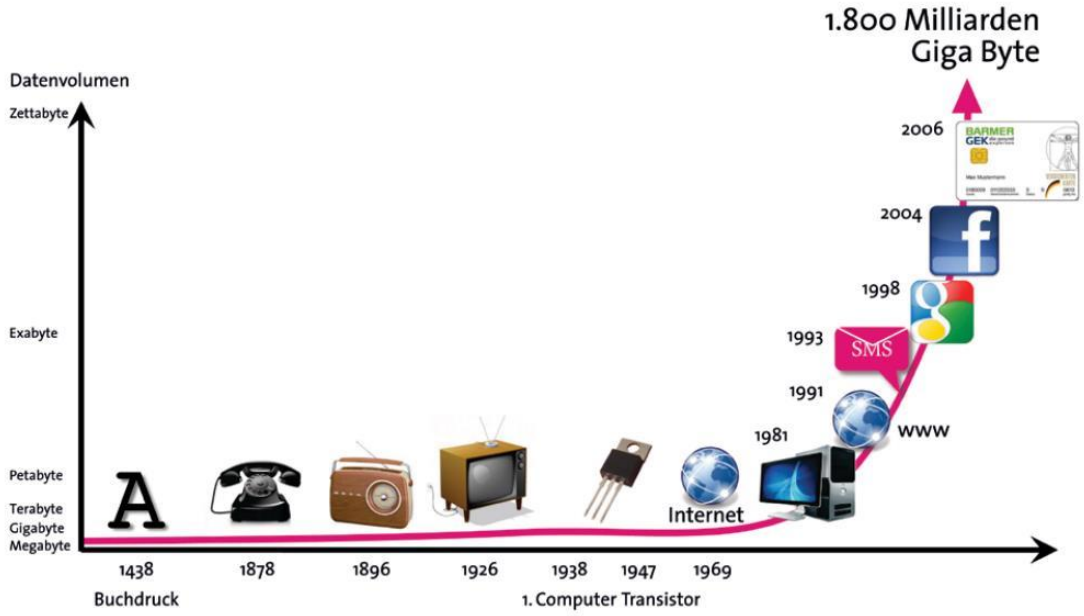

Abbildung 15: Wachstum der Datenmengen über die Zeit (BITKOM, S. 12)

Die obige Abbildung beschreibt einen weiteren Aspekt in der Abgrenzung des Big-Data-Begriffs. Neben der reinen statistischen Größe der zu betrachtenden Daten geht auch eine Verbreiterung der Datenvielfalt einher. Die Daten stammen nicht nur aus klassischen, transaktionalen Systemen, aus denen sie in strukturierter Form exportiert werden, sondern liegen zu 85 Prozent unstrukturiert vor. Mögliche Datenlieferanten sind z. B. Internettransaktionen, Social Networking, Machine-to-Machine-Lösungen, Sensoren, mobile Endgeräte oder Messungen jeglicher Art.

Die Auswertung dieser potenziell wertvollen Daten war jedoch bislang mit herkömmlichen Technologien nicht zu einem vertretbaren Aufwand zu realisieren, weshalb sie in der Entscheidungsfindung von Unternehmen oder Organisationen keine Rolle spielten. (BITKOM, S. 12)

Ein Ansatz, den Begriff und die Intention hinter Big Data möglichst treffend zu beschreiben, könnte der folgende sein.

Big Data bezeichnet den Einsatz großer Datenmengen aus vielfältigen Quellen mit einer hohen Verarbeitungsgeschwindigkeit zur Generierung eines monetären Nutzens. (BITKOM 2014, S. 12)

Hinsichtlich dieser Abgrenzung umfasst der Big-Data-Begriff die vier folgenden wesentlichen Facetten:

<u>Datenmenge (Volume):</u> Viele Unternehmen und Organisationen verfügen in der heutigen Zeit über Datenbestände, die von Terrabytes bis hin zu Größenordnungen von Petabytes reichen.

Datenvielfalt (Variety): Daten von Unternehmen und Organisationen entstammen einer Vielfalt von Quellen und liegen in den unterschiedlichsten Formaten vor. Die Daten lassen sich in grob unstrukturiert, semistrukturiert und strukturiert gruppieren, was unter dem Begriff von polystrukturierten Daten beschrieben wird. Klassische unternehmensinterne Daten werden durch externe Daten, wie z. B. Daten aus sozialen Netzwerken, ergänzt.

Geschwindigkeit (Velocity): Die Datenbestände von Unternehmen und Organisationen müssen immer schneller – bis hin zur Echtzeit – ausgewertet werden. Die Geschwindigkeit der verarbeitenden Systeme hat mit dem Wachstum der Datenbestände Schritt zu halten, wodurch sich eine Vielzahl von Herausforderungen ergibt. Große Datenbestände sollen in Sekunden analysiert und generierte Daten müssen in hoher Geschwindigkeit übertragen sowie verarbeitet werden.

Analytics: Analytics umfasst die Methoden und Verfahren zur möglichst automatischen Erkennung von Mustern, Zusammenhängen und Bedeutungen innerhalb der Daten. Hierbei werden unter anderem statistische Verfahren, Vorhersagemodelle, Optimierungsalgorithmen, Data Mining sowie Text- und Bildanalytik eingesetzt. Bestehende Analyseverfahren werden durch diese Entwicklung beträchtlich erweitert. (BITKOM 2014, S. 12)

Big Data ist eines der am höchsten priorisierten Themen innerhalb der Wirtschaft, der öffentlichen Verwaltung und der Politik. Zum einen wird dies daraus ersichtlich, dass alle führenden Anbieter von Unternehmenssoftware in dieses Marktsegment drängen und versuchen, ihr Produkt zu etablieren. Zum anderen offenbart sich dies jedoch auch dadurch, dass Big Data, wenn auch nicht unter diesem Namen, bereits heute in der Öffentlichkeit Beachtung findet. In den vergangenen Jahren wurde in den Medien häufig über Datenskandale berichtet, die auf ein breites Echo gestoßen sind und somit auch die Politik auf den Plan gerufen haben. Diese muss gesetzliche Rahmenbedienungen schaffen, um einen Missbrauch von Daten verhindern.

Neben medial kommunizierten Nebenwirkungen birgt Big Data allerdings insbesondere für die Wirtschaft und auch für die öffentliche Verwaltung ein erhebliches Potenzial. In den vergangenen zwei Dekaden hat sich die Information bzw. das Wissen für viele westliche Unternehmen neben den Faktoren Boden, Kapital und Arbeit als ein wesentlicher Produktionsfaktor etabliert. In vielen Branchen hat sich dieser vierte Produktionsfaktor sogar als der wichtigste aller Faktoren erwiesen, da die gesamte Wertschöpfung auf diesem basiert und sich zum entscheidenden Differenzierungsmerkmal gegenüber dem Wettbewerb etabliert hat.

Das aus Big-Data-Analysen resultierende Potenzial für die Wirtschaft und die Aussicht auf die Erschließung neuer Märkte resultiert unmittelbar aus einer Forderung

seitens der Unternehmen nach neuen Technologien zur Datenverarbeitung. (BITKOM 2014, S. 13–20)

Innerhalb der Literatur wird indes davon ausgegangen, dass die unstrukturierten Daten, deren Auswertung bislang kaum möglich war, ein erhebliches finanzielles Potenzial in sich bergen, welches jedoch aufgrund der fehlenden Analysemöglichkeiten häufig nicht genutzt werden kann. Zur Verdeutlichung dieses Umstandes dient die Metapher, diese Daten als das neue Öl des digitalen Zeitalters zu bezeichnen. Wie Rohöl liefern die Daten in unverarbeiteter Form keinen Nutzen, sondern müssen zur Beantwortung neuer Fragestellung und Generierung von monetärem Nutzen zunächst mittels aufwendiger Analysen und Verfahren in eine strukturierte Form überführt werden. (Gesellschaft für Informatik e. V. 2014)

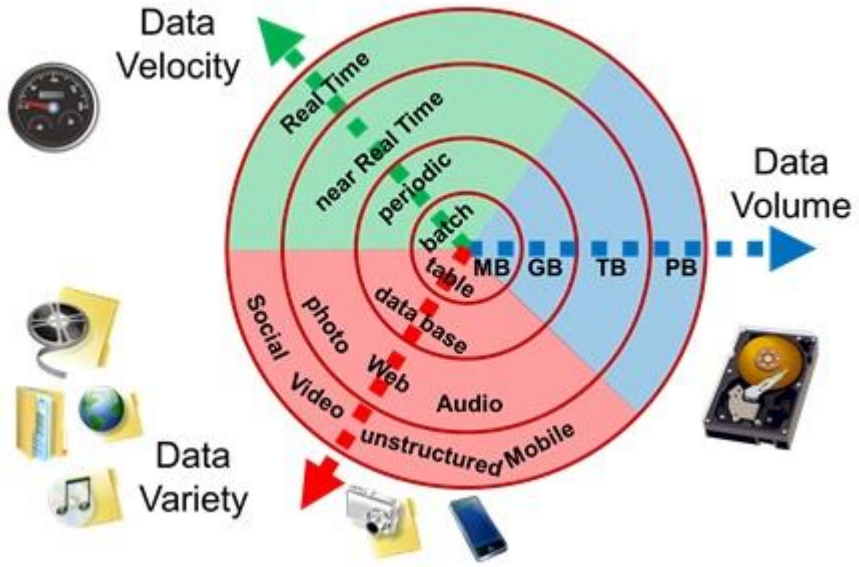

Abbildung 16: V-Modell für Big Data (Gesellschaft für Informatik e. V. 2014)

Die im obigen Absatz beschriebene Entwicklung lässt sich anhand einiger Zahlen verdeutlichen. Im Zeitraum der Jahre 2000 bis 2002 sind mehr Daten erstellt worden als in der gesamten Geschichte der Menschheit zuvor. In den darauffolgenden beiden Jahren von 2003 bis 2005 hat sich diese Datenmenge abermals vervierfacht. Bis 2012 hat sich das Volumen digitaler Daten auf 2,5 Zettabytes im Vergleich zu 2006 verzehnfacht. Unter der Annahme, dass sich diese Entwicklung in den kommenden Jahren so fortsetzt, ist davon auszugehen, dass die globalen Datenbestände schneller wachsen werden als die für die Analyse dieser zur Verfügung stehenden Ressourcen in der Datenverarbeitung. (Kisker 2014, S. 12)

3.2 Begriffsprojektion

Im vorherigen Kapitel sind die Herkunft und die verschiedenen Ausprägungen des Begriffs Big Data nochmals vorgestellt worden, ohne jedoch konkret zu werden, was

Big Data im konkreten Fall bedeutet. Dieses Vorgehen wurde bewusst gewählt, um eine möglichst allgemeine Auffassung des Begriffs zu vermitteln. Leser mit einer bereits konkreten Vorstellung von Big Data konnte diese so vielleicht bekräftigen oder mit wertvollen Erkenntnissen versorgen. Alle Leser, die unbedarft an das Thema herangegangen sind, sollten nun jedoch eine vage Vorstellung des Begriffs haben und vor allen Dingen nachvollziehen können, weshalb das Thema in den vergangenen Jahren so gehypt wurde.

Im nächsten Abschnitt soll das Thema bewusst weniger abstrakt angegangen werden. Stattdessen werden die vier Facetten (Datenvielfalt, Datenmenge, Geschwindigkeit, Analytics) des Begriffs anhand konkreter Beispiele betrachtet. Ziel ist es, nicht nur transparent zu machen, wo überall von Big Data gesprochen wird, sondern auch zu verstehen, weshalb so viele verschiedene Auffassungen des Begriffs existieren.

Gestartet werden soll mit dem wohl gewöhnungsbedürftigen V der drei Vs, nämlich der Variety bzw. der Datenvielfalt. Können sich die meisten Menschen, die in der Informationstechnologie tätig sind, noch recht gut vorstellen, was sich hinter Datenmenge und Geschwindigkeit verbirgt, so wird es vielen dieser Personen jedoch schwerfallen, sich ein konkretes Szenario zum Thema der Datenvielfalt vorzustellen. Obwohl jede dieser Person tagein, tagaus mit verschiedenen Datenformaten arbeitet oder diese konsumiert, wird den meisten dieser Person die Vorstellung, diese Daten analytisch zu verwenden und auszuwerten, dennoch schwerfallen. An dieser Stelle hinterlässt die jahrelange Arbeit mit relationalen Datenbanken und transaktionalen Systemen die wohl tiefsten Spuren. Das Dogma, dass Daten strukturiert vorzuliegen haben, um sie IT-gestützt verarbeiten zu können, ist nach wie vor vorherrschend und wird auch noch in vielen Bereichen für lange Zeit eine Daseinsberechtigung haben. Nichtsdestotrotz haben in den vergangenen zehn Jahren verschiedene Unternehmen und Institutionen erkannt, dass neben diesen strukturierten Daten auch die unstrukturierten Daten ein enormes Potenzial bergen.

Für ein besseres Verständnis, wie der Begriff Datenvielfalt im Big-Data-Kontext zu verstehen ist, soll das folgende Beispiel dienen.
Die Barmenia Lebensversicherung A.G. stand vor der Herausforderung, ihre alten host-basierten IT-Systeme durch moderne IT sowohl seitens der Hardware als auch der Software abzulösen. Im konkreten Fall handelte es sich um ein Host-System zur Verwaltung von Lebensversicherungsverträgen, die eine starke Heterogenität aufwiesen. Sowohl das Datenvolumen als auch die Geschwindigkeit, mit der die Daten verarbeitet werden müssen, waren in diesem Fall zu vernachlässigen, da die meisten der Verträge nicht mehr bearbeitet werden und im operativen Geschäft keine Bedeutung mehr spielen, sondern nur noch zur Bearbeitung von Sonderfällen vorgehalten werden. Zur Bewältigung dieser Herausforderung wurde Konsequent auf Big-Data-Methodik und -Technologien gesetzt, indem die Lebensversicherungsverträge – trotz

der Heterogenität von Lebensversicherungen, die sich über Jahrzehnte wandelten – vom Host auf die neue IT-Systeme transferiert wurden. Technisch wurde hierfür die dokumentenorientierte NoSQL-Datenbank MongoDB eingesetzt. Die Daten konnten unverändert in das neue System übernommen werden, ohne vorab sämtliche Datenbestände analysieren und anpassen zu müssen, um diese so in relationalen Datenbankmanagementsystemen ablegen zu können. Die neue Lösung erlaubt innerhalb des migrierten Datenbestandes weiterhin die Suche nach sämtlichen Attributen mit umfangreichen Anfrage-Operatoren. Zudem besteht die Möglichkeit, nachträglich notwendige Transformationen oder Analysen innerhalb der neuen Systeme und damit auch auf aktueller Infrastruktur umsetzen zu können. (BITKOM-Arbeitskreis Big Data 2015, S. 77)

Das obige Beispiel ist bewusst ausgewählt worden, um den Punkt Variety möglichst einfach zu erklären. Zwar wird durch den Einsatz von Big-Data-Technologien in diesem Fall kein zusätzlicher Mehrwert oder ein neues Wertschöpfungsfeld erschlossen, dennoch hat das gewählte Vorgehen auch einen unmittelbaren monetären Erfolg. Jeder IT-Verantwortliche, der in seiner Karriere in einem Migrationsprojekt ein Legacysystem ablösen musste, wird sicherlich berichten können, mit welchen Problemen und Aufwänden er konfrontiert wurde. Die Aussicht, bestehende Datenbestände *as-is*, also wie sie sind, ohne Anpassungen in ein neues System übernehmen zu können, kann somit nicht nur viel Arbeit, sondern auch bares Geld einsparen. (BITKOM-Arbeitskreis Big Data 2015, S. 77)

Das nächste Beispiel soll zur Veranschaulichung des Punktes Velocity, also der Geschwindigkeit, dienen. Dieser Aspekt ist vermutlich auf den ersten Blick der am einfachsten nachzuvollziehende Punkt bzw. der Punkt, unter dem sich die meisten Personen unmittelbar etwas vorstellen können. Ganz einfach formuliert ließe sich das Ganze wie folgt zusammenfassen. Etwas, das vorher lange gedauert hat, geht nun schnell bzw. schneller als zuvor. Da Zeit im kollektiven Bewusstsein ja bekanntlich Geld entspricht, liegt der hieraus resultierende Vorteil auf der Hand. Zur Vermittlung einer konkreten Vorstellung, welchen Mehrwert die Ersparnis von Zeit tatsächlich haben kann, soll das folgende Beispiel dienen. Hierbei handelt es sich um ein IT-Projekt aus dem Jahr 2013, in dem die SAP-In-Memory-Datenbank Hana zum Einsatz kommt.

Die Mercedes-Tochter AMG hat sich darauf spezialisiert, einzigartige und leistungsstarke Fahrzeuge zu entwickeln und zu produzieren. Ein großer Teil der Arbeit entfällt dabei insbesondere in der Neuentwicklung von Fahrzeugteilen auf das Testen und ist somit ein sehr ein aufwendiger und kostspieliger Teil der gesamten Fahrzeugentwicklung. Die Aussicht, Testergebnisse schneller verfügbar und analysierbar zu machen, resultiert daher nicht nur aus sinkenden Kosten, sondern auch daraus, ein Produkt

vor der Konkurrenz zur Marktreife zu bringen. Aus diesem Grund wurde die bestehende Datenbank im Motorenprüfstand gegen eine In-Memory-basierte Datenbank ausgetauscht, die in der Lage ist, Tausende Datensätze je Sekunde zu verarbeiten und zu visualisieren. Durch den Vergleich dieser Ergebnisse mit vorhandenen Aufzeichnungen können in nahezu Echtzeit kleine Abweichungen entdeckt und entsprechend auf diese reagiert werden. Auf diese Weise können fehlgeschlagene Testläufe bereits nach drei statt erst nach 50 Minuten abgebrochen werden, was sich schließlich auf einen zusätzlichen Testtag pro Woche summieren kann. (BITKOM-Arbeitskreis Big Data 2015, S. 40)

Das letzte Beispiel soll den Bereich betrachten, der von der reinen Analyse des Wortstamms Big (engl. groß) Data wohl am häufigsten mit diesem assoziiert wird. Gemeint ist der Punkt Volumen, also Datenmenge. Das zur Vorstellung dieses Punktes verwendete Beispiel wurde bzw. wird derzeit bei einem meiner eigenen Kunden umgesetzt. Zwar habe ich persönlich nicht direkt an dem Projekt mitarbeiten können, dennoch dient es hervorragend als Beispiel.

Im vorliegenden Fall handelt es sich um ein Unternehmen, dessen Hauptaufgabe der Transport von Erdgas innerhalb der BRD in einem über 12.000 km langen Fernleitungsnetz darstellt und somit unmittelbar für die nationale Infrastruktur relevant ist. Zum Vergleich: Die Strecke von 12.000 km entspricht etwa der Gesamtlänge des deutschen Autobahnnetzes. Im Rahmen der Digitalisierung in den letzten zwei Dekaden ist dieses Leitungsnetz flächendeckend mit digitalen Sensoren ausgestattet worden, die sekündlich Werte wie Druck, Fließgeschwindigkeit und Temperatur erfassen. Im Rahmen eins Predictive-Maintenance Projekts sollten diese Daten nun langfristig erfasst und mittels Data-Mining-Algorithmen ausgewertet werden. Ziel dieser Auswertung ist es, den Zustand von Anlagen zu erfassen und vorhersagen zu können, wann eine Wartung durchgeführt werden sollte. Dieser Ansatz verspricht Kosteneinsparungen gegenüber Routine oder zeitbasierten vorbeugenden Wartungen, da Aufgaben nur dann durchgeführt werden, wenn diese tatsächlich erforderlich sind.

Mit Initiierung des Projekts stellten sich somit zwei entscheidende Fragen. Wo bzw. wie sollen die anfallenden Daten abgelegt und auf welche Art anschließend verwaltet werden. Innerhalb der IT-Umgebung koexistierten zu Beginn des Projekts bereits verschiedene Plattformen zur Datenhaltung, mit denen bereits Erfahrungen gesammelt und etliche Projekte umgesetzt wurden, unter anderem ein klassisches Oracle-basiertes Data Warehouse sowie ein SAP Business Warehouse auf der SAP-Hana-Plattform. Dennoch wurde entschieden, das Projekt mittels des Apache Hadoop Software Frameworks umzusetzen. Hierfür spricht mehr als der zunächst anzunehmende Gedanke, dass eine sehr große Datenmenge zu verarbeiten und die Hadoop-Technologie hierfür hervorragend geeignet ist. Eine klassische relationale Datenbank wie z. B. eine Oracle 12c oder auch eine SAP HANA kann ebenfalls fast beliebig skaliert

werden, und partitionierte Tabellen und Indizes wären hier nur eine zu nennende, bereits lange verfügbare Technologie aus dem RDBMS-Umfeld, mit denen große Datenmengen effizient gespeichert werden können. Gleichwohl wurde die Entscheidung getroffen, bei der Speicherung auf das Apache Hadoop Framework zu setzen. Hierfür sprechen zum einen die gegenüber klassischen Datenbankmanagementsystemen geringeren Speicherkosten und zum anderen auch die Möglichkeit, bei der Struktur der eingehenden Daten flexibel sein zu können.

Der aber wohl wesentlichste Grund für den Einsatz des Apache Hadoop Frameworks waren die mit diesem Framework zur Verfügung stehenden integrierten Analysemöglichkeiten. An dieser Stelle kommt die vierte Facette des Begriffs Big Data zum Tragen. Gemeint ist der Punkt Analytics. Im vorliegenden Beispiel waren hier die beiden Data-Mining- und Machine-Learning-Apache-Projekte Mahout und Apache Spark MLib ausschlaggebend, bei denen es sich um die populärsten Data-Mining-Komponenten im Hadoop Ökosystem handelt.

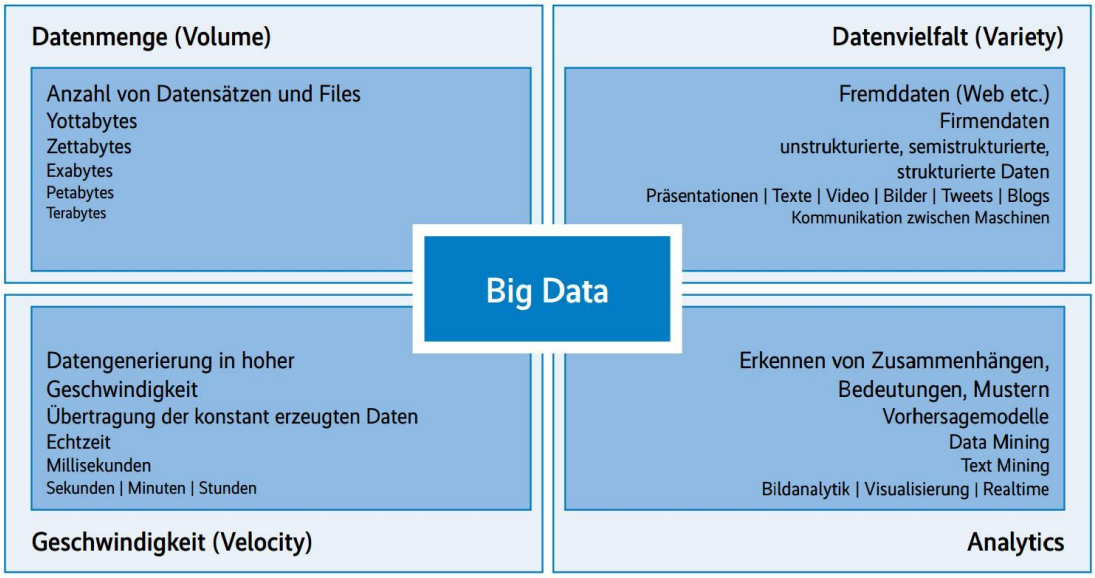

Abbildung 17: Heise Medien GmbH & Co. KG (Heise Medien GmbH & Co. KG 2015, S. 9)

Die obige Abbildung zeigt die vier Facetten des Begriffs Big Data, wie sie vom Autor des Artikels „Raffinierte Daten“ im iX-Developer-Sonderheft „Big Data“ verstanden wird. Die Grafik selbst liefert hinsichtlich der Separierung des Begriffs Big Data nicht mehr Erkenntnisse oder Inhalt als bereits die Abbildung des V-Modells im vorherigen Kapitel. Dennoch weist sie eine entscheidende Besonderheit auf, indem sie den Punkt Analytics gleichberechtigt mit den anderen drei Punkten Velocity, Volume und Variety darstellt. Dieser vierte Punkt wird an anderer Stelle auch als das vierte V-Wort – Value – des Begriffs Big Data verstanden, um den Wertschöpfungsaspekt der Daten zu betonen. (Heise Medien GmbH & Co. KG 2015, S. 8)

Bei der rückblickenden Betrachtung der Big-Data-Begriffsherkunft und der Einleitung dieser Arbeit kann der Punkt Analytics als der wesentlichste Punkt von Big-Data-Projekten betrachtet werden. An mehreren Stellen wurde auf das monetäre Potenzial von bislang ungenutzten Daten hingewiesen oder diese sogar als das Öl des 21. Jahrhunderts bezeichnet, jedoch ohne zu erklären, wie dieses nun tatsächlich vonstatten gehen soll. Erst die Nutzung und Auswertung der Daten ist es, was einen Mehrwert generieren kann, nicht jedoch das reine Vorhandensein. In diesem Zusammenhang gibt es das folgende, sehr passende Zitat aus einer Präsentation der SAP zum Thema SAP, Hadoop und Big Data.

Abbildung 18: Unterschied zwischen Datennutzung und Datenbesitz(SAP 2015, S. 6)

In den beiden vorherigen Kapiteln ist versucht worden, den Dunstschleier um den Begriff Big Data ein wenig zu lüften. Hierzu sind der Begriff und dessen Herkunft zunächst differenzierter betrachtet worden, um ihn anhand von konkreten Beispielen in die Praxis zu projizieren und ihn somit greifbarer werden zu lassen. Aus dieser Projektion sollen sich für den Leser die folgenden Erkenntnisse ergeben.

1. Große Datenmengen können ein Bestandteil von Big-Data-Projekten sein, müssen es aber nicht.
2. Ein Big-Data-Projekt kann alle vier Vs des Begriffs Big Data umfassen, aber letztendlich kann jede der vier Facetten für sich allein stehen.
3. Big Data kann nicht klar abgrenzt werden. Die Tatsache, dass jede Person unter dem Begriff zu Recht etwas anderes versteht und hiermit assoziiert, ist die logische Schlussfolgerung aus dieser Betrachtung.

Dennoch stellt sich die Frage, was Big Data denn nun für den Einzelnen ist. Diese Frage soll und kann an dieser Stelle nicht beantwortet werden. Gleichwohl kann hier

eine Hilfestellung für die Beantwortung gegeben werden. Im September 2015 hat die SAP ein neues Produkt veröffentlicht, welches den Namen *SAP HANA Vora* trägt. Auf der Webseite zu SAP Hana Vora wird auf Managementebene erklärt, welchen Mehrwert der Einsatz des Produkts haben kann. Hierfür werden wenig konkrete Aussagen getroffen, die jedoch für jeden Einzelnen eine unterschiedliche Interpretation ermöglicht. So heißt es:

„Kombinieren Sie Ihre Geschäftsdaten mit Daten aus externen Quellen, um eine schnellere und fundierte Entscheidungsfindung zu ermöglichen – auch unter sich ständig ändernden Umständen.“ (SAP HANA Vora | Hadoop | In-Memory-Abfrage)

In der Aussage werden zwei unterschiedliche Arten von Daten genannt – auf der einen Seite die Geschäftsdaten und auf der anderen Seite Daten aus externen Quellen. Es ist anzunehmen, dass es sich bei den Geschäftsdaten um Daten handelt, die unmittelbar im Zusammenhang mit dem Ziel einer Organisation stehen und vermutlich auch bereits systematisch verwendet und analysiert werden. Für einen Autovermieter könnten dies z. B. die einzelnen Vermietungen und der Fahrzeugbestand sein, eben jene Daten, auf denen unmittelbar die Wertschöpfung des Unternehmens geschieht. Würde der Chef dieser Autovermietung gefragt, ob die Daten über einzelne Vermietungen und über den Fahrzeugbestand als Big Data zu verstehen seien, würde dieser die Frage vermutlich verneinen und angeben, dass hierfür bereits umfangreiche Auswertungen bestünden. Würde dieselbe Frage jedoch zu den Telemetrie-Daten der Fahrzeuge gestellt, fiele die Antwort vermutlich völlig anders aus. Ebendiese Datenquellen, seien sie extern oder auch intern sind es, bei denen es sich im vorliegenden Beispiel um Big Data handelte. Die technischen Eigenschaften (Variety, Volume, Velocity) dieser Daten sind hierbei fast zu vernachlässigen. Wichtiger ist in dieser Betrachtung der Umstand, dass Daten erfasst und analysiert werden, die zunächst keine unmittelbare fachliche Verbindung zu den Geschäftsdaten des Unternehmens aufweisen.

Big Data kann also immer dann vorhanden sein, wenn Daten erfasst und analysiert werden, deren Zusammenhang zum eigentlichen Ziel einer Organisation nicht unmittelbar erkenntlich ist, die Erfassung und Analyse bei näherer Betrachtung jedoch in einen Mehrwert münden können. Zentrale Aufgabe für die Person eines Unternehmens, welche für die Datenerhebung und Auswertung verantwortlich ist, wird es zukünftig also sein, ebendiese Daten zu identifizieren und zu monetisieren.

4 Hadoop

4.1 Einführung

Im folgenden Abschnitt soll die Technologie rund das Apache Hadoop Framework vorgestellt werden. Hierzu wird zunächst kurz die ursprüngliche Idee hinter dem Apache Hadoop Framework erörtert, um anschließend einige der populärsten Softwareprojekte im Hadoop-Ökosystem vorzustellen. Die nachfolgenden Absätze entstammen zu Teilen einer bereits zur Vorbereitung dieser Abhandlung veröffentlichten Arbeit mit dem Namen „Prototypische Implementierung eines Apache-Hadoop-Clusters auf Basis von ARM Kleinstrechnern“. Die entsprechende Arbeit kann dem Anhang dieser Studie vollständig entnommen werden.

Hadoop entstand ursprünglich aus drei Technikentwürfen, sogenannten White Papers, die von Google zwischen 2003 und 2005 veröffentlicht wurden. Diese drei White Papers beschreiben das Google File System (FGS), MapReduce (MR) und Googles Big Table. Aus diesen Konzeptpapieren sind in der Folge die Technologien Hadoop Distributed File System (HDFS), MapReduce und Apache HBase hervorgegangen. Hierbei ist zu beachten, dass die von Google veröffentlichten Dokumente lediglich Konzepte und Architekturen, indes keine konkrete Implementierung enthielten. Doug Cutting, ein damaliger Softwareingenieur von Yahoo, nahm sich dieser Entwürfe an und implementierte in den Systemen von Yahoo die erste Version dessen, was heute als Apache Hadoop bekannt ist. Neben den ersten Implementierungen gehen auf ihn auch die Namen der Technologien zurück. (Martin Bayer 2013)

Abbildung 19: Ursprung des Apache Hadoop Frameworks

Die erste Implementierung von Hadoop wurde im Anschluss als ein Projekt in die Apache Software Foundation aufgenommen und von da von vielen großen internationalen Unternehmen weiterentwickelt. (Sameer Farooqui 2013)

Es stellt sich nun jedoch die Frage, was sich genau hinter der Technologie mit dem lustigen Namen und dem noch viel lustigeren Logo verbirgt und welche Ziele damit verfolgt werden.

Abbildung 20: Apache-Hadoop-Logo (The Apache Software Foundation 2015)

An erster Stelle ist es wichtig, herauszustellen, dass Hadoop nicht als eine einzelne, isolierte Technologie verstanden werden darf. Hadoop wird in der Regel immer als ein Open Source Software Framework beschrieben, das eine Vielzahl verschiedener Technologien umfasst. Häufig wird aus diesem Grund auch der Begriff Hadoop-Ökosystem angeführt, um die Komplexität und Größe der Thematik zu verdeutlichen. (Owen O'Malley 2013)

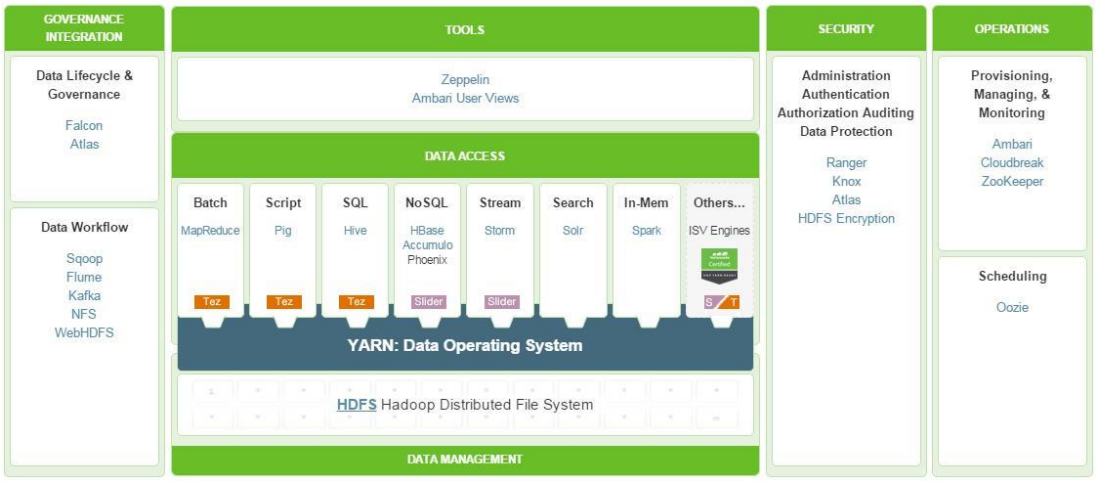

Abbildung 21: Hadoop-Ökosystem nach Hortonworks (Hortonworks Inc. 2014)

Die obige Abbildung zeigt eine beispielhafte Abbildung des Hadoop-Ökosystems, wie es vom Unternehmen Hortonworks Inc. verstanden wird. Hortonworks selbst ist dabei eine der erfolgreichsten und bekanntesten Hadoop-Distributionen am Markt, die vollständig auf dem originalen Apache-Softwareprojekt basiert. Laut Owen O‘Malley, Mit-

gründer von Hortonworks, dient Hadoop als Lösung, um große Datenmengen zu verarbeiten, die sich schnell ändern und durch Heterogenität gekennzeichnet sind. Um dies zu bewerkstelligen, ist Hadoop in der Lage, Standard-Rechnersysteme so zusammenzubringen und zu nutzen, dass riesige Mengen an Daten zuverlässig von ihnen verwaltet werden können. Aus diesen zusammengeschalteten Rechnersystemen ergibt sich ein Rechenverbund, der durch verteilte Rechenoperationen nicht nur in der Lage ist, diese großen Datenmengen zu speichern und zu verwalten, sondern auch aktiv zu verarbeiten und zu analysieren. Ebendiese Funktionen – das verteilte Speichern und Analysieren – sind die beiden zentralen Bestandteile einer jeden Hadoop-Installation. Technologisch spiegeln sich diese Fähigkeiten in den beiden zentralen Softwarekomponenten Hadoop Distributed File System (HDFS) und der Analyse- und Verarbeitungskomponente MapReduce wider. (Owen O'Malley 2013)

Da Hadoop sich historisch aus den beiden zentralen Komponenten, dem Hadoop Distributed Filesystem und der Analysekomponente MapReduce, zusammensetzt, soll deren Funktionsweise in den folgenden Abschnitten näher erläutert werden. Diese beiden Techniken dienen am besten dazu, das Grundprinzip hinter Hadoop zu verstehen. Es ist jedoch wichtig, zu bedenken, dass es nicht die beiden einzigen Komponenten sind. Das Hadoop-Ökosystem beherbergt etliche Softwareprojekte, die in unmittelbarer Konkurrenz zu diesen beiden Systemen stehen und bereits an deren Thron sägen. Insbesondere für das MapReduce Framework gibt es mittlerweile etliche Konkurrenzprodukte, wie etwa Apache Tez oder auch Apache Spark.

4.2 Architektur

Der grundlegende Gedanke hinter Hadoop ist nicht neu, da es sich im weiteren Sinne um eine klassische Clusterarchitektur handelt, wie sie in vielen Bereichen der IT eingesetzt wird. Wenn allerdings versucht würde, Hadoop in eine der drei Kategorien der Hochverfügbarkeits-, Load-Balancing- oder High Performance Computing Cluster einzuordnen, so wird dies nicht auf Anhieb gelingen.

Hadoop basiert auf einer klassischen Master-Slave-Architektur. Dies bedeutet, dass es innerhalb eines Hadoop-Clusters einen oder mehrere übergeordnete Instanzen gibt, denen die Verwaltung der restlichen untergeordneten Instanzen obliegt. Die Master-Slave-Architektur steht im Gegensatz zu einer klassischen Peer-to-Peer-Architektur, in der alle Knoten eines Clusters gleichberechtigt sind. Eine solche Peer-to-Peer-Architektur findet sich z. B. in Amazons Dynamo-Technologie. Im Falle von Hadoop handelt es sich bei der Masterinstanz um den sogenannten Namenode, dem wiederum Tausende Slave-Instanzen, die Datanodes, zugeordnet sein können. Der Namenode stellt dabei einen Dienst zur Verfügung, der als Jobtracker bezeichnet

wird, wohingegen von den Datanodes ein Tasktracker ausgeführt wird. (Sameer Farooqui 2013)

Abbildung 22: Vereinfachte schematische Darstellung einer Hadoop-Architektur (Sameer Farooqui 2013)

Wie zuvor in der Hadoop-Einführung angekündigt, setzt sich Hadoop historisch aus zwei zentralen Komponenten zusammen – zum einen das Hadoop Distributed Filesystem, das für die Speicherung der im Cluster abgelegten Daten zuständig ist, sowie MapReduce, das als Batchverarbeitungssystem die abgelegten Daten verarbeitet und analysiert. Es handelt sich hierbei um zwei eigenständige Komponenten, die in verschiedenen Diensten verwaltet werden. HDFS greift hierbei auf die Name- und Datanodekomponenten zurück, wohingegen MapReduce Job- und Tasktrackerdienste verwendet. (Sameer Farooqui 2013)

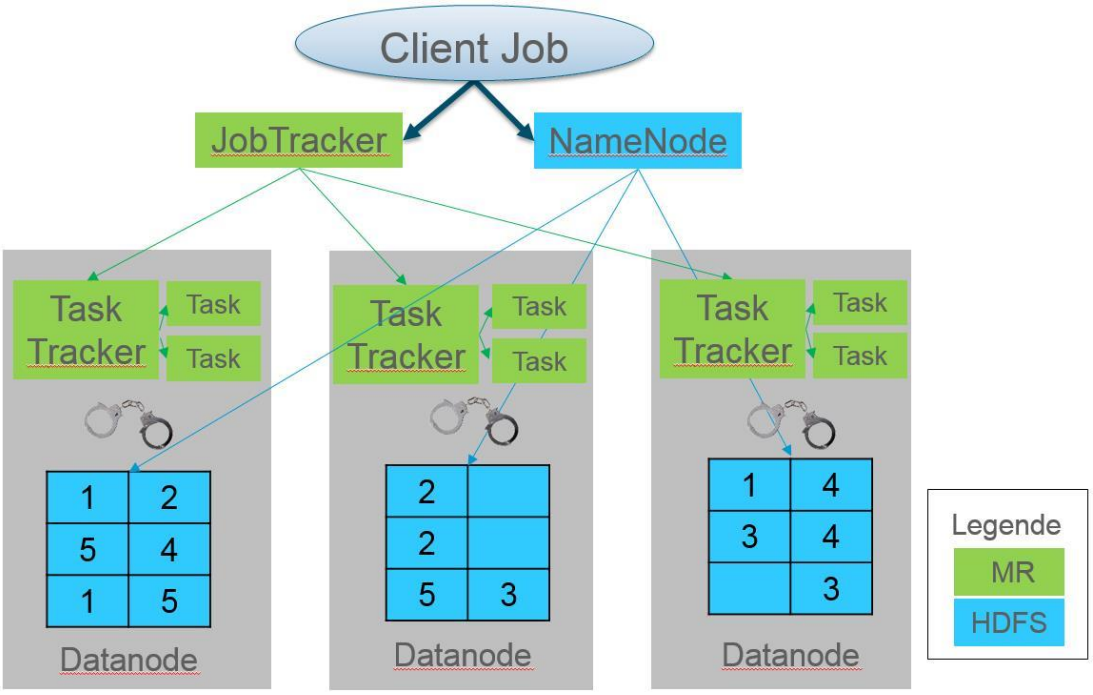

Abbildung 23: Funktionsteilung innerhalb eines Hadoop-Clusters

Bei der vorgestellten Architektur wird einem technisch versierten Leser schnell auffallen, dass es sich beim Namenode um einen Single Point of Failure (SPOF) handelt. Ein solcher Single Point of Failure ist als eine Komponente in einem IT-System zu verstehen, dessen Ausfall den Ausfall des gesamten Systems nach sich zieht. Bei weiterer intensiver Beschäftigung mit dem Thema Hadoop wird aus diesem Grund eine weitere Komponente innerhalb des Hadoop Frameworks auffallen, die den Namen Secondary Namenode, also zweiter Hauptknoten, trägt. Nun könnte angenommen werden, dass dieser Secondary Namenode ein Backup für den Namenode darstellt und im Falle eines Ausfalls dessen Aufgabe übernimmt. Dies ist so nicht korrekt. Der Secondary Namenode stellt kein Backup für den Namenode zur Verfügung, sondern lagert Aufgaben von diesem aus, die dieser selbst nicht ausführen kann. Hierunter gehört z. B. die dauerhafte physische Persistierung sämtlicher Metadaten. Auf die Funktion des Secondary Namenode im Detail soll an dieser Stelle jedoch eingegangen werden. Zum einen würde dies den Prozess und damit die Verständlichkeit nur unnötig verkomplizieren und zum anderen die Funktion des Secondary Namenode für das Grundverständnis irrelevant werden lassen. Der einzige Grund, weshalb der Secondary Namenode überhaupt erwähnt wurde, besteht darin, dem Trugschluss vorzubeugen, dass es sich bei diesem um ein Backup für den Namenode handele.

4.3 HDFS

HDFS steht für Hadoop Distributed Filesystem, was auf Deutsch so viel wie Hadoop-verteiltes Dateisystem bedeutet. Wie der Name nahelegt, handelt es sich dabei um das logische Dateisystem eines Hadoop-Clusters. Wichtig ist, dass es nicht mit klassischen physischen Dateisystemen wie ext3, ext4 oder XFS verwechselt wird. HDFS ist ein virtuelles Dateisystem, das über ein physisches Dateisystem gelegt wird, sich dabei jedoch auf viele verteilte Knoten erstreckt. Diese Aufgabe wird dabei in einem Cluster von den zuvor erwähnten Datanodes übernommen, die von einem übergeordneten Namenode verwaltet werden. (Sameer Farooqui 2013)

Besonderheit des HDFS ist jedoch, dass eine zu speichernde Datei nicht in Gänze auf einem einzelnen Datanode abgelegt wird, sondern diese ab einer gewissen Größe aufgeteilt und über mehrere Datanodes verteilt wird. In Abhängigkeit eines frei definieren Replikationsfaktors werden diese Dateifragmente jedoch nicht nur auf einem Knoten, sondern auf *n* verschiedenen Knoten abgelegt, wobei *n* der Höhe des Replikationsfaktors entspricht. Das heißt, dass bei einem Replikationsfaktor von drei eine Datei exakt dreimal im gesamten Cluster verfügbar ist. Auf diese Weise kann der Ausfall einzelner Datanodes kompensiert werden. Das folgende Beispiel soll dieses Verfahren verdeutlichen. (Sameer Farooqui 2013)

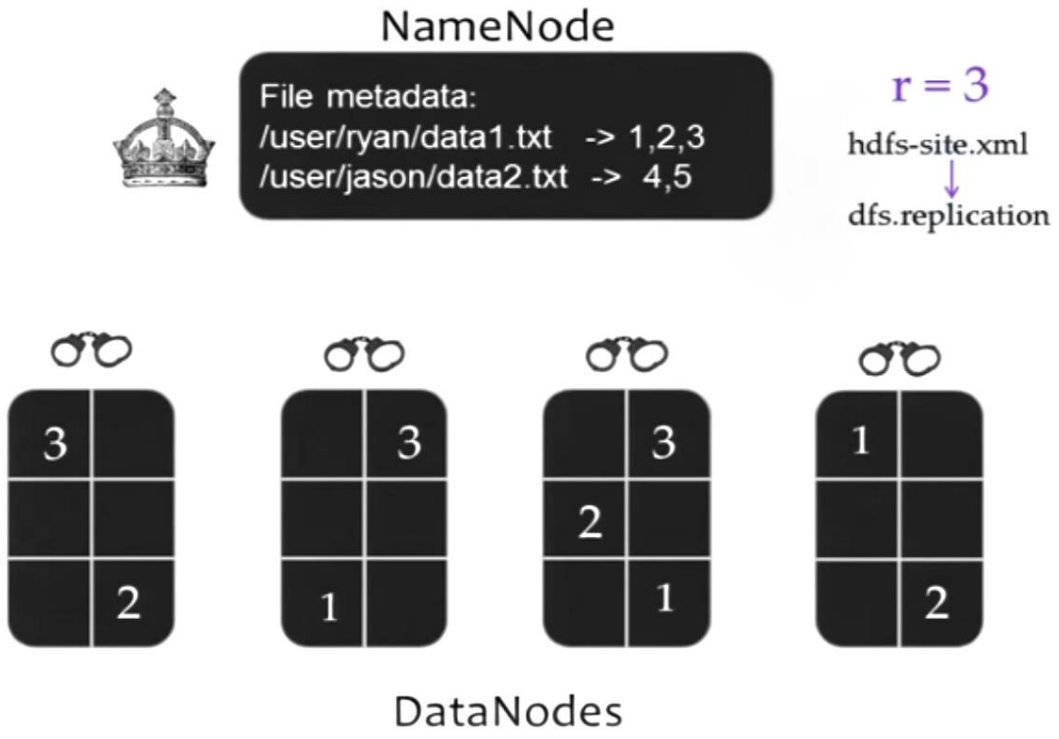

Abbildung 24: HDFS-Verteilung und Replikation der Datenblöcke (Sameer Farooqui 2013)

Im dargestellten Beispiel sollen die Dateien data1.txt und data2.txt im HDFS abgelegt werden. Zum aktuellen Stand ist bislang jedoch nur die Datei data1.txt abgelegt worden. Die Datei data1.txt wurde dabei in drei Dateiblöcke (1, 2, 3) aufgeteilt. Der Replikationsfaktor, also der Wert, der angibt, wie oft eine Datei im Clustervorhanden sein soll, liegt bei drei (r = 3). Im Beispiel gibt es vier Datanodes. Der Namenode hat zuvor entschieden, dass die Datei aufgrund ihrer Größe in drei Blöcke aufgeteilt wird, und hat anschließend angewiesen, dass diese drei Blöcke gleichmäßig auf die vier Datanodes verteilt werden. Gleichmäßig bedeutet in diesem Sinne, dass die einzelnen Dateiblöcke so auf die Datanodes verteilt werden, dass möglichst viele von diesen ausfallen können, ohne dass die Datei in Gänze nicht wiederhergestellt werden kann. Im vorliegenden Beispiel können bis zu zwei beliebige Nodes entfernt werden, die Datei kann indes immer noch aus den beiden verbleibenden Nodes vollständig zusammengesetzt werden. Das Verfahren, über das die einzelnen Dateiblöcke im HDFS verteilt werden, ist um ein Vielfaches aufwendiger als hier vereinfacht beschrieben. Unter anderem kann die physische Verteilung der Server im Netzwerk und in einem Rechenzentrum mit in das Verfahren einfließen, was allerdings an dieser Stelle nicht weiter behandelt werden soll. (Sameer Farooqui 2013)

Die Information, wo welches Dateifragment liegt, wird an zentraler Stelle im Metadatenverzeichnis des Namenode verwaltet. Fällt eine Slave-Instanz respektive Datanode aus, so ist das Hadoop-Cluster in der Lage, sich selbst zu heilen. Der Namenode registriert einen solchen Ausfall zeitnah und handelt entsprechend, indem er die Dateien, die auf dem ausgefallenen Knoten lagen, von anderen Knoten so oft auf andere Knoten repliziert, bis das vorgeschriebene Replikationslevel wieder erreicht ist. Dies ist möglich, da bei einem Ausfall eines Knotens und einem Replikationsfaktor von drei noch mindestens zwei Kopien dieser Dateiblöcke auf anderen Knoten liegen müssen. Selbiges gilt, wenn ein neuer Knoten in ein HDFS Cluster aufgenommen wird. Der Namenode erkennt diesen und repliziert von anderen Knoten Dateiblöcke auf diesen, die dann wiederum von anderen Knoten entfernt werden können. (Sameer Farooqui 2013)

Durch die Replikation der Dateien innerhalb eines Clusters verringert sich der verfügbare Datenspeicher immer um den jeweiligen Replikationsfaktor. Im Beispiel bedeutet dies, dass ein Cluster, das aus vier Knoten zu je 60 GB Dateispeicher besteht, eine Bruttokapazität von 240 GB aufweisen würde. Faktisch wird jedoch jede Datei dreimal abgelegt, wodurch auch der verfügbare Speicher durch drei dividiert werden muss, woraus eine Nettokapazität von 80 GB resultierte.

Obwohl das HDFS nicht mit physischen Dateisystemen zu verwechseln ist, so unterstützt es gleichwohl eine Vielzahl der typischen Operationen, die auch von klassischen Dateisystemen unterstützt werden. Als Beispiel wären hier die folgenden zu nennen:

- *bin/hadoop fs –ls* --> Ausgeben der Dateien und Verzeichnisse
- *bin/hadoop fs –mkdir* --> Ordner im HDFS erstellen
- *bin/hadoop fs -rm* --> Datei löschen
- *bin/hadoop fs –chmod* --> Zugriffsrechte für eine Datei anpassen

(Sameer Farooqui 2013)

4.4 MapReduce

Neben HDFS bildet MapReduce die zweite entscheidende Komponente von Hadoop. MapReduce stellt die Verarbeitungslogik und gleichzeitig auch das Programmierparadigma hinter Hadoop dar.

Wenn von MapReduce gesprochen wird, ist in der Regel immer von sogenannten MapReduce Jobs die Rede. Diese Jobs werden von der zentralen Job-Tracker-Komponente eines Clusters verwaltet, der deren Ausführung an die einzelnen Task Tracker eines Clusters delegiert. Wie HDFS nutzt auch MapReduce den Vorteil einer Cluster-Architektur. Hierdurch wird bei Hadoop also nicht nur die Speicherkapazität der einzelnen Slave-Instanzen zur Ablage großer Datenmengen verwendet, sondern auch deren Rechenkapazitäten. Neben dem akkumulierten Speicherplatz, den die Slaves bereitstellen, kann somit auch von ihrer gemeinsamen Rechenleistung profitiert werden. Damit ist Hadoop auch in diesem Punkt genauso skalierbar wie beim Speicherplatz. (Sameer Farooqui 2013)

MapReduce Jobs können auf unterschiedliche Weise erstellt werden. Für die Erstellung eines solchen Jobs wird typischerweise eine Kombination aus einer Anwendung (Software Bibliothek) und einer dazu kompatiblen Programmiersprache benötigt. Anwendung und Sprache können zum einen unmittelbar auf dem MapReduce-Ansatz basieren oder aber auch durch dazwischenliegende Schichten davon abstrahiert werden. Bei einer direkten Implementierung von MapReduce Jobs durch den Anwender verwendet dieser in der Regel die Programmiersprache Java – es können jedoch auch Sprachen wie JavaScript, Python, C#. C++ oder R eingesetzt werden. (Lynn Langit 2014)

Bei einer unmittelbaren Umsetzung von MapReduce Jobs steht dem Nutzer bzw. dem Programmierer die größtmögliche Flexibilität und damit das höchste Potenzial zur Verfügung. Je näher ein Programmierer sich allerdings unmittelbar am Framework bewegt, desto höher werden die Komplexität und der damit für ihn verbundene Aufwand. Aus diesem Grund wurden in der Vergangenheit etliche Zwischenschichten entwickelt, die in der Lage sind, höhere, weniger komplexe Sprachen in MapReduce Jobs zu übersetzten. Eine der bekanntesten dieser Zwischenschichten ist Hive, die mit HiveQL eine an SQL angelehnte Syntax anbietet, die wiederum in MapReduce Jobs übersetzt wird. Ein Problem dieser generisch übersetzten MapReduce Jobs ist

jedoch, dass diese häufig weniger performant sind als ihre nativ entwickelten Äquivalente. Stark abstrahiert kann dieses Vorgehen mit der klassischen Programmierung verglichen werden. Auf der einen Seite steht die hardwarenahe, sehr schnelle, indes aufwendige Assemblerprogrammierung. Dem gegenüber befinden sich moderne, gemanagte Programmiersprachen wie Java oder C#.

Der Ablauf eines MapReduce Jobs erfolgt in drei Schritten, wobei MapReduce sich den Umstand zunutze macht, dass die Datenblöcke einer Datei auf vielen Slave-Instanzen des Clusters verteilt liegen. Der erste Schritt wird also nur von Instanzen ausgeführt, welche die hierfür relevanten Datenblöcke enthalten. Die Zuweisung, welcher Knoten eines Clusters dabei welchen Teil übernimmt, wird vom zentralen Job Tracker übernommen, der den untergeordneten Task Trackern entsprechende Aufgaben zuteilt. Die Information, welcher Datenblock auf welchem Knoten vorhanden ist, wird dabei vom Namenode zur Verfügung gestellt. Durch den Umstand, dass die einzelnen Datenblöcke auf mehreren Knoten liegen, kann die Last so gleichmäßig im Cluster verteilt werden. (Hortonworks 2013)

Die einzelnen Schritte setzen sich dabei wie folgt zusammen:

Map: Im Map-Schritt werden aus den Eingabewerten Schlüssel/Wert Paare gebildet. Die Umsetzung dieses Teils muss durch den Programmierer erfolgen.

Shuffle: Die erzeugten Schlüssel-/Wert-Paare dienen dem Shuffle-Teil als Eingangsparameter. Hier werden die Daten entsprechend ihren Schlüsseln sortiert und zum Reduce-Schritt weitergeleitet.

Reduce: Der Reduce-Teil muss wieder durch den Programmierer erstellt werden. Hier findet eine Aggregation der zuvor gemappten, sortierten Daten statt.

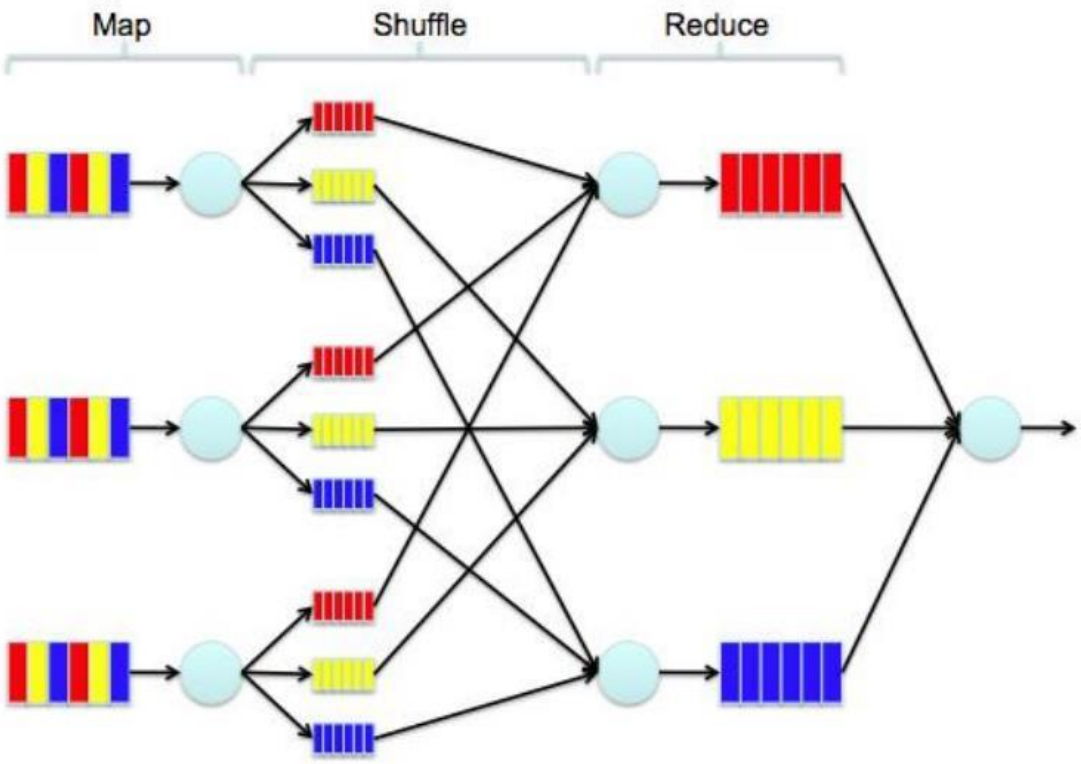

Abbildung 25: MapReduce: Abstrahiertes Funktionsprinzip (Lynn Langit 2014)

Der genaue Ablauf eines MapReduce Jobs soll hierzu an einem kurzen Beispiel erläutert werden. Ziel des MapReduce Jobs ist es, ein gewöhnliches Skat-Kartenspiel zu sortieren, wobei die Bilder (Bube, Dame, König, Ass) aussortiert werden sollen.

- Das Hadoop-Dateisystem HDFS hat das Kartenspiel auf mehrere Blöcke aufgeteilt. In jedem Block können sich Karten aller vier Farben (Caro, Herz, Pik, Kreuz) befinden.
- Im Map Teil des Jobs werden nun Schlüssel-/Wert-Paare gebildet. Die Farbe der Karte stellt dabei den Schlüssel und die Zahl den Wert dar. Die Bilder werden im Map-Teil aussortiert. Die einzelnen Blöcke werden bereits im Map-Vorgang vorsortiert. Ergebnis sind nun bis zu vier Blöcke je Knoten.
- Im Shuffle-Teil werden die nun vier gebildeten Blöcke in Abhängigkeiten ihrer Schlüssel zu einem bestimmten Reducer geschickt, der sich im Regelfall auf einer anderen Instanz befindet. Im Beispiel heißt dies, dass vor dem Shuffle-Vorgang jede Instanz bis zu maximal vier unterschiedliche Blöcke erstellt hat. Nach dem Shufflevorgang werden die einzelnen Blöcke anhand ihrer Farbe zusammengelegt. Instanz eins hat alle Herz-Karten, Instanz zwei alle Karo-Karten usw.
- Ein Reducer aggregiert nun die vier Zwischenergebnisse zusammen, wobei er aufgrund der Schlüsselwerte diese in der Reihenfolge Karo, Herz, Pik und Kreuz zusammenlegt, um abschließend ein vollständiges sowie sortiertes Kartendeck zu erhalten. (Jesse Anderson 2013)

Das Konstrukt von MapReduce birgt in seinem zentralen Vorteil auch den entschiedensten Nachteil. Durch die strikte Ausrichtung auf Batch-Verarbeitung ist dieses wenig flexibel und interaktiv und somit in vielen Fällen auch vergleichsweise langsam. Aus diesem Grund wurden neben MapReduce in der Vergangenheit weitere Alternativen ins Leben gerufen, welche die Datenverarbeitung weg von der Batch-Verarbeitung in den interaktiven Bereich beschleunigen. Eine dieser Alternativen ist die Execution Engine *Tez (Hindi für Geschwindigkeit)*, die einen gerichteten azyklischen Graph (DAG) zur Beschreibung eines Datenflusses und zur Verarbeitung ebendieser aufbaut. Die Architektur von Tez unterscheidet sich in vielen Punkten stark von MapReduce, wodurch Tez insgesamt in der Lage ist, intelligentere Ausführungsgraphen aufzubauen und die Verarbeitung großer Datenmengen im Vergleich zu MapReduce deutlich zu beschleunigen. Neben Tez avanciert derzeit eine weitere Execution Engine namens *Spark* zum Quasi-de-facto-Standard als Hadoop-Ausführungsengine in Konkurrenz zu Tez und MapReduce. Spark ist eine parallelisierte Engine, die auf eine In-Memory-Verarbeitung der Daten setzt. Da Spark derzeit eine besondere Rolle im Hadoop-Ökosystem zukommt, wird es an anderer Stelle noch detaillierter betrachtet. (Heise Medien GmbH & Co. KG 2015, S. 37)

4.5 Ökosystem

4.5.1 Einleitung

Nachdem nun die grundlegende Architektur hinter Hadoop erläutert wurde, werden im nächsten Abschnitt einige weitere wichtige Softwareprojekte rund um das Apache-Hadoop-Projekt vorgestellt. Die nachfolgend präsentierten Projekte wurden hinsichtlich der Realisierung eines Data Warehouse auf Basis von Apache Hadoop ausgewählt. Neben den nun folgenden Projekten existiert im Hadoop-Ökosystem jedoch noch eine Vielzahl weiterer Projekte.

Die nachfolgende Abbildung zeigt einen Ausschnitt, auf dem die meisten der aktuellen bzw. größten Softwareprojekte im Hadoop-Umfeld zu sehen sind.

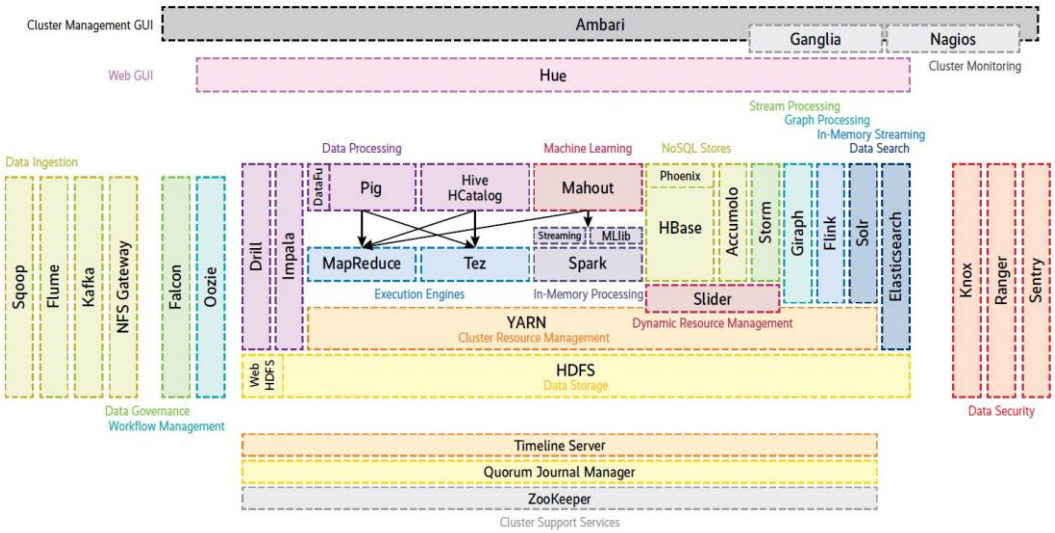

Abbildung 26: Softwareprojekte rund um das Apache Hadoop Framework (Heise Medien GmbH & Co. KG 2015, S. 37)

Bevor mit der näheren Betrachtung der einzelnen ausgewählten Projekte begonnen werden kann, muss noch ein kleiner Nachtrag zur Architektur moderner Apache-Hadoop-Lösungen eingeschoben werden. Die im vorherigen Kapitel beschriebene Architektur, die aus den beiden Komponenten HDFS und MapReduce besteht, entstammt der ersten Version von Apache Hadoop. Die Gültigkeit dieser Architektur ist nach wie vor unverändert und findet sich auch in aktuellen Hadoop-Versionen wieder, jedoch liegt Apache Hadoop seit Oktober 2013 stabil in der Version 2 (15. Oktober 2013, Version 2.2.0) vor. Zentrale Erweiterung dieser Version war die Integration von YARN – einer Zwischenschicht zwischen HDFS und MapReduce. YARN steht für *Yet Another Ressource Navigator,* der dafür zuständig ist, die in einem Cluster verfügbaren Ressourcen gleichmäßig auf alle Komponenten zu verteilen. Hierzu werden die Ressourcen in Container mit unterschiedlich zugewiesenem RAM sowie (virtuellen) CPUs unterteilt, in deren Grenze dann die jeweiligen Applikationen ihre Arbeit verrichten können. Diese Entwicklung war notwendig, da neben dem ursprünglich alleine vorherrschenden MapReduce nach und nach weitere Komponenten von verschiedener Seite in Hadoop integriert wurden. Die nachfolgende Abbildung zeigt die Entwicklung von Hadoop von Version 1 auf Version 2. (Heise Medien GmbH & Co. KG 2015, S. 36)

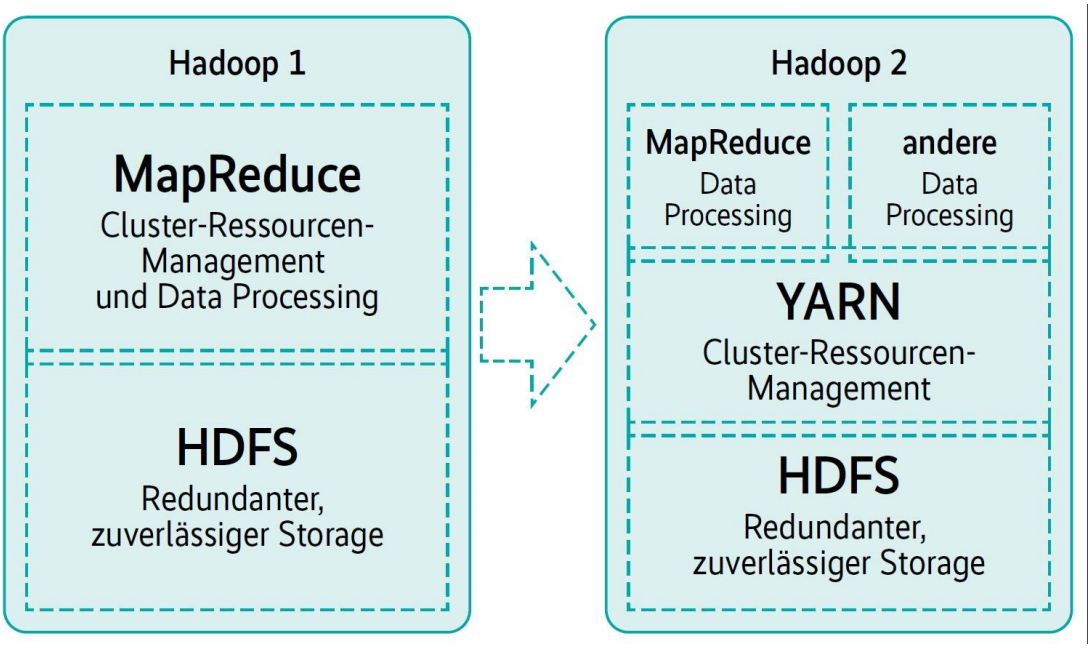

Abbildung 27: Der architektonische Wandel zu Hadoop 2 (Heise Medien GmbH & Co. KG 2015, S. 37)

4.5.2 Hive

In Anbetracht des Titels dieser Arbeit ist die Hive die erste Komponente, die dediziert betrachtet wird. Hive wird in vielen Fällen als Data-Warehouse-Lösung für Hadoop bezeichnet. Hive ermöglicht die strukturierte Ablage von Massendaten, wie sie in einem typischen Hadoop-Szenario bestehen. Hive bringt zu diesem Zweck eine eigene höhere Abfragesprache namens *veQL,* bei der es sich um eine ANSI-SQL-92-kompatible Syntax handelt. Hierdurch fühlt sich Hive in der Verwendung wie eine SQL-Datenbank an, allerdings mit dem Unterschied, dass deren Tabellen auf HDFS-Dateien basieren. Hive verwendet für die Ausführung seiner Operationen MapReduce, Tez oder auch zukünftig Spark. Hierdurch steht auf der einen Seite eine SQL-Datenbank-ähnliche Applikation zur Verfügung, die auf der anderen Seite jedoch alle Vorteile eines Hadoop-Clusters nutzen kann. (Rajesh Kartha 2012)

HiveQL ermöglicht die Formulierung von Anweisungen und Datenabfragen, ohne hierbei direkt MapReduce-Programme unter Verwendung einer Programmiersprache erstellen zu müssen. Hive wandelt hierzu HiveQL unmittelbar in einen MapReduce(oder Tez)-konformen Code um und führt diesen anschließend aus. Hierdurch ermöglicht Hive die Erstellung vieler verschiedener interaktiver Abfragen, ohne langwierige Anpassungen an den einzelnen MapReduce Jobs vornehmen zu müssen. Der HiveQL-Befehlssatz weist eine große Parallelität zu modernen SQL-Dialekten auf, wenn diese auch noch nicht so mächtig ist, wie man es z. B. von einer Oracle-Datenbank erwarten würde. Dennoch werden mit jedem Hive Release weitere Funktionen integriert, sodass die Lücke immer kleiner wird. Beispielsweise waren zur Version

0.11 noch keine Union- oder Intersect-Abfragen möglich, die in der aktuellen Version 0.14 vollständig zur Verfügung stehen. (Rajesh Kartha 2012)

Current SQL Compatibility

Hive SQL Datatypes	Hive SQL Semantics
INT	SELECT, LOAD INSERT from query
TINYINT/SMALLINT/BIGINT	Expressions in WHERE and HAVING
BOOLEAN	GROUP BY, ORDER BY, SORT BY
FLOAT	Sub-queries in FROM clause
DOUBLE	GROUP BY, ORDER BY
STRING	CLUSTER BY, DISTRIBUTE BY
TIMESTAMP	ROLLUP and CUBE
BINARY	UNION
ARRAY, MAP, STRUCT, UNION	LEFT, RIGHT and FULL INNER/OUTER JOIN
DECIMAL	CROSS JOIN, LEFT SEMI JOIN
CHAR	Windowing functions (OVER, RANK, etc)
CARCHAR	INTERSECT, EXCEPT, UNION, DISTINCT
DATE	Sub-queries in WHERE (IN, NOT IN, EXISTS/ NOT EXISTS)
	Sub-queries in HAVING

Color Key
Hive 0.10
Hive 0.11
FUTURE

Abbildung 28: Kompatibilität von HiveQL in der Version 0.11 (Hortonworks, S. 3)

Hive kann über verschiedene Wege verwendet bzw. in bereits vorhandene Systemumgebungen eingebunden werden. So stehen von einem Command-Line-Interface bis hin zu einem vollständigen Webinterface verschiedene Möglichkeiten zur Erstellung von Abfragen zur Verfügung. Die jedoch in der Praxis wohl am häufigsten verwendeten Schnittstellen stellen die ODBC- und JDBC-Treiber dar. Hierbei handelt es sich um standardisierte Datenbankschnittstellen, die es anderen Anwendungen ermöglicht, mit Hive zu kommunizieren und Daten zu lesen und oder zu schreiben, ohne die zugrunde liegende Architektur kennen zu müssen. (Rajesh Kartha 2012)

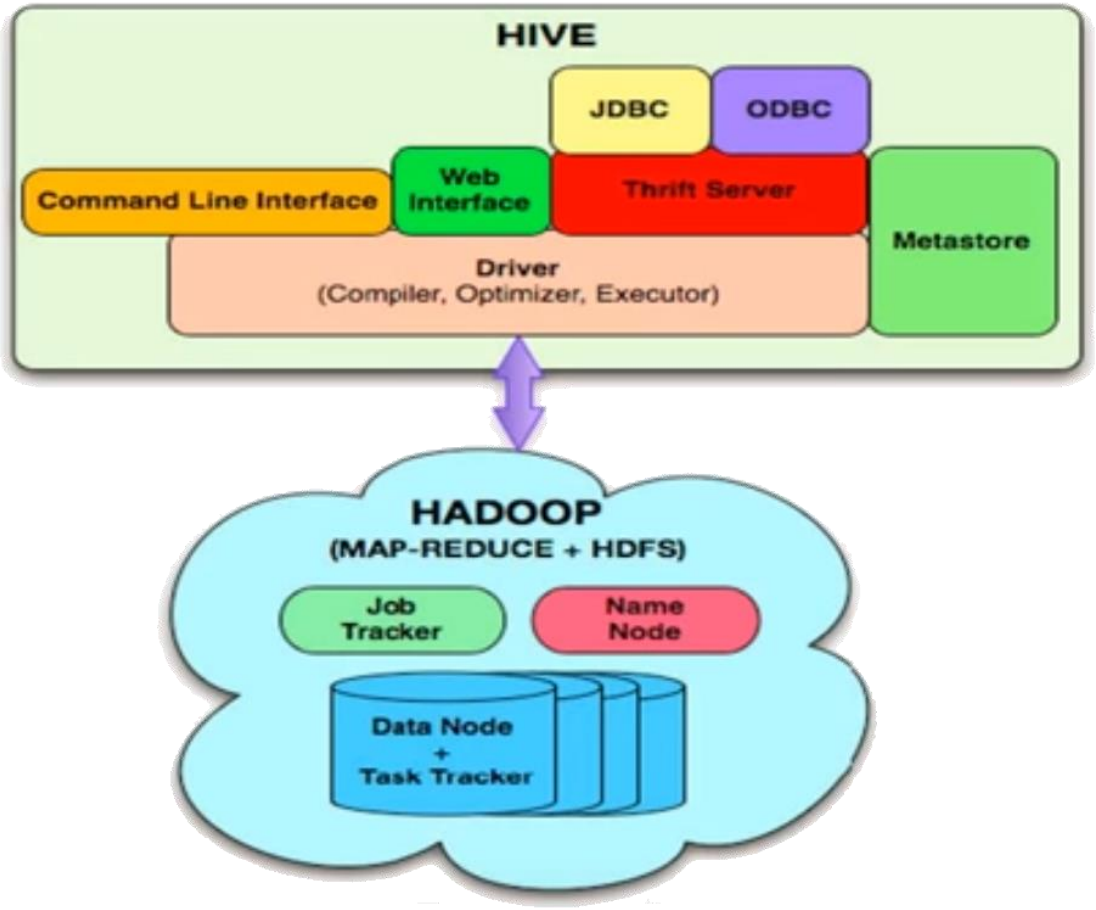

Abbildung 29: Hive Datastore inkl. Schnittstellen (Rajesh Kartha 2012)

An dieser Stelle wird die Ähnlichkeit von Hive zu klassischen RDBMS deutlich. S halten ebenfalls strukturierte Daten und bieten über die gleichen Schnittstellen einen Zugriff darauf an. Hive unterliegt indes auch hier einigen Einschränkungen. Die für transaktionale Systeme so wichtigen ACID-Transaktionen werden in Hive nicht unterstützt. Demzufolge kann keine Transaktionssicherheit gewährleistet werden. Zudem benötigen insbesondere kleine Transaktionen auf Hive im Vergleich zu klassischen RDBMS viel Zeit, was meist mit den Anforderungen transaktionaler Systeme nicht vereinbar ist. Das langsame Antwortverhalten von Hive erklärt sich mit der zugrunde liegenden MapReduce-Ausführungsengine, die für Batch-Jobs, jedoch nicht für schnelle Reaktionszeiten ausgelegt ist. Hive wurde dahin gehend optimiert, ad-Hoc Query, Aggregationen und Analysen von großen Datenmengen durchzuführen. Der Fokus von Hive liegt folglich auf einem lesenden Zugriff der Daten. Echtzeitabfragen, bei denen die Ergebnisse innerhalb weniger Sekunden zurückgeliefert sein müssen, oder Updates auf Zeilenbasis gehören somit nicht zu den Stärken von Hive. (Rajesh Kartha 2012)

Hive stellt allerdings mit *Schema-On-Read* eine Funktion dar, die klassische relationale Datenbanken nicht beherrschen. Schema-On-Read spiegelt sehr deutlich den Punkt Variety (Datenvielfalt) im Hadoop-Ansatz wider. Auf der einen Seite ist es nicht nur möglich, beliebige Dateitypen und Strukturen in das Hadoop-Dateisystem HDFS

zu schreiben, sondern diese auch flexibel betrachten zu können. Schema-On-Read ist als Gegenteil zu Schema-On-Write zu betrachten. Die meisten Werkzeuge sowie Anwender dieser Systeme erwarten, dass Daten bei Ablage in einer fest definierten Struktur abgelegt werden. Dieser Ansatz ist äußerst nützlich, um zum einen Verbindungen zwischen Daten zu erstellen, und bietet zum anderen auch einen Weg, Daten möglichst effizient abzulegen. Der Gedanke hinter Schema-On-Read ist logisch leicht nachzuvollziehen, da es schwer vorstellbar ist, dass bei Ausführung einer Transaktion Daten in unterschiedlichen Strukturen anfallen können. Transaktionale Geschäftsprozesse auf einem solchen Ansatz erfolgreich abzuwickeln, erscheint als völlig utopisch.

Schema-On-Read betrachtet die zu analysierenden Daten indessen nur aus einer lesenden Perspektive und entscheidend erst zum Zeitpunkt des Datenzugriffs, in welcher Struktur die Daten ausgegeben werden sollen. (Tom Deutsch 2013)

Das Konstrukt des Schema-On-Read ist für viele Personen, die aus einer klassischen transaktional, relational geprägten Welt stammen, zu Beginn nur schwer verständlich bzw. vorstellbar. Aus diesem Grund soll das folgende Beispiel das Konstrukt kurz erklären.

Die folgenden Befehle laden Goethes Faust aus dem Gutenberg-Projekt herunter und kopieren es anschließend auf ein Hadoop-Cluster, sodass es anschließend im HDFS liegt.

```
wget http://www.gutenberg.org/files/21000/21000-0.txt
mv 21000-0.txt faust.txt
hadoop dfs -mkdir /user/hue/books
hadoop dfs -put faust.txt /user/hue/books
```

Das folgend SQL-Statement legt nun eine Tabelle auf ebendieser Textdatei an, ohne hierbei jedoch die Daten direkt in diese Struktur zu laden. Die Tabelle ist somit vergleichbar mit einer View auf eine Texttabelle, die den gesamten Text in einzelne Worte zerlegt und je Wort eine Zeile darstellt.

```
CREATE EXTERNAL TABLE FaustWordCount
(word STRING, count INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE LOCATION '/user/hue/books;
```

Auf Basis dieser virtuellen bzw. externen Tabelle könnte nun die Anzahl der Wörter gezählt werden. Das folgende Statement zählt beispielsweise die zehn häufigsten Worte in Goethes Faust.

```
SELECT word, count
FROM FaustWordCount
ORDER BY count DESC LIMIT 10
```

Die Textdatei selbst liegt allerdings unverändert im HDFS vor, und mittels einer anderen externen Tabelle könnte die Textdatei in einer anderen Struktur betrachtet werden, beispielsweise zum Zählen von Sätzen oder einzelnen Buchstaben.

Das vorliegende Beispiel hilft daher, zwei Sachverhalte zu verstehen. Zum einen kann nachvollzogen werden, was Schema-On-Read bedeutet, zum anderen jedoch auch, was der Begriff Variety im Big-Data-Umfeld bedeutet. Es können nicht nur unstrukturierte Daten abgelegt und verwaltet, sondern auch zu Auswertungszwecken bei Bedarf dynamisch in eine beliebig strukturierte Form überführt werden.

Schema-On-Read hat gleichwohl nicht nur Vorteile, sondern bringt auch einige Nachteile mit sich, die bei einer differenzierten Betrachtung genannt werden müssen. Zum einen kann dieses Vorgehen sehr rechenintensiv sein, da erst mit dem Select Statement das Dokument in die entsprechende Struktur überführt werden muss. Es wird folglich mit jedem Lesevorgang eine Datentransformation durchgeführt, statt diese einmalig durchzuführen. Andernfalls sind die modernen Big-Data-Lösung wie Apache Hadoop durch ihre verteilten Architekturen darauf ausgelegt, exakt solche Probleme performant zu lösen. Ein weiterer Nachteil ist, dass die Daten ohne eine Struktur nicht selbsterklärend sind bzw. es für den Analysten aufwendiger ist, Inhalt und Bedeutung dieser Daten zu klären. Abschließend ist der Schema-On-Read-Ansatz in vielen Fällen sehr entwicklungsintensiv, da die einzelnen Schema-Schichten erst erstellt werden müssen, bevor sie im Reporting verwendet werden können. (Tom Deutsch 2013)

Daten können mittels flachet Dateien (txt, csv, ocr) in einen Hive Datastore geladen oder aber mittels Standard-JDBC- oder ODBC-Schnittstelle per SQL-Insertbefehl eingefügt werden. Da Hive Daten strukturiert in Tabellen und Datenbanken ablegt, liegt auch der Gedanke nahe, Datenbestände relationaler Datenbankmanagementsysteme nach Hive zu übernehmen. Für diese Funktion steht innerhalb von Hive jedoch keine Standardschnittstelle vor, sodass hierfür ein alternatives Werkzeug verwendet werden muss. Dieses Werkzeug trägt den Namen *Sqoop* und wird im nächsten Absatz vorgestellt. (Rajesh Kartha 2012)

4.5.3 Sqoop

Apache Sqoop (Kurzform von „SQL to Hadoop“) ist ein Werkzeug für den Import sowie Export großer Datenmengen nach Hadoop oder aus Hadoop heraus. Es handelt sich um ein Kommandozeilen-Tool, das die strukturierten Datenquellen mit Apache Hadoop verbindet. Sqoop arbeitet dabei bidirektional und kann sowohl Daten aus relationalen Datenbanken nach HDFS oder in mit Hive erstellte Tabellen importieren als auch umgekehrt Daten aus beispielsweise MapReduce-Jobs zurück in SQL-Datenbanken transferieren. Hierbei setzt Sqoop das Vorhandensein einer Tabelle in Apache Hive nicht voraus, sondern kann diese auch zur Laufzeit selbstständig aus

den Metadaten des Quellsystems anlegen. Gesteuert wird Sqoop über entsprechende Argumente auf der Kommandozeile, um etwa ganze Tabellen, einzelne Spalten oder auch Abfrageergebnisse aus SQL-Datenbanken zu importieren. Technologisch übersetzt der Sqoop Client die Transferaufgabe hierzu in einen MapReduce-Job, der lediglich aus der Map-Phase besteht und die Steuerung des Datentransfers übernimmt. Hierdurch kann Sqoop die von MapReduce bekannten Parallelisierungsverfahren und so die entsprechenden Vorteile in der Geschwindigkeit nutzen.
Sqoop verfolgt architektonisch einen modularen Ansatz, wodurch neben allgemeinen JDBC-Konnektoren auch datenbankspezifische Konnektoren implementiert werden können. Hierdurch können neben klassischen relationalen Datenbanken auch andere Big-Data-Systeme wie Terdata- oder NoSQL-Datenbaken angebunden werden. (Heise Medien GmbH & Co. KG 2015, S. 40)

Für eine vereinfachte Bedienung steht in der aktuellen Version (Sqoop 2) neben der Kommandozeile ein weiteres Userinterface in Form einer Webapplikation zur Verfügung. Hierüber können über eine grafische Benutzeroberfläche Datenimporte und -exporte erstellt und gesammelt als Job geplant werden. (Apache Software Foundation / Cloudera Inc. 2012)

4.5.4 Pig

Neben Apache Hive trägt Apache Pig einen wesentlichen Teil zu den Datawarehouse-Komponenten von Apache Hadoop bei. Der Name Pig (engl. Schwein) versinnbildlicht in diesem Fall die Möglichkeiten von Pig, da es scheinbar mit jeder Art von Daten umgehen kann und somit ein echter Allesfresser ist, also wie ein richtiges Schwein. Pig stellt ein Softwareframework für das Laden, Transformieren und Verarbeiten von Daten dar, wodurch es das T und das L aus dem ETL (Extract, Transform, Load) umfasst. Mit Pig Latin stellt es eine High-Level-Sprache zur Beschreibung von Datenflüssen dar, die eine Skript-ähnliche Syntax zur Verfügung stellt und somit schnell erlernt werden kann. Die erstellten Skripte werden durch den mitgelieferten Compiler in einen MapReduce-Ausführungsplan (seit Version 0.14 wird auch Tez als Execution Engine unterstützt) übersetzt, der anschließend auf dem Hadoop-Cluster ausgeführt werden kann. Durch die Abstraktion kann sich der Entwickler auf das Erstellen des Datenflusses konzentrieren und muss im Vergleich zu MapReduce deutlich weniger Zeilen Java-Programm-Code schreiben. Durch diese Abstraktion werden gleichzeitig auch einige Freiheiten in der Entwicklung aufgegeben, da nicht mehr auf jedes Detail in der Verarbeitung Einfluss genommen werden kann, was jedoch in den meisten Fällen ohnehin nicht notwendig ist. Darüber hinaus steht mit Pig standardmäßig eine Vielzahl von vordefinierten Bibliotheken zur Verfügung, die benutzerdefinierte Funktionen (User-Defined-Functions, kurz: UDF) beinhalten und in Pig-Skripte eingebunden werden können. Über diese Funktionen stehen verbreitete Algorithmen und Hilfsfunktionen für Datenprobleme, etwa aus den Bereich Statistik, Schätzung

und Mengenoperationen, Stichprobenverfahren, Linkanalysen sowie einigen weiteren Bereichen bereit. (Heise Medien GmbH & Co. KG 2015, S. 38)

Neben bereits bestehenden Funktionen können auch eigene Funktionen geschrieben werden, die in einer eigens dafür vorgesehenen Datenbank namens Piggybank abgelegt werden können. Mit Grunt steht darüber hinaus eine interaktive Shell für das Erstellen von Pig-Skripten in der Sprache Pig Latin zur Verfügung. Pig Latin ist darauf optimiert, Transformationen auf Daten auszuführen, wozu viele der typischen Datenoperationen wie Vereinigungen (Joins), Sortierungen (Sorts) und Filter zur Verfügung stehen.
Ein Programm in Pig ist dabei immer so aufgebaut, dass im ersten Schritt die zu verarbeitenden Daten adressiert (Load) und anschließend im zweiten Schritt transformiert (Transform) werden. Das Ergebnis dieser Transformation kann ausgegeben oder wahlweise gespeichert werden, um beispielsweise in einem weiteren Schritt weiterverarbeitet zu werden. Aus einem Pig-Skript können so implizit mehrere MapReduce-Jobs entstehen, wodurch auch komplexere Abläufe abbildbar sind. Die so entstehenden Abläufe werden als Daten-Pipelines bzw. als Dataflows bezeichnet. Die folgende Abbildung visualisiert beispielhaft einen solchen Dataflow. (IBM Big Data & Analytics)

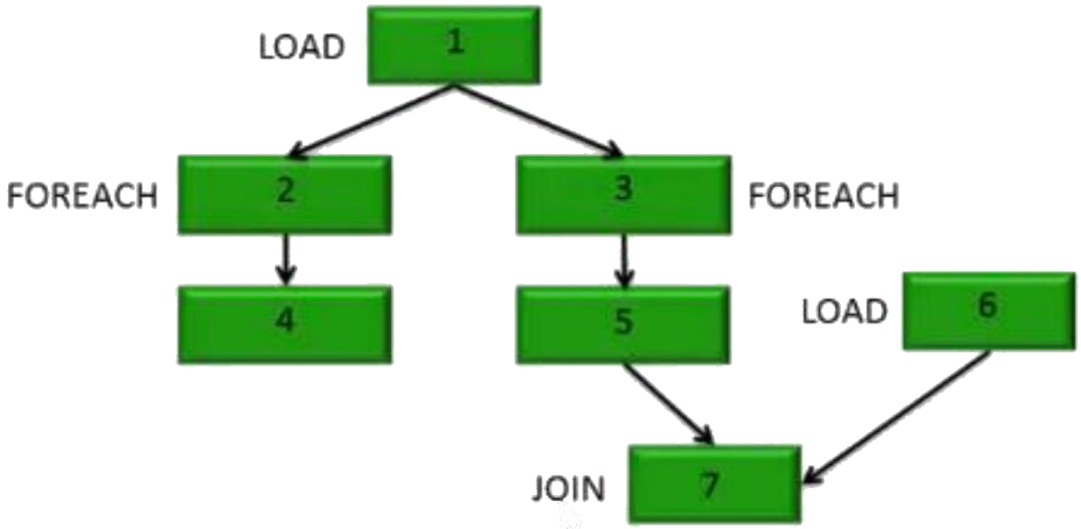

Abbildung 30: Beispielhafter Ablauf eines Pig-Dataflows (Hortonworks Inc. 2013)

Pig wird aufgrund seiner Fähigkeiten verbunden mit der Möglichkeit, auf einfache Weise typische Datenoperationen und -transformationen durchführen zu können, häufig als das ETL-Werkzeug im Hadoop-Ökosystem bezeichnet. Dieser Umstand wird dadurch gefördert, dass viele klassische ETL-Werkzeuge, die in der Lage sind, HDFS-Dateien zu verarbeiten, ihre Dataflows in native Pig-Skripte übersetzen und direkt auf dem jeweiligen Hadoop-Cluster ausführen.

4.5.5 Kylin

Im September 2015 hat eBay mit Apache Kylin eine weitere Komponente zum Hadoop-Ökosystem dazugesteuert. Auch wenn Kylin zum aktuellen Zeitpunkt noch in einer sehr frühen Version (1.2) vorliegt und somit noch am Anfang seines Lebenszyklus steht, ist eine Betrachtung der Software in Hinblick auf die Themenstellung der vorliegenden Arbeit äußerst interessant. Es handelt sich bei Kylin um eine OLAP-Engine auf Basis von Apache Hadoop und führt somit die seit vielen Jahren in der Auswertung etablierten OLAP-Cubes in die Welt von Apache Hadoop ein. Zu Beginn des Hadoop-Grundlagenkapitels wurde Apache Hive als die Data-Warehouse-Lösung im Umfeld von Apache Hadoop vorgestellt. Diese Aussage hat auch mit der Einführung von Apache Kylin weiterhin Gültigkeit, da es zum einen auf Hive aufbaut und zum anderen den letzten Bereich in einer klassischen DWH-Architektur abdeckt, der durch Hive bislang nicht abgedeckt werden konnte. Hierzu bietet es sich an, noch einmal eine Muster-Data-Warehouse-Architektur mit dem Fokus auf die spezialisierte Datenbereitstellung zu betrachten. Die folgende Grafik zeigt eine solche Architektur und hebt die beiden Komponenten Data Marts und OLAP-Cubes hervor.

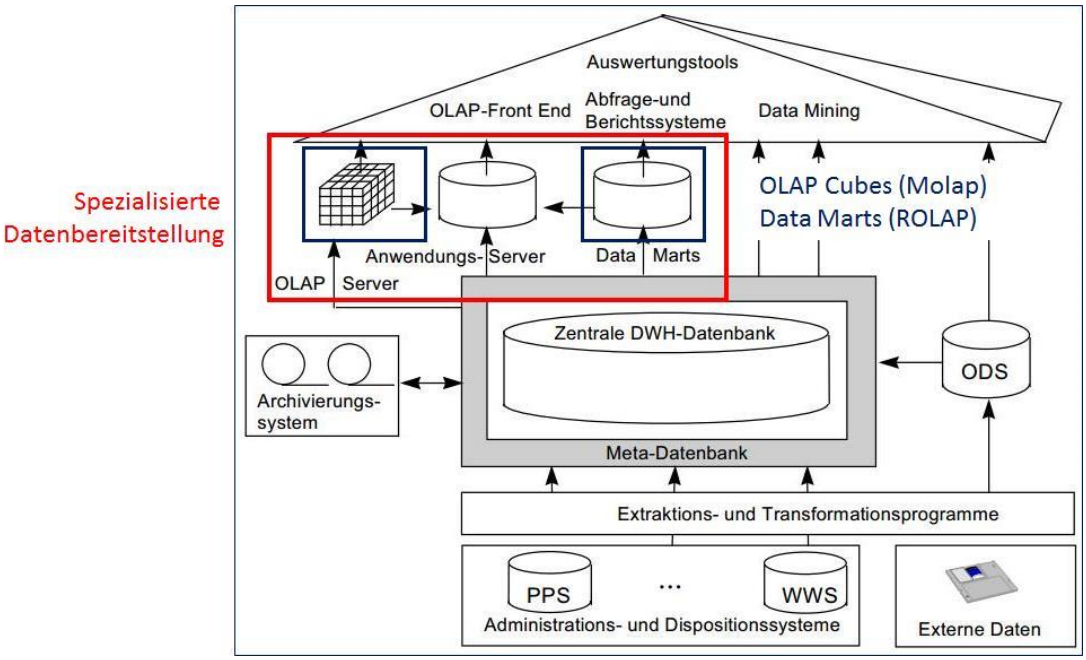

Abbildung 31: Spezialisierte Datenbereitstellung, OLAP Cubes und Data Marts im DWH (Mucksch, H., Behme, W. 2000, S. 14)

Sowohl im Fall von Data Marts als auch OLAP-Cubes handelt es sich in der Regel um ein OLAP-Verfahren, die sich in ihrer Speichertechnik unterscheiden. In beiden Fällen wird ein für Reportingaufgaben optimiertes Datenmodell, wie etwa ein Stern- oder Galaxy-Schema, verwendet. Liegt dieses Datenmodell in einem relationalen Kontext vor, so wird von ROLAP (Relational OLAP) gesprochen. Dem gegenüber ste-

hen die MOLAP-Systeme (Multidimensional OLAP). Hierbei liegen die Daten voraggregiert in einer multidimensionalen Datenbank. Dieser zusätzliche Aufwand ermöglicht während der Analyse eine kürzere Ausführungsdauer, führt jedoch gleichzeitig zu einem größeren Speicherplatzverbrauch, da die Aggregationen für jede Kombination der Dimensionen vorauszuberechnen sind. Eine solche Dimensionskombination wird als Cuboid bezeichnet, deren Anzahl exponentiell mit der Anzahl der Dimensionen in einem Cube steigt.

Neben reinen ROLAP- oder MOLAP-Systemen gibt es auch Hybridsysteme, die Vorteile beider Verfahren miteinander kombinieren. Zu einem solchen System gehört auch Apache Kylin. Hierzu legt Kylin ausgewählte Cubes als Cuboids in der Key-Value-Datenbank HBase ab. Eingehende Anfragen werden zunächst an die Cubes geleitet und von dort aus beantwortet. Stehen nicht alle angeforderten Dimensionen in den Cubes zur Verfügung, wird die entsprechende Anfrage an das zugrunde liegende Hive-Datenmodell weitergeleitet. Da die Antwortzeiten aus den Hive-Tabellen höher ausfallen, muss der zuständige Architekt der Cubes die Anforderungen der Datenanalysten bestmöglich berücksichtigen, um möglichst viele Anfragen über die Cubes beantworten zu können. Die nachfolgende Abbildung zeigt einen solchen Anfrageprozess auf Apache Kylin. (Sébastien Jelsch)

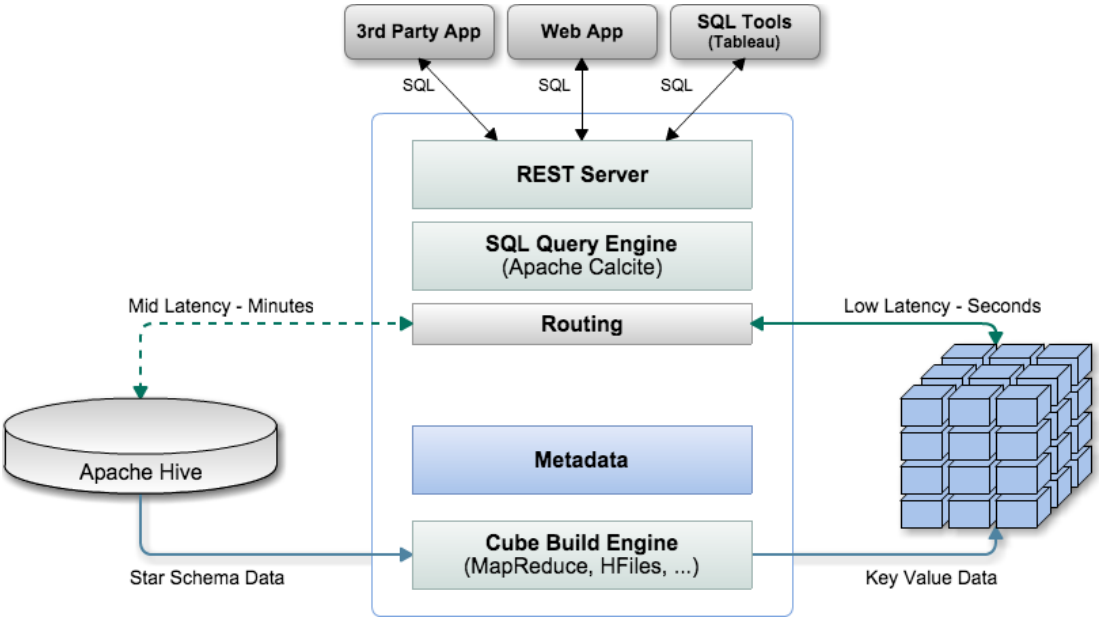

Abbildung 32: Kylin-Architektur mit Komponenten aus dem Hadoop-Ökosystem (Sébastien Jelsch)

Neben Apache Hive zur Modellierung von relationalen Data-Warehouse-Architekturen steht seit September 2015 mit Apache Kylin ein Werkzeug zur multidimensionalen Modellierung von Abfragen im Umfeld von Hadoop zur Verfügung. Diese Entwicklung schließt zum einen eine Lücke im Business-Intelligence-Produktportfolio von Apache Hadoop, zeigt zum anderen jedoch auch den Trend in diesem Umfeld, insbesondere

neue Werkzeuge zur Analyse von Big Data zu entwickeln. Hierbei werden nicht nur grundlegend neue Ansätze entwickelt, sondern auf bewährte Ansätze, wie die Speicherung von Analysedaten in OLAP-Cubes, zurückgegriffen, die mit den Vorteilen der horizontalen Skalierbarkeit von Hadoop kombiniert werden.

4.6 Stärken von Hadoop

Bevor das Grundlagenkapitel zu Apache Hadoop vollständig geschlossen wird, sollen zunächst noch einmal kurz die Stärken von Apache Hadoop herausgestellt werden.

Zunächst wäre dort die Möglichkeit, sehr große Datenmengen halten und verteilt verarbeiten zu können. Dieser Punkt wird durch die problemlose und flexible Skalierbarkeit eines Hadoop-Clusters bekräftigt. Der aber zunächst von vielen Lesern stark unterschätzte Aspekt ist die hohe Flexibilität, die das Ökosystem zur Verfügung stellt. Mit den einzelnen Software-Projekten rund um Hadoop steht ein breites Spektrum an Applikationen zur Verfügung, mit denen eine Vielzahl von Problemen auf strukturierten sowie unstrukturierten Daten gelöst werden kann. Diese Vielzahl von Möglichkeiten stellt das eigentliche Potenzial von Hadoop dar. Bei rückblickender Betrachtung der Fallbeispiele aus dem Kapitel Big Data wird ersichtlich, dass Hadoop nicht nur dann eine Wahl sein kann, wenn viele Daten verarbeitet werden müssen, sondern insbesondere auch dann, wenn Probleme gelöst werden müssen, die nicht oder nur schwer mit klassischen Datenbanksystemen gelöst werden können.

5 Hadoop als Data-Warehouse-Plattform

5.1 Problemstellung

In den vorhergehenden Kapiteln wurde mit den Grundlagen zu den Bereichen Data Warehousing, Big Data und Apache Hadoop der Grundstock für das folgende Kapitel gelegt. Im Folgenden soll das Apache Hadoop Framework stärker hinsichtlich der zentralen Fragestellung betrachtet werden, inwiefern es als technologische Grundlage für die Realisierung eines Enterprise Datawarehouse geeignet ist. Hierzu soll das Framework zunächst gegen einige technische Anforderungen geprüft werden, die es in Hinblick auf das Grundlagenkapitel Data-Warehouse-Architektur erfüllen muss, um technologisch für eine Implementierung geeignet zu sein. Anschließend wird Hadoop aus einem architektonischem Blickwinkel untersucht, indem verschiedene Szenarien dahin gehend betrachtet werden, wie Hadoop in bestehende Data-Warehouse-Architekturen integriert werden kann.

5.2 Technische Überprüfung

Für eine Überprüfung, ob das Apache Hadoop Framework technologisch geeignet ist, als ein Data-Warehouse-System zu fungieren, müssen hierfür zunächst Kriterien ermittelt werden, gegen die die Lösung geprüft werden kann. Für die Ermittlung dieser Kriterien wird auf die Grundlagenkapitel zum Thema Data-Warehouse-Architektur und Data Warehouse (System) zurückgegriffen. Bei Betrachtung des nachfolgenden Schaubildes, das eine mögliche Data-Warehouse-Architektur darstellt, lassen sich drei zentrale Kriterien identifizieren, die grundlegend erfüllt werden müssen, damit ein System technologisch geeignet ist, um als Data-Warehouse-System zu fungieren.

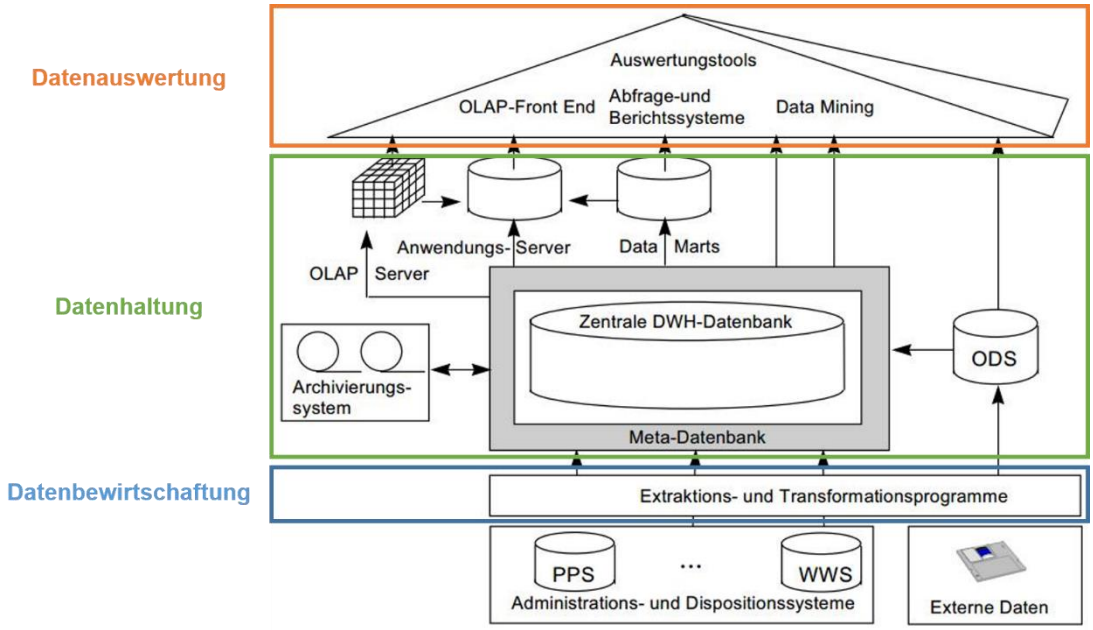

Abbildung 33: Zentrale Anforderungen an Apache Hadoop als ein DWH-System (Mucksch, H., Behme, W. 2000, S. 14)

5.2.1 Datenbewirtschaftung

Der Punkt Datenbewirtschaftung umfasst grundlegend sämtliche ETL-Aufgaben innerhalb einer Data-Warehouse-Architektur, kurzum also die Extraktion der Daten aus den Quellsystemen, die Transformation der Daten hinsichtlich den Zielsystemen und das anschließende Laden dieser Daten in die jeweiligen Datenhaltungskomponenten. Zu überprüfen ist nun, ob diese drei Aufgaben mit Softwaresystemen innerhalb des Hadoop-Ökosystems realisiert werden können.

<u>Extraktion</u>

Die Datenextraktion beschreibt das Anbinden und Laden von Daten unterschiedlicher Quellsysteme und anderer Datenquellen in ein Data Warehouse bzw. in die Staging Area eines Datawarehouse. Bei Betrachtung der Grundlagen zu Hadoop kann die Frage, ob Hadoop diese Aufgaben übernehmen kann, grundsätzlich mit Ja beantwortet werden. Ein Leser aus einer klassischen relationalen Datenbankwelt wird hierbei an das vorgestellte Apache Sqoop denken. Sqoop ist die Schnittstelle von Hadoop, welche mittels generischer JDBC-Datenbanktreiber Daten aus fast allen gängigen relationalen Datenbankmanagementsystemen lesen und auch zurückschreiben kann. Sqoop unterstützt dabei sogar nicht nur das vollständige Extrahieren von Tabellen und das anschließende Übertragen nach Hadoop, sondern besteht auch aus Deltamechanismen. Im Beispiel extrahiert das folgende Skript nur Daten aus einer Quelltabelle, deren Änderungszeitpunkt größer ist als der übergebene Parameter *{last_import_date}*.

sqoop import --connect jdbc:teradata://{host name or ip address}/Database=retail --connection-manager org.apache.sqoop.teradata.TeradataConnManager --user-name dbc --password dbc --table SOURCE_TBL --target-dir /user/hive/incremental_table -m 1--check-column modified_date --incremental lastmodified --last-value {last_import_date}

Das im Beispiel veranschaulichte Verfahren ist nur eine von vielen weiteren Möglichkeiten, mittels derer Sqoop Deltadatensätze identifizieren und extrahieren kann.

In vielen Data-Warehouse-Systemen dienen neben Datenbanken auch Flat Files, also CSV-Dateien, oder ähnliche Formate als Datenquelle. Diese Quellen stellen in Hinblick auf die Architektur von Hadoop quasi die essenzielle Funktion von Hadoop dar. Bei HDFS handelt es sich um ein virtuelles Dateisystem, in das zunächst einmal jegliche Dateien gespeichert werden können. Hierbei kann HDFS mittels NFS sogar als ein Standardnetzwerkverzeichnis auftreten. Quelldateien können also unmittelbar mittels Standard-Kopieroperationen direkt nach HDFS geschrieben werden, ohne dass der Quellapplikation bekannt ist und ohne dass es sich bei dem Zielverzeichnis um ein virtuelles, verteiltes Dateisystem handelt. Somit können selbst Daten aus Legacysystemen, die beispielsweise nur einen Dateiexport unterstützen, nach HDFS exportiert werden. Einmal im HDFS gespeichert, können die Daten beliebig mittels MapReduce, Pig oder Hive weiterverarbeitet werden. An dieser Stelle stellt Hadoop sogar einen Vorteil gegenüber klassischen Data-Warehouse-Systemen dar. In Hinblick auf den Punkt Variety (Datenvielfalt) können auch unstrukturierte Daten in ein Data-Warehouse-System geschrieben werden. Diese Daten können zu einem späteren Zeitpunkt dank Schema-On-Read in beliebiger Form ausgewertet und transformiert werden, ohne dass die ursprüngliche Quellinformation verloren geht.

<u>Transformation</u>

Die Anforderung der Transformation von Daten bezieht sich darauf, Daten innerhalb eines Datawarehouse in eine Reporting-optimierte Funktion zu überführen. Der Prozess der Transformation und die Bedeutung des Punktes sind im Grundlagenkapitel zum Punkt Transformation umfangreich beschrieben. Aus technischer Sicht betrachtet stellt der Punkt Transformation folgende Anforderungen: Zum einen müssen Daten von einer bestehenden Form in eine andere Form überführt werden, um einer gewünschten Zielstruktur zu entsprechen. Dies können beispielsweise die Umwandlung einer Zahl in einer Zeichenkette oder die Extraktion von Monats- und Jahresangaben aus einem Datum sein. Neben der reinen Umwandlung ist es häufig notwendig, dass Daten angereichert, also mit anderen Daten kombiniert werden. Eine solche Anreicherung wäre beispielsweise die Kombination von Ist- und Planzahlen in einer Tabelle, sodass diese in einem gemeinsamen Kontext betrachtet werden. Der dritte Bereich betrifft die Verdichtung bzw. die Aggregation von Daten. In vielen Fällen ist es

für ein Berichtswesen nicht notwendig oder sogar schlicht hinderlich, Daten auf kleinster Granularität abzulegen. Werden z. B. Planzahlen nur auf Monaten erfasst, während Ist-Zahlen jedoch auf Tagesbasis vorliegen, müssen diese für eine erfolgreiche Kombination zunächst auf Monatswerte verdichtet werden.

Apache Hadoop bietet für diese Aufgabe prinzipiell eine Vielzahl von Möglichkeiten. Die offensichtlichste wäre die Erstellung von MapReduce-Programmen, innerhalb derer jedoch die gewünschte Aktion durchgeführt und auf kleinster Eben gesteuert werden kann. In vielen Fällen ist dieses jedoch zum einen zu aufwendig und zum anderen nicht erforderlich. Die für diese Aufgabe besser geeigneten Werkzeuge stellen Hive und Pig dar. Liegen die Daten in Hadoop als Hive-Tabellen vor, können viele Operationen unter der Verwendung von HiveQL durchgeführt werden. Klassischerweise beträfe dies die Kombination und Anreicherung von Daten, welche sich als klassische Join-Operation zweier Tabellen tarnt, sowie die Aggregation derselben. Mittels gängiger Projektionsfunktionen wie *SUM()* oder *AVG()* in Kombination mit *group by* können Daten beliebig verdichtet werden. Einfache Datentransformationen können ebenfalls per Hive durchgeführt werden, allerdings bietet sich an dieser Stelle ebenso Pig an. Pig hat gegenüber Hive den Nachteil, dass Pig mit Pig Latin eine eigene Skriptsprache verwendet, die zunächst erlernt werden muss. Da HiveQL einer gängigen SQL-Syntax entspricht und diese vielen Entwicklern im Data-Warehouse-Umfeld bekannt ist, entfällt dieser Punkt gegenüber Pig. Ein entscheidender Vorteil von Pig ist jedoch die Möglichkeit, viel mehr Datenanpassungen als in Hive vornehmen und sogar benutzerdefinierte Funktionen für wiederkehrende Aufgaben anlegen zu können. Pig stellt somit die Möglichkeit dar, Aufgaben zu lösen, die ausschließlich unter Verwendung von HiveQL nicht realisierbar wären, ohne jedoch auf die Verwendung von MapReduce zurückgreifen zu müssen.

<u>Laden</u>

Der Ladevorgang beschreibt die Integration der zuvor transformierten Daten in den bestehenden Datenbestand eines Data Warehouse. In Hinblick auf die Evaluierung von Apache Hadoop ist es wichtig, zunächst zu differenzieren, wohin die Daten geladen werden sollen. Hier lassen sich zwei Szenarien unterscheiden. Szenario 1: Die Daten verlassen den Hadoop-Kontext und werden in ein externes System in Form einer relationalen Datenbank oder eines Dateiexports geschrieben. Szenario 2: Die Daten verbleiben innerhalb eines Hadoop-Kontextes und werden in eine Hive-Tabelle verarbeitet.

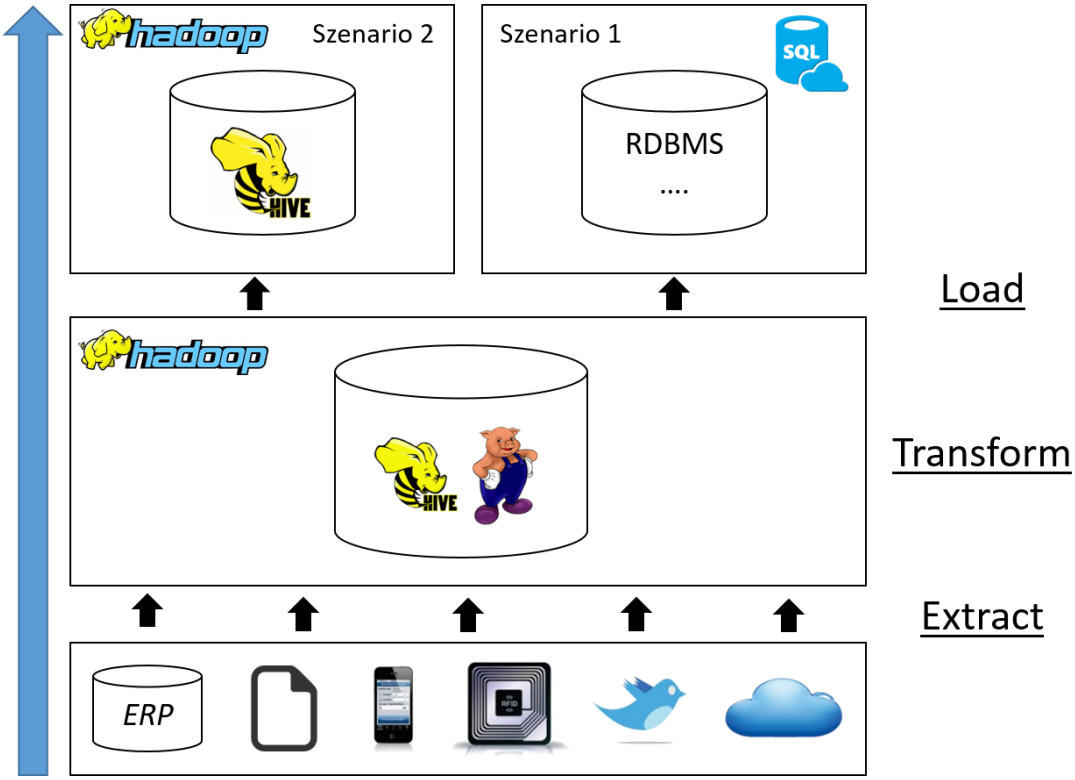

Abbildung 34: Data Warehouse unter Verwendung von Hadoop als zentrale Komponente

Szenario 1 lässt sich in zwei Teile separieren. In Teil 1 müssen neue Datensätze in die Zieldatenbank eingefügt werden. In Teil 2 existieren Teile der Datensätze bereits in der Zieltabelle, sodass diese aktualisiert werden müssen. Die beiden folgenden Grafiken visualisieren diese beiden Vorgehensweisen.

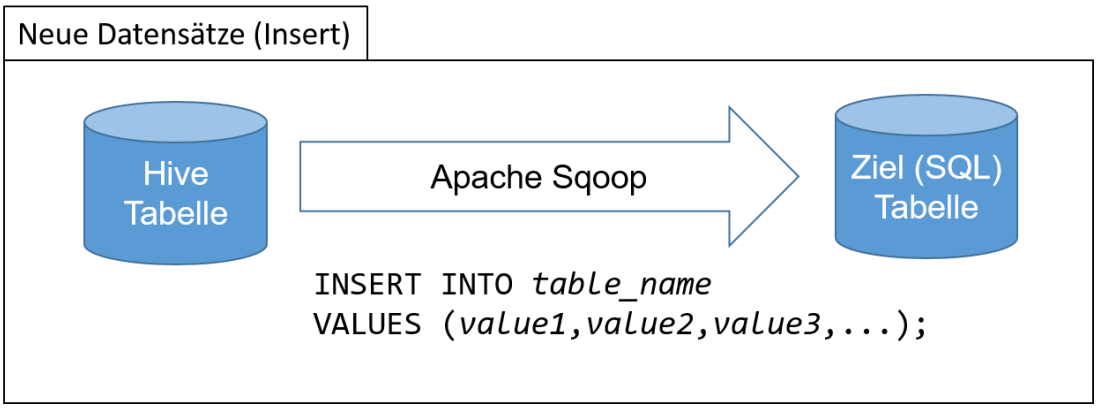

Abbildung 35: Laden, Szenario 1, neue Datensätze müssen eingefügt werden

In dem dargestellten Fall handelt es sich um neue Datensätze, die in der Zieltabelle noch nicht vorhanden sind. Diese können wie zuvor bei der Extraktion in umgekehrter Richtung mittels Sqoop direkt in die Zieldatenbank geschrieben werden. Es besteht hierbei die Annahme, dass bekannt ist, dass sich in der Quell-Hive-Tabelle nur neue Datensätze befinden. Das nächste Verfahren funktioniert ähnlich wie das vorherige, nur dass in diesem keine neuen Datensätze in die Zieltabelle geschrieben werden,

sondern bestehende Datensätze aktualisiert werden müssen. Die Grafik visualisiert zudem, wie zunächst ermittelt werden kann, welche Datensätze bereits in der Zieldatenbank existieren und somit aktualisiert werden müssen. Nachdem diese Ermittlung stattgefunden hat, können die Datensätze in der Zieldatenbank mittels Sqoop aktualisiert werden.

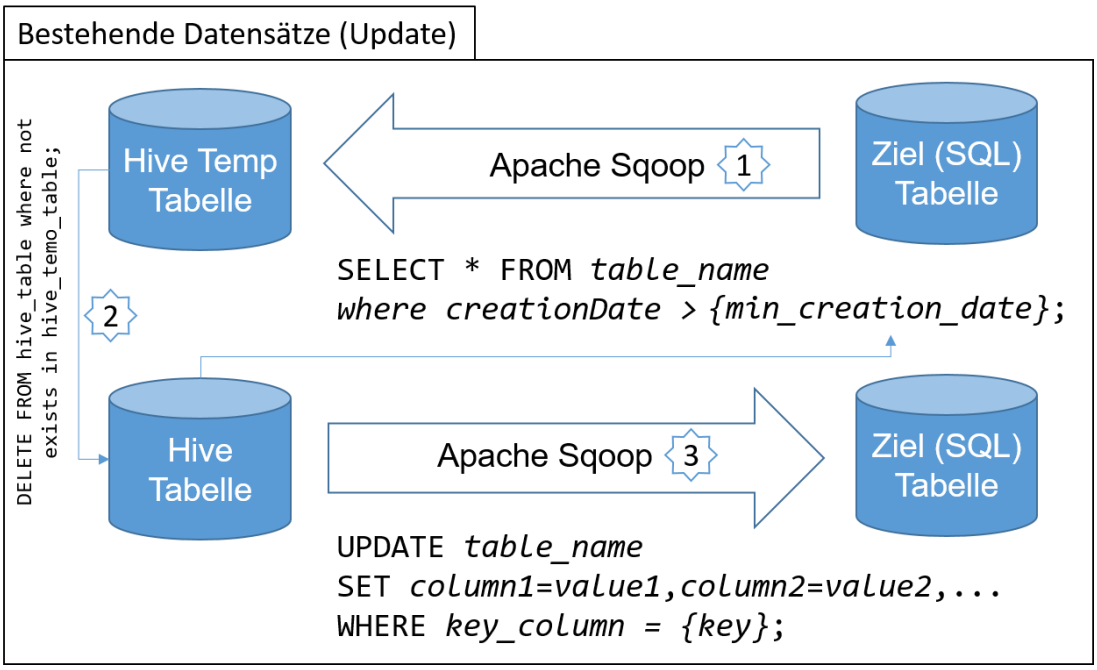

Abbildung 36: Laden, Szenario 1, bestehende Datensätze müssen aktualisiert werden

Alternativ kann in Szenario 1 auch ein Export mittels Flatfile in ein Zielsystem stattfinden. Liegt die zu exportierende Information bereits als Datei im HDFS vor, kann diese über ein einfaches Skript an die gewünschte Destination kopiert werden. Liegen die relevanten Informationen in einer Hivetabelle vor, müssen diese zunächst extrahiert und in eine Datei geschrieben werden. Hierzu eignet sich ein einfaches Pig-Skript, welches eine Datei im HDFS schreibt, die anschließend wie an den entsprechenden Ort kopiert wird.

Szenario 1 wird vielen Personen, die bereits Berührungspunkte mit Data-Warehouse-Systemen und ETL-Werkzeugen bekannt vorkommen. In vielen Fällen werden die einzelnen Schritte überhaupt nicht mehr im Detail implementiert werden müssen, da moderne ETL-Werkzeuge die einzelnen Schritte automatisiert übernehmen.

Szenario 2 unterscheidet sich in gewisser Weise grundlegend von Szenario 1. Hive wurde bislang als die Data-Warehouse-Komponente eines Hadoop-Ökosystems vorgestellt. Es wurde gezeigt, dass Hive viele Funktionen von klassischen relationalen Datenbanksystemen unterstützt und der SQL-Dialekt fast alle gängigen SQL-Funktionen unterstützt. Wichtig ist jedoch, im Hinterkopf zu behalten, dass Hadoop

bzw. Hive keine relationale Datenbank darstellt. Dies wird bereits in einem sehr einfachen Beispiel deutlich. Wie auch in Szenario 1 sollen Datensätze in einer bestehenden Hive-Tabelle aktualisiert werden. Diese zunächst sehr trivial anmutende Aufgabe stellt unter Apache Hive (bis Version 0.13) eine nicht zu unterschätzende Herausforderung dar. Grund hierfür ist, dass HiveQL keine Update Statements unterstützt. Im Folgenden wird anhand von vier Schritten beschrieben, wie ein SQL Update Statement im Kontext von Apache Hive aussehen könnte.

In dem Beispiel existiert eine Hive-Basistabelle (basis_table), die einmalig, z. B. per Apache Sqoop, aus einem Quellsystem initialisiert wurde. Die Tabelle hält somit zu Anfang des Aktualisierungsprozesses alle Datensätze des Quellsystems. Im Laufe des Verfahrens sollen nun Änderungen innerhalb des Quellsystems inkrementell in diese Tabelle überführt werden.

Neben der Basistabelle (base_table) sind hierfür drei weitere Tabellen notwendig: (Greg Phillips 2014)

- Inkrement-Tabelle (incremental_table): eine Hive-Tabelle, die alle geänderten Datensätze (Einfügungen und Aktualisierungen) aus dem Quellsystem hält. Diese Datensätze müssen in die Basistabelle überführt werden. Nach erfolgreicher Verarbeitung werden alle Daten in der Inkrement-Tabelle gelöscht.
- Berichtstabelle (repoting_table): eine Hive-Tabelle, die für das Berichtswesen verwendet wird. Die gespeicherten Daten sind statisch und bleiben bis zum nächsten Verarbeitungszyklus unverändert. Dies sorgt im Berichtswesen für Konsistenz zwischen Verarbeitungszyklen. Zudem wird die Berichtstabelle eines jeden Verarbeitungszyklus verwendet, um die Basistabelle zu überschreiben.
- Konsolidierungssicht (reconcile_view): Hierbei handelt es sich um eine View auf die Basis- und Inkrement-Tabelle. Die View wird hierbei so erstellt werden, dass sie jeweils nur den aktuellsten Datensatz zu einer ID anzeigt. Zudem wird aus der View die Berichtstabelle abgeleitet.
- Alle Tabellen bzw. Views weisen dabei die folgende Struktur auf:

  ID, string (Primärschlüssel)
  field1 - field5, string
  MODIFIED_DATE, string

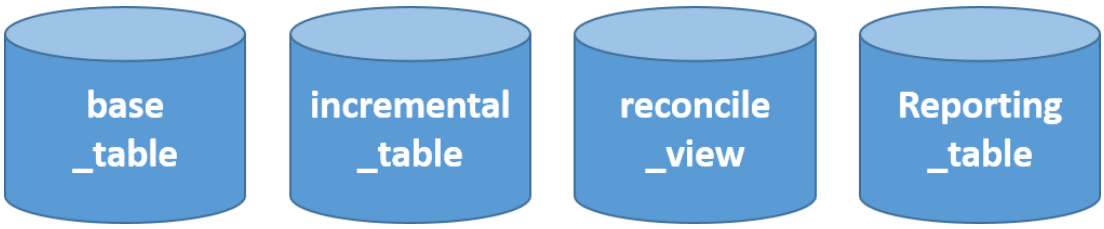

Abbildung 37: Hive Update – Vier-Schritt-Vorgehen – Tabellenübersicht

Ausgangslage: In der Basistabelle befindet sich der vollständige Datenbestand einer beliebigen Quelltabelle bis zu einem gewissen Zeitpunkt. Mittels Apache Sqoop wurden alle Datensätze aus der Quelltabelle in die Inkrement-Tabelle geladen, deren Änderungszeitpunkt dem Zeitpunkt in der Quelltabelle entspricht. (Greg Phillips 2014)

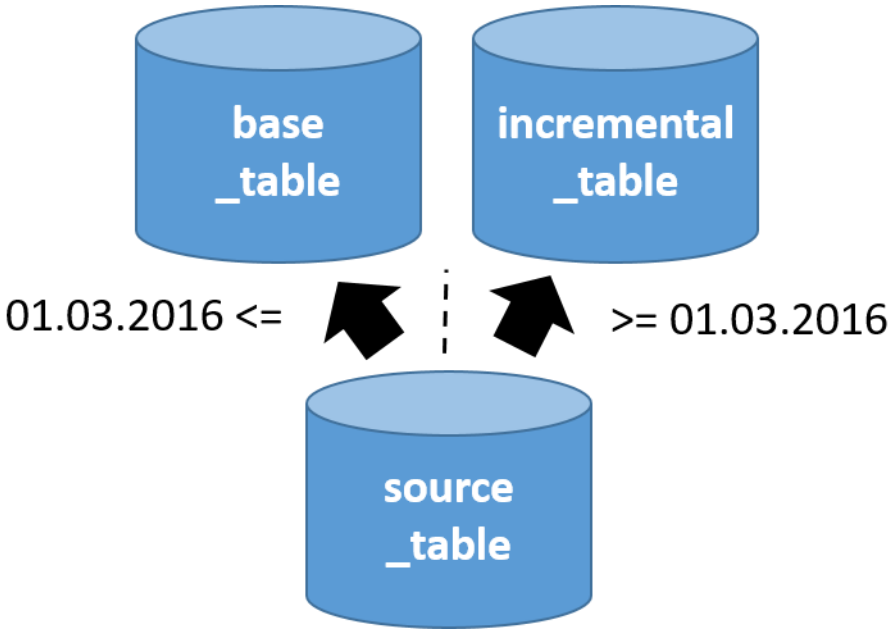

Abbildung 38: Hive Update – Vier-Schritt-Vorgehen – Ausgangslage

Ein vereinheitlichter Blick auf die Daten aus den beiden Tabellen könnte den folgenden Datenbestand anzeigen. Hierbei ist darauf zu achten, dass für die ID 1 und 2 sowohl Datensätze in der Basis- als auch in der Inkrement-Tabelle enthalten sind. (Greg Phillips 2014)

UNION: (SELECT * FROM base_table UNION SELECT * from incremental_table)

id,	*field1,*	*field2,*	*field3,*	*field4,*	*field5,*	*modified_date*	
1,	abcd,	efgh,	4,	7,	10.2,	2014-02-01 09:22:52	base_table
2,	abcde,	efgh,	5,	7,	10.2,	2014-02-01 09:22:52	base_table
3,	abcde,	efgh,	6,	7,	10.2,	2014-02-01 09:22:52	base_table
1,	abcdef,	efgh,	7,	7,	10.2,	2014-03-01 09:22:52	incremental_table
2,	abc,	efgh,	8,	7,	10.2,	2014-03-01 09:22:52	incremental_table

Abbildung 39: Hive Update – Vier-Schritt-Vorgehen – Daten aus Basis- und Inkrement-Tabelle (Greg Phillips 2014)

Im nächsten Schritt wird aus diesen beiden Tabellen eine Datenbankview erstellt, die zu allen IDs jeweils den aktuellsten Datensatz anzeigt. Für die Erstellung der View wird das folgende SQL-Skript verwendet. (Greg Phillips 2014)

```
1  CREATE VIEW reconcile_view AS
2      SELECT t1. * FROM
3          (SELECT * FROM base_table
4           UNION ALL
5           SELECT * FROM incremental_table) t1
6      JOIN
7      (SELECT id, max (MODIFIED_DATE) max_modified FROM
8          (SELECT * FROM base_table
9           UNION ALL
10          SELECT * FROM incremental_table) t2
11          GROUP BY id) s
12     ON t1.id = s.id UND t1.modified_date = s.max_modified;
```

Abbildung 40: Hive Update – Vier-Schritt-Vorgehen – erzeugte Konsolidierungsansicht (Greg Phillips 2014)

Ein Blick auf die Daten der Konsolidierung zeigt nun lediglich drei Datensätze im Vergleich zu den fünf Datensätzen der vorherigen Ansicht. (Greg Phillips 2014)

VIEW: reconcile_view

id,	*field1,*	*field2,*	*field3,*	*field4,*	*field5,*	*modified_date*
1,	abcdef,	efgh,	7,	7,	10.2,	2014-03-01 09:22:52
2,	abc,	efgh,	8,	7,	10.2,	2014-03-01 09:22:52
3,	abcde,	efgh,	6,	7,	10.2,	2014-02-01 09:22:52

Abbildung 41: Hive Update – Vier-Schritt-Vorgehen – Daten in der Konsolidierungsansicht (Greg Phillips 2014)

Die Konsolidierungsansicht (reconcile_view) enthält den jeweils aktuellsten Datensatz, basierend auf der Kombination der Felder ID und MODIFIED_DATE. Auf Basis dieser Tabelle könnte nun mittels Reporting Tools zugegriffen werden. Es bietet sich jedoch in einem solchen Fall an, die Daten der reconcile_view zu persistieren, da sonst bei jedem Zugriff zunächst die Konsolidierungsansicht aufgebaut werden müsste, was zulasten der Ausführungsgeschwindigkeit der Abfragen ginge. Zudem hat das Anlegen einer Berichtstabelle den Vorteil, dass aus dieser nachgelagert die Basistabelle befüllt werden kann. (Greg Phillips 2014)

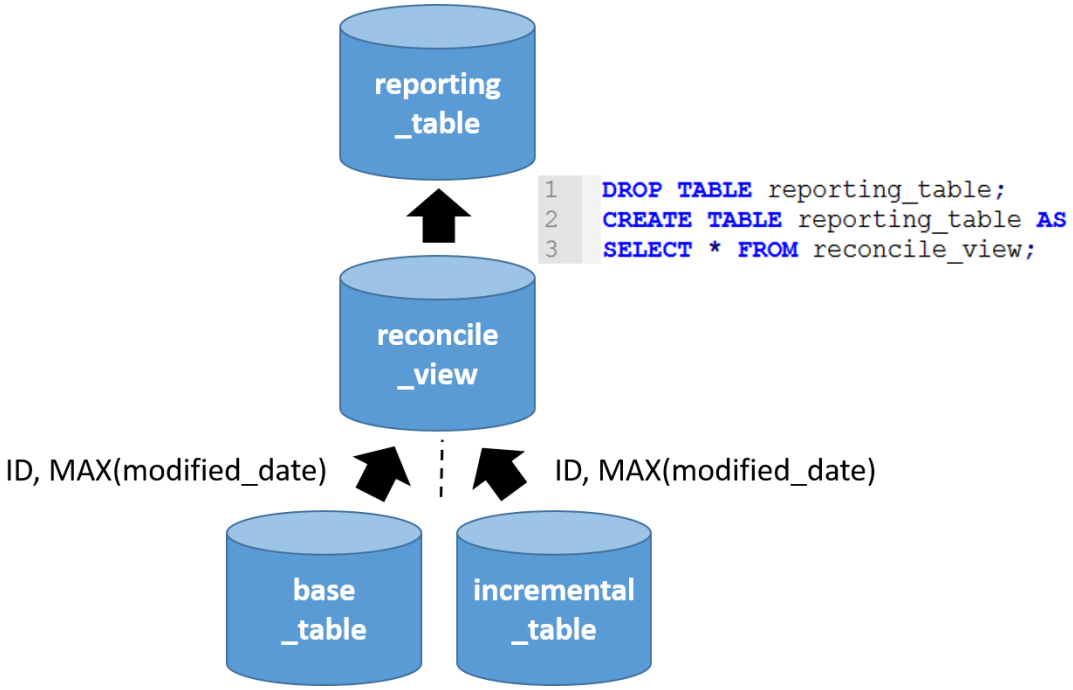

Abbildung 42: Hive Update – Vier-Schritt-Vorgehen – Persistieren der Konsolidierungsansicht

Im nächsten Schritt soll auf das eigentliche Ziel des gesamten Verfahrens zurückgekommen werden, nämlich die Übernahme der Aktualisierungen und Neueinfügungen aus der Quelltabelle in die Basistabelle. Zu diesem Zweck wurde eine Berichtstabelle angelegt, die den gewünschten Datenbestand der Basistabelle enthält. Der nächste Schritt zum Abschließen des gesamten Aktualisierungsvorgangs ist somit, die bestehenden Daten aus der Basistabelle zu löschen und die Daten der Berichtstabelle in die Basistabelle zu schreiben.

An dieser Stelle wird nochmals erkenntlich, dass es sich bei Hive bzw. Hadoop nicht um eine relationale Datenbank handelt. In einer relationalen Datenbank wäre es vorstellbar, dass die Basistabelle Bestandteil einer Fremdschlüsselbeziehung ist. In einem klassischen Datenmodell mit referenzieller Integrität wäre es somit nur schwer möglich, alle Daten zunächst zu löschen und anschließend wieder einzuspielen. Zwar besitzen einige Datenbankmanagementsysteme die Möglichkeit, Fremdschlüsselbeziehungen zeitweise zu deaktivieren, jedoch ist dies aus einer OLTP-Sicht schlicht undenkbar.

Im Anschluss an das Erstellen der Basistabelle aus der Berichtstabelle müssen anschließend noch die Daten aus der Inkrement-Tabelle gelöscht werden. Dies ist erforderlich, damit die Konsolidierungsansicht keine falschen Daten ausliefert, sollte sie dennoch einmal Quelle einer Abfrage sein, und das beschriebene Verfahren erneut angewandt werden kann. (Greg Phillips 2014)

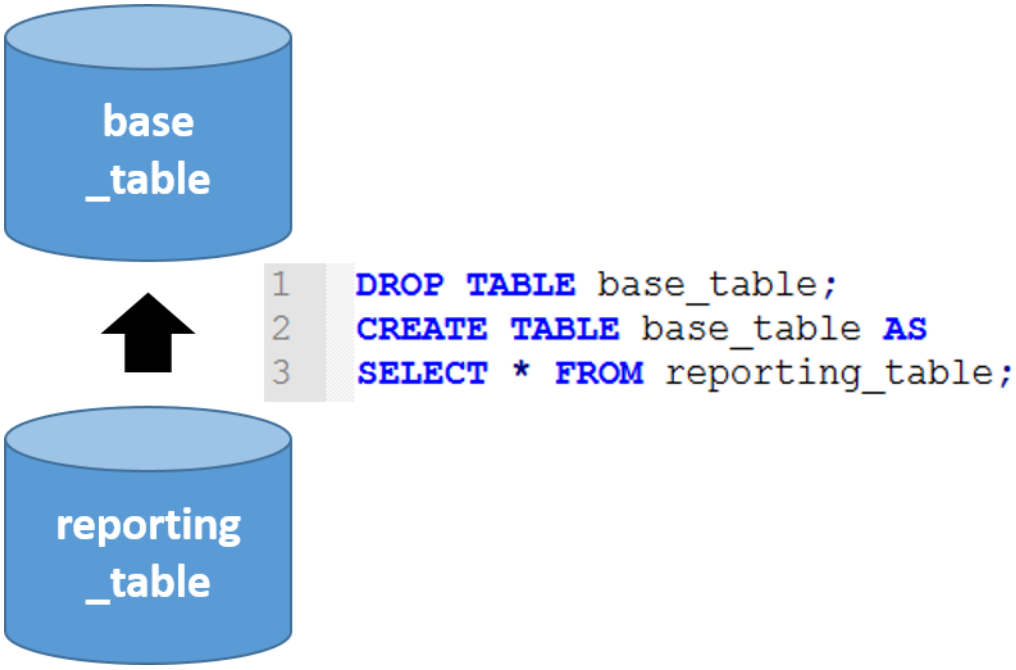

Abbildung 43: Hive Update – Vier-Schritt-Vorgehen – erstellte Basistabelle aus Berichtstabelle

Das sehr ausführlich dargestellte Beispiel wurde bewusst ausgewählt, um verständlich zu machen, dass Hadoop bzw. Hive nicht mit klassischen relationalen Datenbanksystemen gleichzusetzen ist. Bei der Betrachtung der drei Punkte Extraktion, Transformation und Laden wird ersichtlich, dass das Hadoop Framework durchaus zur Datenbewirtschaftung geeignet ist. Es muss indes bedacht werden, dass es sich an vielen Stellen anders verhält, als es zu erwarten gewesen wäre. Ob Hadoop eine Alternative zu klassischen Ansätzen darstellt, darf nicht nur aus rein technischer Perspektive betrachtet werden, sondern muss stets unter Berücksichtigung von Kontext und Zielsetzung erfolgen, in dem Hadoop eingesetzt werden soll. Die Aussicht, flexible Datenstrukturen (Variety) einsetzen und große Datenmengen (Volume) parallel verarbeiten zu können, kann einen Entwicklungsmehraufwand bzw. eine Anpassung bestehender Datenflüsse in vielen Aspekten rechtfertigen.

Im bisherigen Verlauf des Abschnitts wurde der Abschnitt Datenbewirtschaftung sehr abstrakt, wenn auch unter Verwendung konkreter Beispiele, betrachtet. Hierbei ist jedoch ein entscheidender Faktor bislang fast gänzlich außen vor gelassen worden. Der Prozess der Datenbewirtschaftung bzw. der ETL-Prozess wurde bisher so dargestellt, als handle es sich um eine Kombination vieler einzelner Systeme und Programme, die über manuell zu erstellende Skripte gesteuert und in Abhängigkeit zueinander ausgeführt werden. In der Praxis wird es jedoch schwer sein, ein Data-Warehouse-System zu finden, bei dem sämtliche ETL-Prozessschritte manuell erstellt und weiterentwickelt werden. Diese Aufgaben werden in der Realität in der Regel von sogenannten ETL-Werkzeugen übernommen oder zumindest durch diese hierbei unterstützt.

Bei einem ETL-Werkzeug handelt es sich um eine Software, die alle Prozesse rund um die Erstellung von ETL-Prozessen unterstützt und vereinfacht. Ein ETL-Werkzeug bietet klassischerweise die Möglichkeit, verschiedenste Datenquellen anzubinden und Daten zu extrahieren. Anschließend können die Daten transformiert und in ein wiederum beliebiges Ziel geladen werden. Ein ETL-Werkzeug abstrahiert hierbei jedoch die zugrunde liegenden Technologien von dem Entwickler und ermöglicht die Erstellung von Prozessen über eine einheitliche grafische Schnittstelle. Hierdurch wird dem Entwickler die Möglichkeit gegeben, verschiedene Datenquellen in sein Data Warehouse zu integrieren, ohne spezielle Programmiersprachen erlernen zu müssen. Darüber hinaus bieten ETL-Werkzeuge die Möglichkeit, Jobsteuerungen zu erstellen und somit abhängige Schritte automatisiert, sequenziell ausführen zu können.

Ein ETL-Werkzeug stellt in der Regel jedoch nicht nur das Modellierungswerkzeug an sich zur Verfügung, sondern verfügt in der Regel auch über eine eigene Serverkomponente, die vertikal zu allen Schichten eines Data-Warehouse-Systems einzuordnen ist. (Intricity)

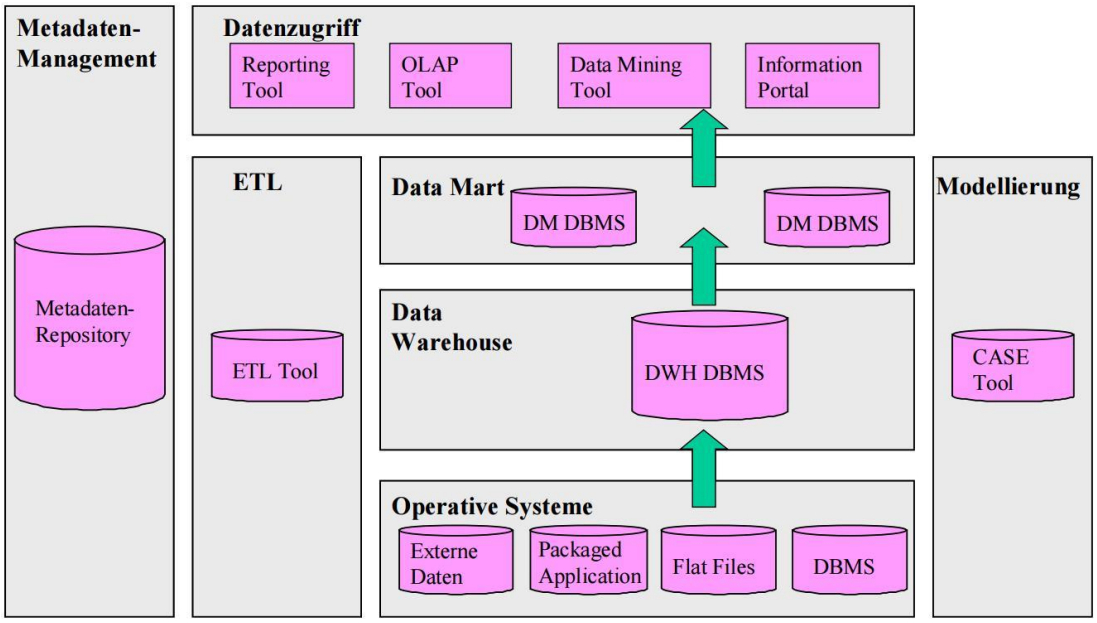

Abbildung 44: Vollständiges Data-Warehouse-Modell inkl. technischer Unterstützungssysteme (Do et al. 2000, S. 3)

Zu der Fragestellung, ob die Hadoop-Technologien für die Datenbewirtschaftung geeignet sind, kommt somit noch die Fragestellung hinzu, ob diese sich in bestehende ETL-Werkzeuge integrieren lassen. Diese Frage wird exemplarisch am Beispiel des SAP Data Services beantwortet.

Prinzipiell unterstützt Data Services die Integration von Hadoop-Datenquellen über zweierlei Wege. Zum einen kann Hive mittels ODBC wie jede andere beliebige Datenbank auch mittels spezieller ODBC-Treiber als Datastore konfiguriert werden. Dies bedeutet, dass auf dem ETL-Server ein spezieller ODBC-Datenbanktreiber für Apache Hive installiert wird. Das ETL-Werkzeug kann fortan wie gewohnt Abfragen und Inserts erstellen, die vom Treiber in Hive-kompatible SQL-Statements übersetzt werden. Darüber hinaus werden über den Treiber Metadaten vom Hive-Server abgerufen, sodass Tabelleninformationen und Strukturen in das ETL-Werkzeug importiert werden können.

Die folgende Abbildung zeigt beispielhaft zwei sogenannte Datastore-Konfigurationen im SAP Data Services. Beim oberen der beiden Einträge handelt es sich um einen klassischen MS-SQL-Server und beim unteren Eintrag um eine Verbindung zu einem Hive-Server. Für einen Entwickler verhalten sich beide Einträge auf den ersten Blick identisch, und er kann Hive-basierte Tabellen und Datenbank auf exakt dieselbe Art und Weise verwenden, wie er es z. B. vom MS-SQL-Server gewohnt ist.

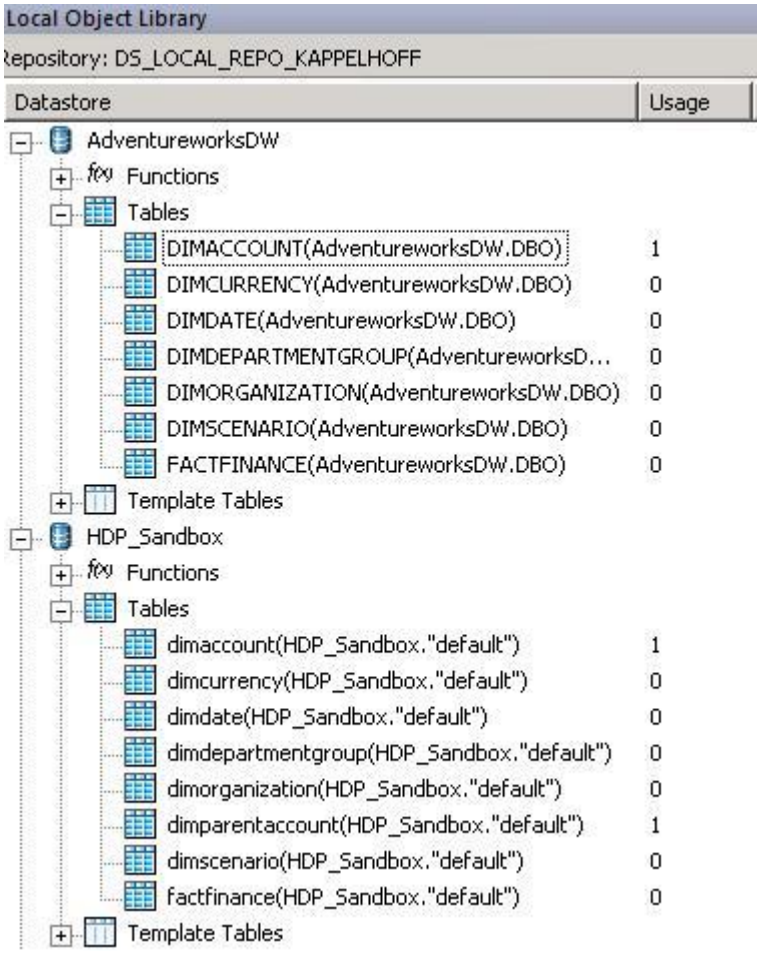

Abbildung 45: SAP Data Services – Darstellung von Hive-Datenquelle im Vergleich zu MS-SQL-Server

Neben der Einbindung von Hive-Datenquellen mittels der ODBC-Datenquelle besteht die Möglichkeit, einen spezifischen Hive-Adapter zu aktivieren. Dieser bietet gegenüber der Verwendung der ODBC-Schnittstelle den Vorteil, dass Operationen immer unmittelbar auf dem Hadoop-Cluster ausgeführt werden. Damit diese Option verwendet werden kann, muss der Data Services Jobserver jedoch Bestandteil des Hadoop-Clusters selbst sein, der Server also als ein Datanode konfiguriert sein. Damit diese Voraussetzung erfüllt werden kann, muss der Data Services Server auf einem Linux-basierten Betriebssystem ausgeführt werden. Apache Hadoop kann, mit einer Ausnahme, zum aktuellen Zeitpunkt ausschließlich auf Linux-basierten Betriebssystemen verwendet werden.

Die zweite Alternative, Hadoop in Data Services zu verwenden, bezieht sich auf die direkte Einbindung von HDFS-Dateien. Diese werden wie normale Flatfiles definiert, jedoch mit der Einschränkung, dass neben einem eindeutigen Verzeichnis auch ein Namenode angeben werden muss. Data Services sind also in der Lage, direkt auf HDFS-Verzeichnisse zuzugreifen und dort lesend und schreibend zu operieren. Eine wichtige Komponente bei der Konfiguration von HDFS-Dateien im Data Services kann zunächst schnell übersehen werden, ist allerdings von entscheidender Tragweite. Neben der Angabe eines Verzeichnisses bzw. eines Namenodes muss ein Pig-Arbeitsverzeichnis angegeben werden.

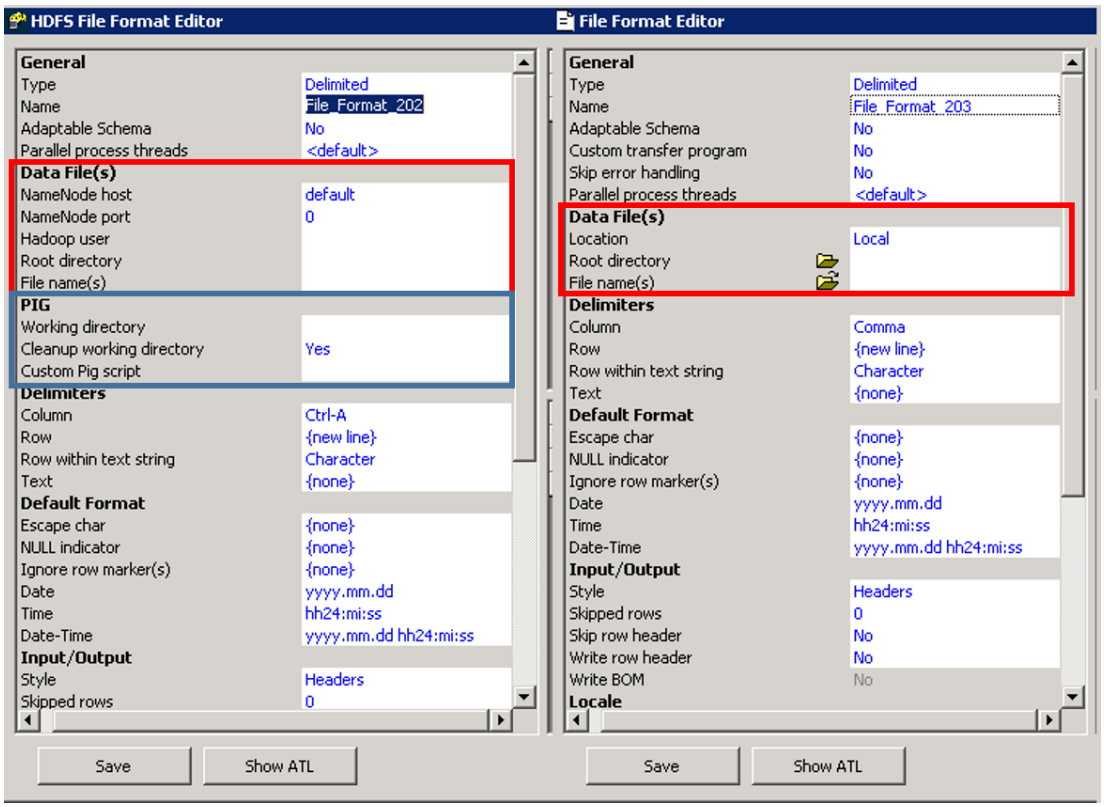

Abbildung 46: SAP Data Services – Anlage eines HDFS Flat Files gegenüber einem normalen Flat File

Dieser zunächst unscheinbare wirkende Punkt in der Konfiguration von HDFS Flat Files im SAP Data Services zeigt seine Tragweite erst dann, wenn dies im Kontext der Architektur von ETL-Werkzeugen betrachtet wird. ETL-Werkzeuge, wie etwa SAP Data Services, basieren auf einer klassischen Client-Server-Architektur. Das ETL-Werkzeug stellt dabei einen eigenen Server dar, der zu verarbeitende Dateien extrahiert, transformiert und anschließend in ein Zielsystem lädt. Die nachfolgende Abbildung zeigt exemplarisch, wie eine solche Architektur aussehen kann.

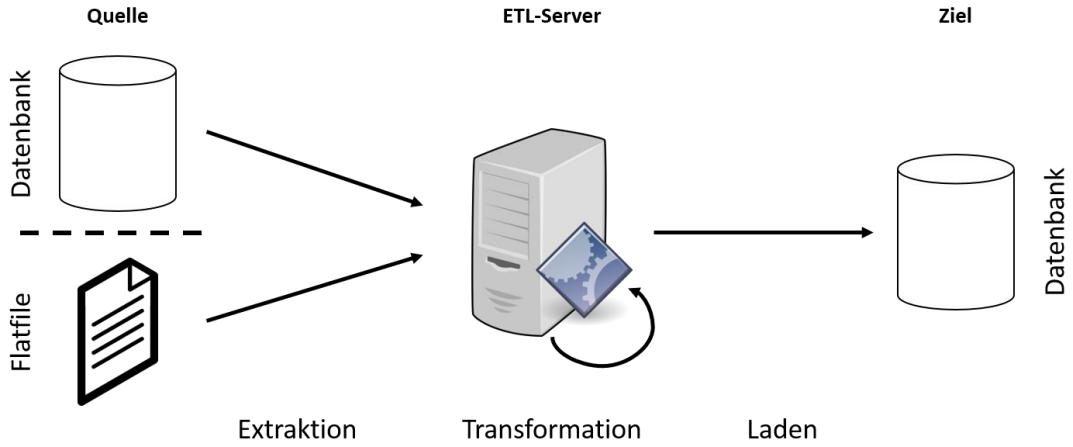

Abbildung 47: ETL – Beispiel Architektur

Das obige Schaubild ermöglicht die Schlussfolgerung, dass die Extraktion zunächst eine temporäre Ablage der Daten im ETL-Server bewirkt, bevor diese in diesem transformiert werden. Dieses Verfahren trifft insbesondere bei der Verarbeitung von Flat Files zu. Damit z. B. die Differenz zwischen zwei Datumsangaben in der Quelldatei berechnet werden kann, muss diese hierfür in eine Anwendung geladen werden, die hierzu in der Lage ist. In diesem Fall übernimmt die eingebaute Transformationskomponente des ETL-Servers diese Aufgabe. Nun stellt sich jedoch die Frage, was passiert, wenn es sich bei der zu verarbeitenden Datei nicht um eine einfache Datei in einem regulären Dateisystem handelt, sondern um eine HDFS-Datei, die sich über viele Knoten im Hadoop-Cluster erstreckt. HDFS ist in Hinblick auf den Punkt Volume für die Verwaltung sehr großer Datenmengen ausgelegt. Eine einzelne Datei im HDFS könnte sich somit theoretisch über viele Tausend Knoten in einem Cluster erstrecken und mehrere Hundert Gigabyte groß sein. Eine Transformation einer solchen Datei würde also bedeuten, die gesamte Datei zunächst auf den ETL-Server zu laden, um diese dort anschließend zu bearbeiten. Ein solches Vorgehen erscheint allein schon aus dem Grund nicht realistisch, da ein einzelner ETL-Server vermutlich nicht in der Lage ist, eine solche Datei zu speichern und zu bearbeiten, wenn auf der anderen Seite hierfür ein HDFS benötigt wird. An dieser Stelle wird ersichtlich, weshalb also bei der Einrichtung von HDFS-Dateien in SAP Data Services ein Pig-Ar-

beitsverzeichnis angegeben werden muss. Im Laufe des aktuellen Kapitels wurde beschrieben, dass Pig eine vollwertige ETL-Komponente innerhalb des Hadoop Frameworks darstellt. Die Angabe des Pig-Arbeitsverzeichnisses zielt genau hierauf ab, indem Transformationen nicht vom ETL-Server, sondern direkt von Pig im Hadoop-Cluster durchgeführt werden. Das ETL-Werkzeug bietet in diesem Fall lediglich eine grafische Schnittstelle, die Pig-Latin-Skripte erstellt und diese zur Ausführung auf das Hadoop-Cluster überträgt. Die Ausführung selbst erfolgt somit unter Verwendung der Parallelisierung- und Skalierungsmöglichkeiten von Hadoop direkt auf dem Cluster, ohne dass Daten auf den ETL-Server übertragen werden müssen. Dieses Verfahren findet nicht nur im SAP Data Services Verwendung, sondern ist ein gängiges Prinzip von ETL-Werkzeugen und wird als Push-Down bezeichnet. Ziel solcher Push-Down-Operationen ist die Auslagerung von daten- und rechenintensiven Prozessen auf die angeschlossenen, hierauf spezialisierten Reporting-Systeme. Die nachfolgende Abbildung visualisiert ein solches Push-Down-Verfahren exemplarisch am Beispiel von Hadoop.

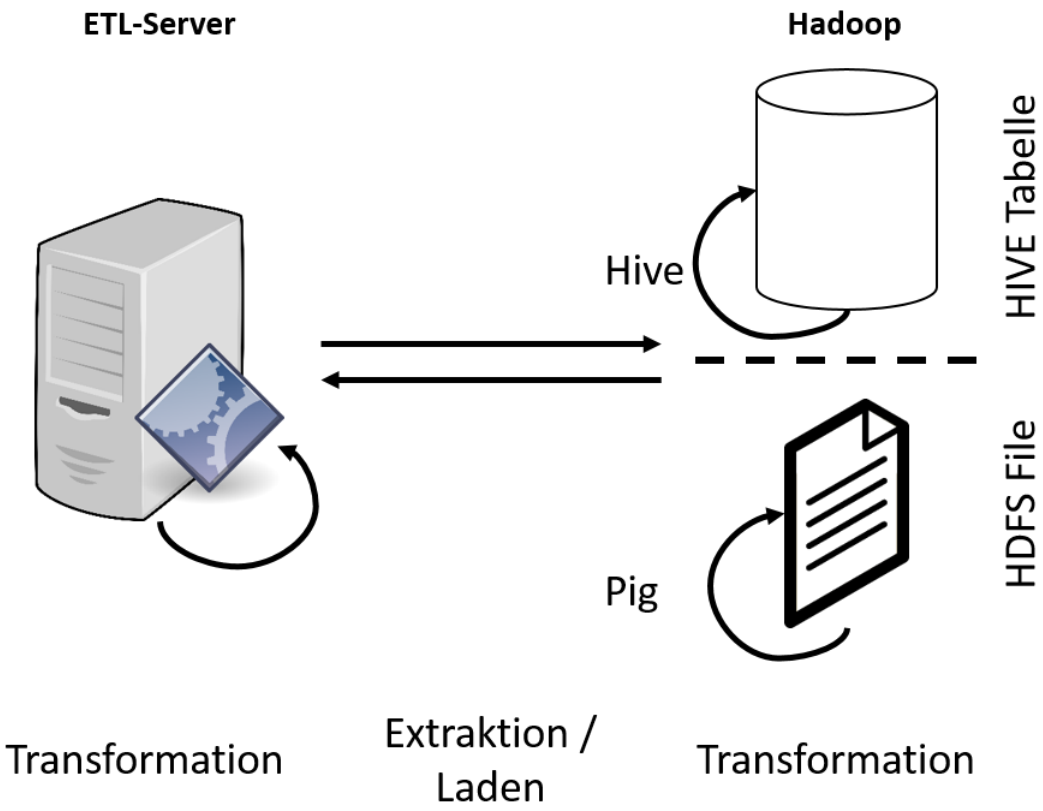

Abbildung 48: Push-Down von Transformationen

Da die Datentransformation nun nicht mehr wie in klassischen Data-Warehouse-Ansätzen extrahiert, transformiert und erst dann in das Zielsystem geladen wird, wird dieses Verfahren auch häufig als ELT anstelle von ETL bezeichnet.

Der Punkt Datenbewirtschaftung kann somit abgeschlossen werden, indem dem Apache-Hadoop-Ökosystem die Fähigkeit, ETL-Prozesse ausführen zu können, ganz klar zugesprochen wird. Dieses zeichnet sich sowohl durch die Verfügbarkeit diverser Werkzeuge wie Hive oder Pig als auch durch die Kompatibilität zu klassischen ETL-Werkzeugen wie SAP Data Services aus.

5.2.2 Datenhaltung

Der Punkt Datenhaltung bezieht sich auf die Fragestellung, inwieweit das Hadoop Framework in der Lage ist, die einzelnen Persistierungsaufgaben innerhalb einer Data-Warehouse-Architektur zu übernehmen. In den Hadoop-Grundlagen wurden mit HDFS und Hive zwei Verfahren zur Datenhaltung vorgestellt, wobei Hive auf HDFS basiert und somit immer in Kombination mit diesem betrachtet werden muss.

Zur Überprüfung, ob Hive und HDFS als Komponenten für die zentrale Datenhaltung geeignet sind, werden sie anhand der folgenden Kriterien geprüft:

– Sicherheit,

– Datensicherung/Wiederherstellung,

– Datenintegrität/Datenkonsistenz.

<u>Sicherheit:</u>

Innerhalb von Apache Hadoop sind vier Kernmechanismen zur Gewährleistung einer vollumfänglichen Sicherheit integriert. Der erste Mechanismus bezieht sich auf die Authentifizierung des Benutzers innerhalb des Hadoop Framework. Dieses basiert auf dem Kerberos-Standard und ist daher mit vielen beliebigen Quellen zur Benutzerverwaltung kombinierbar. Das Kerberos-Verfahren ermöglicht somit eine benutzerspezifische Anmeldung an Hadoop, das darüber hinaus aufgrund des Ticket-basierten Verfahrens gut mit der je nach Clusterlast zeitverzögerten Ausführung von Jobs harmoniert. Da neben direkten Aufrufen viele Dienste (wie etwa WebHDFS) im Hadoop-Ökosystem über HTTP oder REST aufgerufen werden, ist neben der Authentifizierung ein weiteres Sicherungsverfahren für die Perimeter-Sicherung notwendig. Diese Funktion wird von Apache Knox übernommen, welches mit dem Knox-Server eine Komponente zur Verfügung stellt, die als Mittelsmann zwischen den einzelnen Hadoop-Komponenten und dem externen Zugang fungiert. Der Knox-Server bildet somit einen klassischen Reverse-Proxy im Hadoop-Umfeld ab, der über eine URL und eindeutige Portangaben konsistente Pfade zu den einzelnen Hadoop-Anwendungen wie WebHDFS, YARN Hive oder HBase zur Verfügung stellt. Der Aufbau von Knox ist dabei modular, sodass zukünftige, noch nicht entwickelte oder angeschlossene Dienste in Zukunft ebenfalls über den Knox-Server erreicht werden können. Knox selbst ist vollständig über das Vorschalten eines Load Balancer skalierbar und wird somit nicht selbst zu einem Flaschenhals im Hadoop-Cluster. Knox bietet somit

die Vorteile für Administratoren dahin gehend, dass diese nur auf einem System eine SSL-Verschlüsselung konfigurieren müssen und das Anlegen von Benutzern über die Anbindung von Identity-Management-Systemen entfällt. Da die tatsächliche Infrastruktur des Hadoop-Clusters hinter dem Proxy verschleiert wird, bietet dies eine geringe Fläche für Angriffe auf das Hadoop-System und trägt folglich zur gesamten Sicherheit des Systems bei. Neben den Sicherheitsaspekten bietet Knox darüber hinaus den Vorteil, dass Benutzer bzw. Entwickler sich lediglich nur noch eine URL und ein Login für den Zugriff auf das Hadoop-Cluster kennen müssen. Trotz des Einsatzes einer zusätzlichen Komponente reduziert der Einsatz von Knox somit die Gesamtkomplexität von Hadoop-Systemen. (Heise Medien GmbH & Co. KG 2015, S. 42)

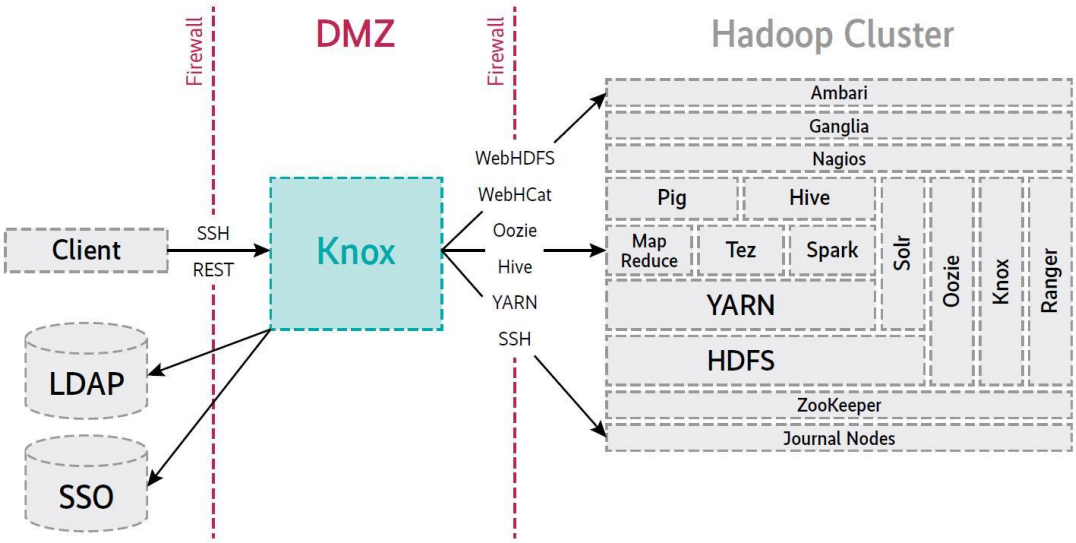

Abbildung 49: Einsatz von Apache Knox als Proxy

Neben der Sicherung des Zugriffs, also die Beantwortung der Frage „Wer darf wie auf das Hadoop-Cluster zugreifen?“ stellt sich immer die Frage, was nach erfolgreicher Authentifizierung getan werden darf. Dieser Punkt der Autorisierung wird in den meisten Komponenten innerhalb des Ökosystems eigenständig gelöst und muss in diesem Fall je Komponente und Einsatzzweck betrachtet werden. So unterstützt HDFS beispielsweise eine vollumfängliche POSIX-kompatible Berechtigungen sowie ACLs, und Hive ermöglicht die Vergabe von Berechtigungen auf Tabellen- bis hin zu Spaltenebene. Die einzelnen Ansätze der verschiedenen Komponenten ermöglichen mithin die Kontrolle des Datenzugriffs, jedoch verursacht dieses einen erhöhten administrativen Aufwand, der letztendlich wieder zu Lücken im Berechtigungskonzept führen kann. Aus diesem Grund wurde mit Apache Ranger ein relativ junges Projekt gestartet, das über einen modularen Aufbau eine zentrale Stelle für die Steuerung von Datenzugriffsberechtigungen im Hadoop-Ökosystem schafft. Hierfür stellt Knox eine einheitliche Weboberfläche für das übergreifende Administrieren von Policies

sowie einen Policy Manager zur Verfügung, der diese speichert und durchsetzt. Darüber hinaus beinhaltet Ranger zudem einen Auditserver, sodass nachträglich ermittelt werden kann, wer welche Aktion durchgeführt hat.

Der letzte Bereich bezieht sich auf die Verschlüsselung. Hierbei muss zwischen zwei Arten von Verschlüsselung unterschieden werden, zum einen die Verschlüsselung der Kommunikation und zum anderen die Verschlüsselung der Daten selbst. Erstere ist zur Verschlüsselung der Datenübertragung über SSL direkt in Hadoop eingebaut und erlaubt es, den gesamten Intra- und Internetverkehr zu verschlüsseln. Dem gegenüber steht die Verschlüsselung der Daten im HDFS selbst. Hierfür muss aktuell noch auf Software von Drittanbietern zurückgegriffen werden. Jedoch wird entsprechende Software Schritt für Schritt in den Hadoop-Standard integriert. (Heise Medien GmbH & Co. KG 2015, S. 42)

Datensicherung/Wiederherstellung

Die Notwendigkeit des Punktes Datensicherung und Wiederherstellung ist im Umfeld von Apache Hadoop nicht sofort ersichtlich. Im Grundlagenkapitel zur Architektur von Hadoop wurde ausführlich beschrieben, dass die verteilte Architektur von HDFS unter Verwendung von Replikationsmechanismen eine zusätzliche Datensicherung überflüssig macht. Dieser Punkt bezieht sich indes nur auf die technische Sicherung von Daten. Ein Schutz vor unabsichtlichem Löschen oder Ändern von Daten durch die Benutzer selbst ist hierdurch jedoch nicht gegeben.

Aus diesem Grund wurde mit der Version 2.2.0 von Hadoop eine sogenannte Snapshot-Sicherung eingeführt. Diese bietet die Möglichkeit, im HDFS Verzeichnisse auf Veränderungen zu überwachen und diese zu protokollieren. Zu diesem Zweck wird mit Aktivierung der Funktion initial eine Kopie der zu überwachenden Verzeichnisse angelegt. Modifikationen innerhalb dieser Verzeichnisse werden anschließend protokolliert und als Inkremente zum vorherigen Stand abgelegt. Auf diese Weise ist eine spätere Wiederherstellung einer früheren Version einer Datei oder eines Verzeichnisses möglich. Da Hive-Tabellen ebenfalls innerhalb des HDFS liegen, können somit theoretisch auch diese zu einem späteren Zeitpunkt wiederhergestellt und somit vor ungewollten Änderungen geschützt werden. Hierbei ist allerdings zu bedenken, dass neben den Dateien auch die zugehörigen Metadaten der entsprechenden Tabellen dieses Zeitpunktes wiederhergestellt werden müssen, da es sonst beispielsweise zu Unterschieden in den Schemata kommen könnte. (Apache Software Foundation 2013)

Datenintegrität/Datenkonsistenz

Der Punkt Datenintegrität und Datenkonsistenz bezieht sich ausschließlich auf die Komponente Apache Hive, da diese einem klassischen relationalen Datenbankmanagementsystem am nächsten kommt.

Aufgrund der Tatsache, dass Hive zum aktuellen Stand keine Fremdschlüsselbeziehungen unterstützt, besteht keine unmittelbare Verbindung der einzelnen Tabellen untereinander. Die Gewährleistung einer referentiellen Integrität eines Datenmodells wird folglich nicht softwareseitig unterstützt. Bestehen in dem zu beladenden Datenmodell mithin solche Abhängigkeiten, muss im Rahmen des ETL-Prozesses sichergestellt werden, dass diese eingehalten werden.
Die derzeit einzige Funktion von Hive, die unter dem Punkt Datenintegrität anzusiedeln wäre, betrifft die Typisierung von Daten in Tabellen. Hierdurch kann sichergestellt werden, dass nur Eingaben bzw. Inhalte eines bestimmten Typs innerhalb einer Spalte vorkommen. Apache Hive unterstützt verschiedene Datentypen für seine Attribute. Welche Datentypen dies genau betrifft, kann im Grundlagenkapitel Hive im Abschnitt Hadoop eingesehen werden.
Insgesamt wird ersichtlich, dass Hive im Vergleich zu klassischen RDBM-Systemen nur einen sehr kleinen Funktionsumfang hinsichtlich der Datenintegrität und -konsistenz anbietet. Die fehlenden Funktionen müssen in der Erstellung der ETL-Prozesse berücksichtigt und abgefangen werden, was deren Entwicklung komplexer und somit aufwändiger gestaltet.

Neben den Möglichkeiten der Datenbewirtschaftung eignen sich daher auch die Datenhaltungskomponenten von Hadoop für die Realisierung eines Data Warehouse. Hierbei steht, wie in den Hadoop-Grundlagen vorgestellt, nicht nur die relationale Datenspeicherung über Apache Hive, sondern mit Apache Kylin auch eine Komponente für die multidimensionale, für Abfragen optimierte Speicherung von Daten zur Verfügung.

Dennoch zeigt sich an dieser Stelle auch, dass Hadoop in Kombination mit Hive nicht uneingeschränkt als Data-Warehouse-System verwendet werden kann. Bereiche, die besonders hohe Anforderungen an die Integrität und Konsistenz haben, sollten genau durchdenken, ob Hive in ihrem Fall eine geeignete Lösung darstellt.

5.2.3 Datenauswertung

Neben der Datenhaltung und Datenbewirtschaftung spielt die Möglichkeit der Datenauswertung eine entscheidende Rolle für die Eignung von Hadoop als Grundlage für ein Data-Warehouse-System. Bei rückblickender Betrachtung der Data-Warehouse-Grundlagen lässt sich die Möglichkeit der Datenauswertung als das primäre Ziel eines Data Warehouse identifizieren. Sämtliche Anstrengungen für die Erstellung eines Data-Warehouse-Systems und die damit verbundenen ETL-Prozesse resultieren aus dem Ziel, Analysen auf diesen transformierten Daten zu erstellen und somit neue Informationen zu generieren.

Prinzipiell ist die Frage, ob eine entsprechende Möglichkeit zur Datenauswertung gegeben ist, bereits im Grundlagenkapitel zu Apache Hive beantwortet worden. Hier

wurde festgestellt, dass Hive mit HiveQL eine SQL-Schnittstelle anbietet, die über den Thriftserver auch Anfragen externer Systeme und Programme unterstützt. Zu diesem Zweck stehen ein ODBC- und JDBC-Treiber zu Verfügung, mittels derer Hive als Datenbank an jede beliebige Anwendung angebunden werden kann, die ODBC- bzw. JDBC-Verbindungen unterstützen. Zu diesen gehören auch die Business Intelligence Tools aller bekannten Hersteller.

Aus diesem Grund soll in dem folgenden Abschnitt der Punkt Datenauswertung weniger technisch betrachtet und stattdessen an einem kurzen praktischen Beispiel erläutert werden.

Das Beispiel basiert auf einem Star-Schema, welches Teil des frei verfügbaren Microsoft-Beispiels Data Warehouse AdventureWorks DW ist. Es umfasst in Summe sieben Tabellen, bei denen es sich um sechs Dimensionstabellen und eine zentrale Faktentabelle handelt. Die Faktentabelle beinhaltet knapp 35.000 Zeilen. Das Datenmodell bildet ein Kontenmodell ab, wie es in klassischen Planungs- und Konsolidierungssystemen zu finden ist. Bei dem Beispiel handelt es sich somit nicht um ein typisches Big-Data-Szenario, sondern würde sich in der vorliegenden Form vornehmlich in klassischen Data-Warehouse-System wiederfinden. Die nachfolgende Abbildung veranschaulicht das Datenmodell.

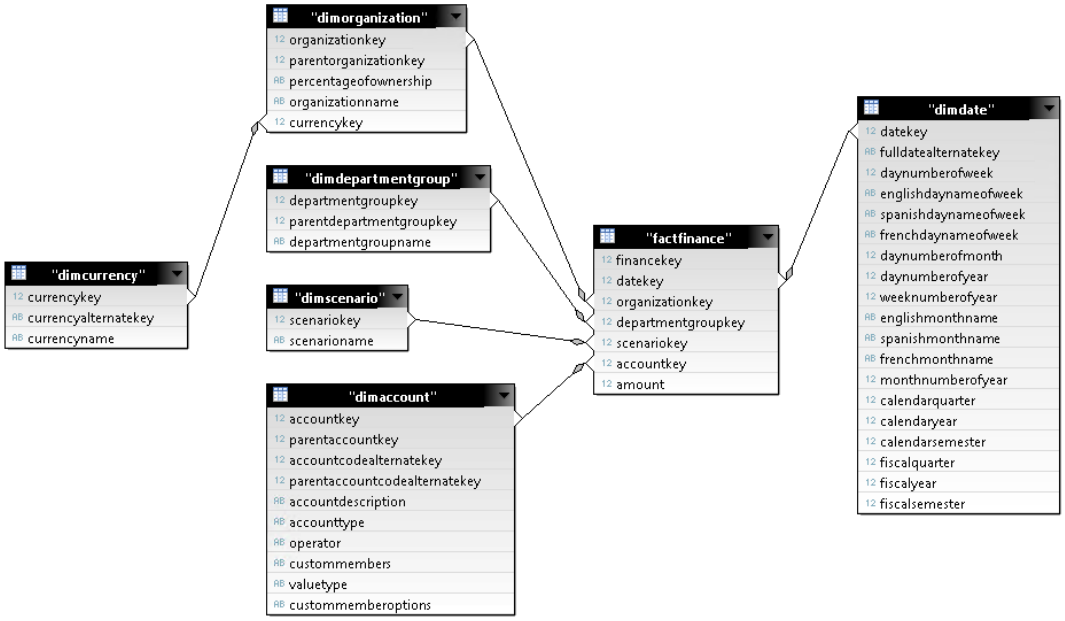

Abbildung 50: Beispiel Star-Schema: Finanzdaten aus Microsoft Adventurworks DW

Das Datenmodell wurde samt Daten in einer MySQL-Datenbank sowie mittels Hive in ein Apache-Hadoop-System implementiert. Beide Implementierungen wurden auf dem gleichen physikalischen Server durchgeführt, sodass für den Test für beide Systeme dieselben Voraussetzungen gelten. Bei dem System handelt es sich um einen Single Node Cluster, welches über 8 Gigabyte RAM und vier virtuelle Prozessoren

mit einer Taktrate zu 2.27 GHz verfügt. Aufgrund der schwachen Hardwarespezifikationen ist es somit ganz klar als Testsystem einzuordnen, das nicht für einen produktiven Einsatz als Data-Warehouse-System geeignet wäre.

Es ist während der Implementierungsphase leider nicht gelungen, einen OLAP Cube in Apache Kylin auf Basis des gezeigten Datenmodells anzulegen und ebenfalls mit in den Test einzubeziehen. Aus diesem Grund wurde Apache Spark als dritte Auswertungskomponente mit in den Test aufgenommen. Bei Spark handelt es sich um eine In-Memory-basierte Ausführungsengine, die in Konkurrenz zu MapReduce und Tez steht. Spark ist eines der derzeit am stärksten weiterentwickelten Projekte im Hadoop-Ökosystem und wird teilweise bereits als legitimer Hadoop- bzw. MapReduce-Nachfolger angesehen. Spark bietet im vorliegenden Fall den Vorteil, dass es als Engine für Apache Hive und somit als bereits implementiertes Datenmodell mehrfach verwendet werden kann.

Für die praktische Durchführung des Tests wird die SAP-Business-Intelligence-Plattform in Version 4.2 verwendet, bei der es sich um die Hauptanalyseanwendung der Firma SAP handelt. Die Business-Intelligence-Plattform stellt eine Vielzahl von Werkzeugen für die Erstellung von Standardberichten und für die interaktive Analyse von Daten zur Verfügung. Hierbei unterstützt sie neben den SAP-eigenen Data-Warehouse-Systemen bzw.- Datenbanken SAP BW, SAP Hana und SAP Sybase IQ jegliche andere Datenbanksysteme, die über eine ODBC- oder JDBC-Schnittstelle angesprochen werden können. Mit der Version 4.2 werden durch die SAP bereits SAP-eigene Standards mit ODBC-Treiber für Apache Spark und Apache Hive mit ausgeliefert. Diese sind mit der im Test eingesetzten Hadoop-Version 2.3 auf der Hortonworks-Data-Platform kompatibel, sodass lediglich ein ODBC-Treiber für den Zugriff auf die MySQL-Datenbank installiert werden muss. (SAP SE 2015b, S. 5)

Im Test wird auf die zuvor implementierten Datenmodelle mithilfe des SAP-Information-Design-Tools zugegriffen. Hierbei handelt es sich um ein Werkzeug, das es erlaubt, eine semantische Schicht auf einem logischen Datenbankmodell zu erstellen. Hierzu wird zunächst für jedes System eine plattformabhängige Datenbankverbindung eingerichtet. Auf dieser Datenbankverbindung wird anschließend das logische Datenbankmodell nachgebildet und mit Verbindungen (Joins) sowie entsprechenden Kordialitäten versehen. (SAP SE 2015a, S. 15)

Die folgende Abbildung zeigt einen Screenshot aus der Anwendung. Auf der linken Seite ist die Verbindung zum Hive-Server und auf der rechten Seite das auf der Verbindung erstellte Datenmodell mit den logischen 1:N-Verknüpfungen der Dimensionstabellen zur zentralen Faktentabelle zu sehen.

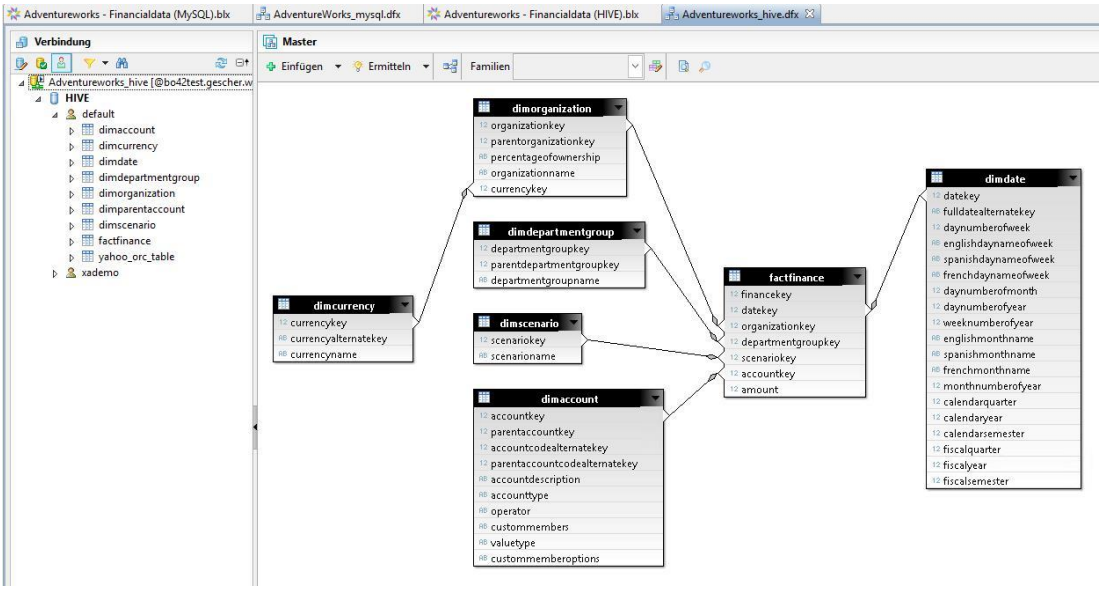

Abbildung 51: Bildschirmausschnitt aus dem SAP-Information-Design-Tool

Auf Basis dieses logischen Datenmodells kann anschließend eine semantische Schicht erstellt werden. Dies bedeutet, dass das physische Datenmodell vom Anwender abstrahiert wird und dieser stattdessen über einen definierten Vorrat von Dimensions- und Kennzahlobjekten selbstständig Abfragen erstellen kann. Hierzu benötigt er weder Kenntnisse über das zugrunde liegende Datenmodell ohne das darunterliegende Datenmodell noch über die Abfragesprache SQL. (SAP SE 2015a, S. 15)

Der folgende Bildschirmausschnitt zeigt die semantische Schicht, die auf dem zuvor erstellten Datenmodell angelegt wurde. In dieser können Objekte von ihren eigentlichen technischen Namen entkoppelt und fachlich korrekt benannt werden, sodass ein Anwender aus einem Fachbereich mit diesen intuitiv in der ihm bekannten Nomenklatur arbeiten kann. Dazu kommt die Möglichkeit, für jedes Element ein spezifisches SQL-Kommando hinterlegen zu können, über das die Daten z. B. eingeschränkt oder umformatiert werden können.

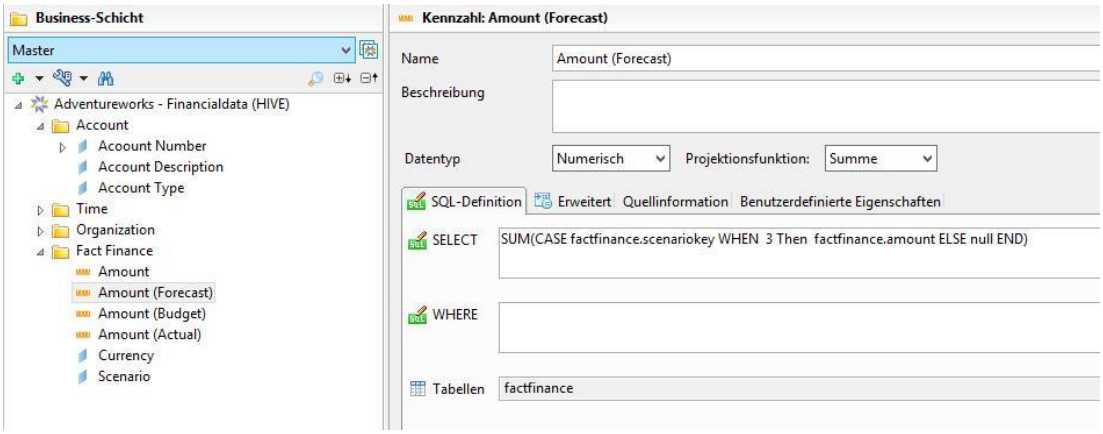

Abbildung 52: Editor für die Erstellung einer semantischen Schicht im SAP-Information-Design-Tool

Auf Basis dieser semantischen Schicht kann ein Anwender anschließend selbstständig Abfragen bzw. Berichte erstellen. Hierzu wird ihm in der ausgewählten Applikation ein Abfrageassistent zur Verfügung gestellt, über den mittels Drag & Drop Abfragen erstellt werden können. Diese werden anschließend von der Business-Intelligence-Plattform automatisch in SQL-Abfragen übersetzt und auf der entsprechenden Datenbank ausgeführt. Bei der Erstellung der Abfragen wird dabei der jeweils in den ODBC-Treibern hinterlegte SQL-Dialekt verwendet. (SAP SE 2015a, S. 15)

Die nächste Abbildung zeigt exemplarisch einen solchen Abfrageassistenten, über den eine Abfrage auf der zuvor angelegten semantischen Schicht erstellt wurde. Zusätzlich ist dazu die hieraus generierte SQL-Abfrage eingeblendet, die dem Endanwender im Normalfall nicht ersichtlich ist.

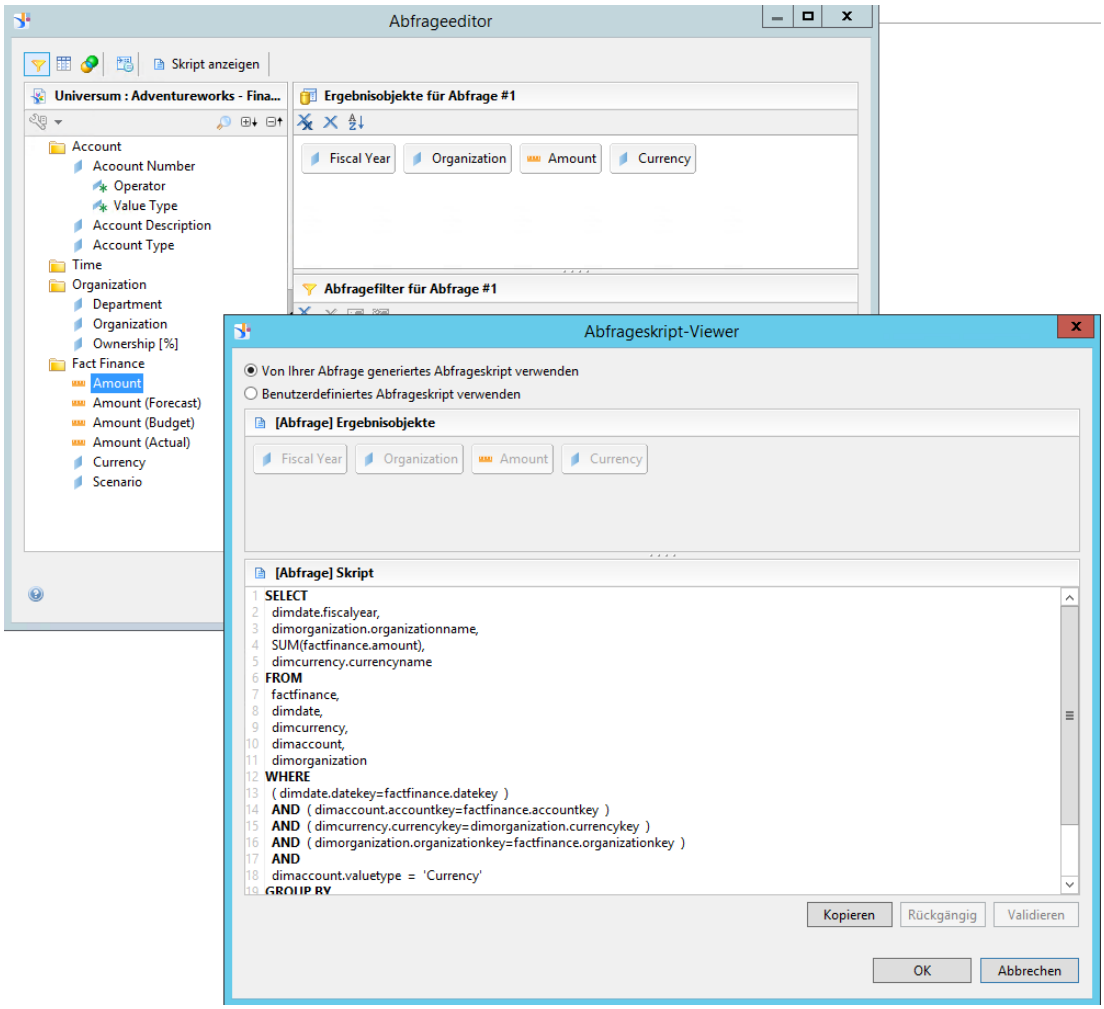

Abbildung 53: Abfrageassistent inkl. generiertem Abfrageskript

Im Sprachjargon der SAP wird die Bündelung der drei Komponenten Verbindung, Datengrundlage und semantische Schicht als ein sogenanntes Universum bezeichnet. Ein solches Universum kann nach erfolgreicher Erstellung auf der Business-Intelli-

gence-Plattform publiziert und anschließend zwecks Auswertung in den verschiedenen Reporting-Werkzeugen verwendet werden. Ein solches Reporting-Werkzeug ist das SAP Web Intelligence. Hierbei handelt es sich um ein einfaches, webbasiertes Werkzeug zur Erstellung von Standardberichten, das so konzipiert ist, dass es nach kurzer Einweisung auch von Endanwendern direkt verwendet werden kann. (SAP SE 2015a, S. 15)

Im vorliegenden praktischen Beispiel wurde auf Basis der drei Plattformen MySQL, Apache Hive und Apache Spark jeweils ein Universum erstellt. Auf allen drei Universen wurden jeweils zehnmal fünf unterschiedliche Abfragen ausgeführt. Die Abfragen erstrecken sich dabei von einem einfachen Select Statement bin hin zu einer komplexen Analyseabfrage. Im Detail wurden die fünf folgenden Abfragetypen definiert. Die einzelnen Abfrageskripte sind im Anhang der Arbeit einsehbar.

– Einfacher Select: simples Select auf eine einzelne Tabelle.
– Einfaches Group By: einfaches Select Statement auf eine einzelne Tabelle mit Verwendung einer Aggregationsfunktion.
– Einfacher Map Join: einfacher Inner Join zwischen zwei Tabellen ohne weitere Einschränkungen oder Aggregationsfunktionen.
– Komplexer Map Join: komplexer Join über fünf Tabellen ohne weitere Einschränkungen oder Aggregationsfunktionen.
– Komplexe Analyseabfrage: komplexer Join über fünf Tabellen mit zwei Aggregationsfunktionen unter Verwendung von Case-When-Anweisungen, Einschränkungen sowie Angaben zur Sortierung des Ergebnisses.

Zur Leistungserfassung wurden die einzelnen Ausführungszeiten der zehn Abfragen je Plattform erfasst und das arithmetische Mittel über diese gebildet. Durch das Vorgehen, mit jedem Schritt komplexere Anfragen an die einzelnen Plattformen zu stellen, sollte in der letzten Aufbaustufe das Ausführen einer Abfrage simuliert werden, wie sie auch in einem produktiven Data Warehouse eingesetzt werden könnte. Der Aufbau der Abfragen hatte dabei zum Ziel, Operationen auf den einzelnen Plattformen zu identifizieren, die die Ausführungsdauer besonders stark beeinträchtigen.

Die folgende Abbildung zeigt das Ergebnis der Auswertung in tabellarischer und grafischer Form.

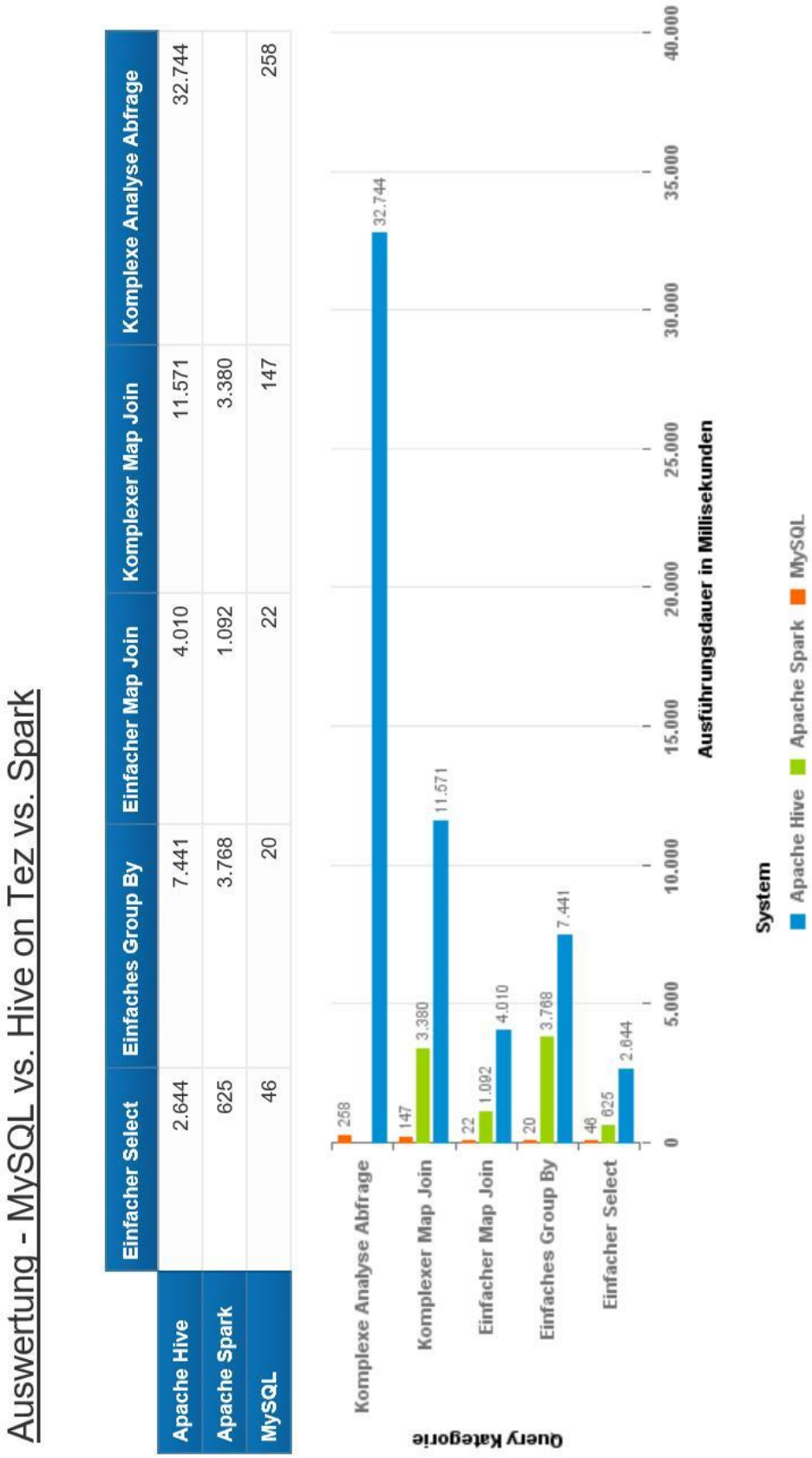

Abbildung 54: Auswertung – Laufzeitanalyse verschiedener Abfragen auf Apache Hive (on Tez), Apache Spark und MySQL (klassische RDBMS). Alle Angaben in Millisekunden.

Bei der Betrachtung der Auswertungsergebnisse ergeben sich drei zentrale Erkenntnisse für den Punkt Datenauswertung über Apache Hadoop. Erstens: Sowohl Apache Hive als auch Apache Spark bieten prinzipiell die Möglichkeit, mit klassischen BI-Tools verwendet werden zu können. Dies hat auch ein nachgelagerter Test gezeigt, bei dem auf Basis der erstellten Universen erfolgreich ein Web-Intelligence-Bericht erstellt werden konnte. Die Fragestellung, ob sich Apache Hadoop also auch für den Punkt Datenauswertung anbietet, kann folglich mit Ja beantwortet werden.

An dieser Stelle kommt jedoch ein großes Aber hinzu. Wie die Auswertung zeigt, haben sowohl Apache Hive als auch Apache Spark im Vergleich zu einem klassischen Datenbankmanagementsystem extrem hohe Ausführungszeiten von selbst einfachsten Abfragen. Dies führt dazu, dass das klassische Datenbankmanagementsystem im vorliegenden Fall um bis zu 372-mal schneller als Apache Hive und bis zu 188-mal schneller als Apache Spark ist. Diese langen Ausführungszeiten bringen ein Problem mit sich, das bei der potenziellen Einführung eines Hadoop-Systems unbedingt berücksichtigt werden muss. MapReduce oder auch Tez sind Batch-basierte Verfahren, die zwar in der Lage sind, gut zu skalieren und hierdurch sehr große Datenmengen verarbeiten zu können, hierdurch jedoch völlig ungeeignet für Ad-hoc-Analysen sind. In Zeiten von In-Memory-Datenbanken wie der SAP Hana erwarten Analysten Antwortzeiten im Bereich von einigen Millisekunden bis maximal einigen Sekunden, allerdings keinesfalls im Bereich von Minuten oder Stunden. An dieser Stelle könnten Ansätze wie das in den Grundlagen vorgestellte Apache Kylin interessant werden, die Abfrageergebnisse in OLAP-Cubes vorberechnen und diese Ergebnisse in der Key-Value-Datenbank HBase ablegen. Erste Ergebnisse, die sich im Internet zu diesem Thema finden lassen, veranschaulichen, dass durch dieses Verfahren auch Antwortzeiten im Bereich von wenigen Sekunden realisiert werden können. Bei der Verwendung von OLAP-Cubes muss jedoch stets bedacht werden, dass diese vorberechnet werden müssen und somit nicht für Echtzeitanalysen geeignet sind.

Abschließend soll noch auf eine Besonderheit in der Auswertung hingewiesen werden. Die Darstellung zeigt, dass Apache Spark in allen Fällen mit Ausnahme der komplexen Analyseabfrage deutlich schneller als Apache Hive war. Diese Abweichung kommt dadurch zustande, dass für die komplexe Analyseabfrage kein abschließendes Ergebnis mit Apache Spark erzielt werden konnte. Apache Spark war im vorliegenden Test ab einem gewissen Komplexitätsgrad nicht mehr in der Lage, Abfragen auszuführen. Dieses Ergebnis deckt sich mit Erkenntnissen, die bei einer anschließenden Analyse im Internet gesammelt werden konnten. Apache Spark ist im aktuellen Entwicklungsstand (Version 1.4) nicht dazu fähig, komplexe Abfragen auszuführen. Hierdurch ist zumindest Apache Spark als Analysewerkzeug zum aktuellen Zeit-

punkt im Data-Warehouse-Umfeld ungeeignet, da die Beantwortung von umfangreichen Abfragen zur Gewinnung neuer Erkenntnisse Kernkompetenz eines Data Warehouse ist.

5.2.4 Zusammenfassung

Bei der rückblickenden Betrachtung der drei vorhergehenden Abschnitte kann Apache Hadoop prinzipiell aus technischer Sicht die Fähigkeit eingeräumt werden, als Data-Warehouse-System zu fungieren. Eine grundlegende Empfehlung, Hadoop uneingeschränkt für jede Anforderung einsetzen zu können, kann indes nicht erteilt werden. Wie bei jedem anderen Datenbankmanagementsystem auch ist eine solche Entscheidung immer in Abhängigkeit der konkreten fachlichen und technischen Anforderungen zu treffen. In einem solchen Entscheidungsprozess muss genau evaluiert werden, welche Komponenten aus dem Hadoop-Ökosystem eingesetzt werden sollen und ob diese die technischen Anforderungen im Detail erfüllen. Hierbei kann es z. B. im Detail auch notwendig sein, alle benötigten SQL-Befehle und Konstrukte zu identifizieren und diese gegen die zur Verfügung stehende Syntax abzugleichen. Darüber hinaus gilt es, genau zu prüfen, ob ein Hadoop-Cluster sinnvoll in eine bestehende IT-Infrastruktur integriert werden kann und ob deren Schnittstellen und Applikationen für die Kommunikation mit Hadoop geeignet sind.

Für solche Evaluierungszwecke stellen die Anbieter der großen Hadoop-Distributionen wie Cloudera, Hortonworks oder MapR Sandboxes auf ihren Webseiten zum Download zur Verfügung. Bei diesen Sandboxen handelt es sich um vollständig vorkonfigurierte Hadoop-Installationen inklusive der meisten gängigen Komponenten wie Hive, Pig oder Sqoop. Über eine solche Sandbox können schnell und unkompliziert erste Erfahrungen mit dem praktischen Einsatz von Hadoop gesammelt werden, ohne sich zuvor langwierig in die Installations- und Konfigurationsprozesse einarbeiten zu müssen. Darüber hinaus können hierdurch auch die einzelnen Distributionen einfach miteinander verglichen werden, um so die beste Lösung für das jeweilige Projekt zu finden.

5.3 Architektonische Ansätze

Nachdem die technische Überprüfung, ob Apache Hadoop als technologische Basis für ein Data Warehouse fungieren kann, im vorherigen Kapitel positiv beantwortet wurde, stellt sich in diesem Zusammenhang jedoch immer noch eine zentrale Frage. Diese Frage bezieht sich nicht auf die Frage, wie sich technisch gesehen ein Data Warehouse auf Apache Hadoop realisieren lässt, sondern auf das Wozu und Warum. Zur Beantwortung dieser Frage werden im folgenden Kapitel verschiedene Szenarien vorgestellt, die beschreiben, wie sich Apache Hadoop sinnvoll in eine bestehende Data-Warehouse-Architektur integrieren lässt.

Hierzu sei vorab angemerkt, dass die Szenarien nicht den Anspruch verfolgen, vollständig zu sein und alle denkbaren Konstellationen abzudecken, sondern lediglich als Denkanstoß dienen sollen, die Problemstellung auf andere Ansätze zu projizieren. Die Szenarien sind dabei so aufgebaut, dass sie möglichst einfach beginnen und von Stufe zu Stufe an Komplexität dazugewinnen.

Abbildung 55: Klassischer Data-Warehouse-Aufbau (Tamara Dull 2015, S. 5)

Das obige Schaubild zeigt eine stark vereinfachte Data-Warehouse-Architektur, die sich aus den drei Komponenten strukturierte Datenquellen, Data Warehouse und klassischen BI- bzw. Analysewerkzeugen zusammensetzt. Diese Form der Architektur wurde in den Grundlagen ausgiebig behandelt, sodass die drei Komponenten klar einzusortieren sein sollten. Bei den strukturierten Datenquellen handelt es sich um typische transaktionale Systeme, deren Daten über einen ETL-Prozess in ein entsprechendes Data Warehouse geschrieben werden. Auf dieses wird anschließend mittels Auswertungswerkzeugen zugegriffen. Wie im vorhergehenden Verlauf dieser Arbeit bereits beschrieben, sind viele dieser Werkzeuge dazu in der Lage, neben klassischen Data-Warehouse-Systemen auch auf Apache Hadoop zugreifen zu können.

Dem klassischen Data-Warehouse-Ansatz steht eine isolierte Hadoop-Architektur gegenüber, wie sie auf der folgenden Abbildung zu sehen ist. Diese setzt sich aus den drei Komponenten unstrukturierte Datenquellen, Hadoop und sogenannten Big-Data-Apps zusammen. Im Bild wird nur HDFS namentlich erwähnt, jedoch ließe sich hier genauso gut jede andere denkbare Datenhaltungskomponente aus dem Hadoo-Ökosystem einsetzen. Bei den unstrukturierten Datenquellen handelt es sich um all die Daten, die gar nicht oder nur schlecht in klassischen Datenbanksystemen abgelegt werden können. Gründe hierfür können eine fehlende Struktur oder aber das Volumen der Daten als auch die Geschwindigkeit sein, mit der diese angeliefert werden. An dieser Stelle schließt sich somit wieder der Kreislauf zum Stichwort Big Data und den damit verbundenen Herausforderungen.

Big-Data-Apps bezeichnen Applikationen und Werkzeuge, die speziell für den Zugriff und die Analyse von Big Data (in Hadoop oder anderen nicht relationalen Speichersystemen) konzipiert wurden. Klassischerweise sind Big-Data-Apps nicht traditionellen Analysewerkzeugen gleichzusetzen, da diese einen anderen Fokus haben. Klassische Analysewerkzeuge haben beispielsweise häufig einen Fokus darauf, Daten möglichst einfach zugänglich zu machen und ansprechend zu visualisieren. Big-Data-Apps hingegen haben häufig überhaupt keine eigene Visualisierungskomponente und werden über Skriptsprachen bedient. Beispiele hierfür wären die Statistiksprache R oder die Data-Mining- und Machine-Learning-Programme Apache Mahout und MLib. Big-Data-Apps beschäftigen sich somit mit der tiefgehenden Analyse von Daten und sind durch ihre Komplexität nicht für den Gebrauch durch Endanwender vorgesehen. In diesem Zusammenhang ist in den vergangenen Jahren die Berufsbezeichnung des Data Scientist etabliert worden, der sich hauptberuflich mit der systematischen Analyse und der Interpretation von Datenzusammenhängen beschäftigt. (Tamara Dull 2015, S. 6)

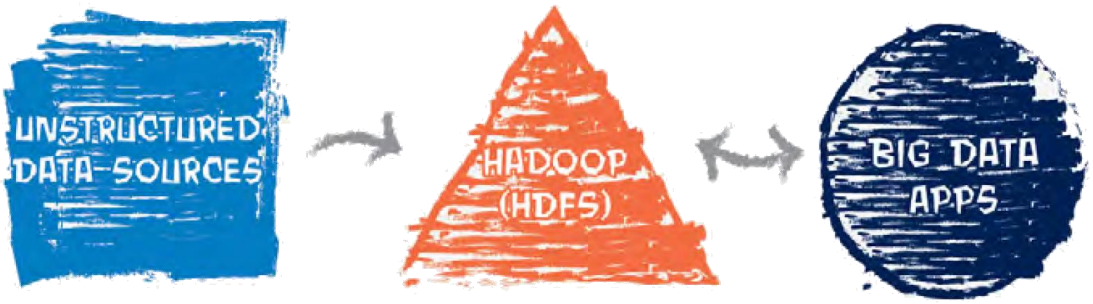

Abbildung 56: Isolierte Hadoop-Anwendung (Tamara Dull 2015)

Bei der Betrachtung der beiden isolierten Architekturen stellt sich nun die Frage, wie diese sinnvoll miteinander verbunden werden können, um so neue Auswertungspotenziale und Synergien über die Verbindung beider Welten zu schaffen.

Zunächst wirft sich die Frage auf, weshalb ein bestehendes Data Warehouse überhaupt um ein Hadoop-Cluster erweitert oder gar von diesem abgelöst werden sollte. Hierfür kann es auf den ersten Blick prinzipiell vier augenscheinliche Gründe geben, die sich aus den Charakteristika des Begriffs Big Data ableiten. Alternative eins: Die bestehende Datenmenge nimmt so stark und so schnell zu, dass das bestehende System diese nicht mehr verarbeiten kann. Alternative zwei: Die Geschwindigkeit, mit der neue Daten zu verarbeiten und bestehende Daten zu analysieren sind, kann vom bestehenden System nicht gewährleistet werden. Option Nummer drei: Neben den klassischen strukturierten Daten kommen neue Datenquellen hinzu, die gespeichert und analysiert werden sollen, mit denen das existierende Data-WarehouseSystem jedoch nicht umgehen kann. Zuletzt besteht die Möglichkeit, dass neue Analysemöglichkeiten benötigt werden, die im klassischen Ansatz nicht zur Verfügung stehen oder

nur eingeschränkt mit diesem kompatibel sind. Natürlich ist neben diesen Einzelfallbetrachtungen auch ein hybrides Szenario aus mehreren oder allen der genannten Optionen möglich bzw. in der Praxis weitaus wahrscheinlicher.

Ausgangslage für das erste Szenario ist eine Situation, in der ein Data-Warehouse-Verantwortlicher vor dem Problem steht, dass sein bestehendes System mittelfristig erweitert werden muss, da dieses die wachsenden Datenmengen nicht mehr dauerhaft speichern und verarbeiten kann. In dieser Situation stehen ihm zwei Möglichkeiten zur Auswahl zur Verfügung. Möglichkeit eins: Er erweitert das bestehende System um weitere Komponenten. Möglichkeit zwei: Er denkt über andere technologische Ansätze nach, die sein Problem langfristig lösen könnten.

Naheliegend wäre die Auswahl der ersten Möglichkeit. Durch den weiteren Einsatz eines bereits bekannten Systems erhöht sich die Komplexität innerhalb der IT zunächst nicht. Dieses schützt Investitionen in die bereits erfolgte Ausbildung der Mitarbeiter und ermöglicht so eine genauere Kalkulation der zu erwartenden Kosten, da auf Erfahrungen aus früheren Projekten zurückgegriffen werden kann. Es besteht jedoch die Möglichkeit, dass dieses Vorgehen sich bei detaillierter Betrachtung doch nicht als praktikabel erweist. Beispielsweise kann dies daran liegen, dass die technischen Grenzen des bestehenden Systems erreicht sind. Die Speicher- oder Rechenkapazität des Systems kann nicht einfach erweitert werden oder aber benötigt eine Umstrukturierung des Systems inklusive bereits erstellter ETL-Strecken, sodass eine Anpassung derselben notwendig werden würde. Ein weiterer Punkt wären die Kosten für teure Spezialhardware oder weitere Lizenzen, die mit einer Erweiterung des bestehenden Systems einhergehen könnten.

Beide Fälle können dazu führen, dass es sinnvoll erscheint, neben der ersten Option die zweite Option zu evaluieren, welche die Prüfung anderer technologischer Ansätze umfasst. An dieser Stelle kommt Apache Hadoop ins Spiel, welches Mittelpunkt einer solchen Evaluierung sein könnte.

Eine Möglichkeit für den Einsatz von Hadoop in diesem Szenario bestünde darin, Hadoop als Staging Area und als Operational Data Storage für das Data Warehouse einzusetzen. Dies bedeutet, dass diese beiden Ebenen aus dem bestehenden Data Warehouse ausgelagert werden könnten. Hierdurch müsste dieses lediglich nur noch die zentrale Datenhaltungskomponente sowie die Reporting-optimierten spezialisierten Datenbereitstellungsebenen vorhalten. Ein Großteil der Daten wird in den günstigeren Hadoop-Speicher ausgelagert, sodass das zentrale Data Warehouse lediglich die transformierten aggregierten Daten vorhalten muss. Dieses Szenario bietet den Vorteil, dass es relativ schnell zu realisieren ist, da die bestehenden ETL-Prozesse zu großen Teilen bestehen bleiben können. Es ist lediglich eine Anpassung des Speicherortes der Staging Area als ODS notwendig. Alternativ könnten ETL-Prozesse je-

doch auch bereits im Hadoop-Cluster realisiert und die Daten dort transformiert abgelegt werden. Dies würde zwar initial einen hohen Aufwand zur Anpassung der ETL-Strecken bedingen, böte jedoch den Vorteil, dass auch bei der Transformation von der horizontalen Skalierbarkeit von Hadoop profitiert werden könnte. Ein solches System ist nicht nur gegen potenzielle Speicherengpässe gesichert, sondern auch gegen fehlende Rechenressourcen zur Verarbeitung der ETL-Prozesse. Das beschriebene Szenario ist in der folgenden Abbildung beispielhaft visualisiert. (Tamara Dull 2015, S. 9)

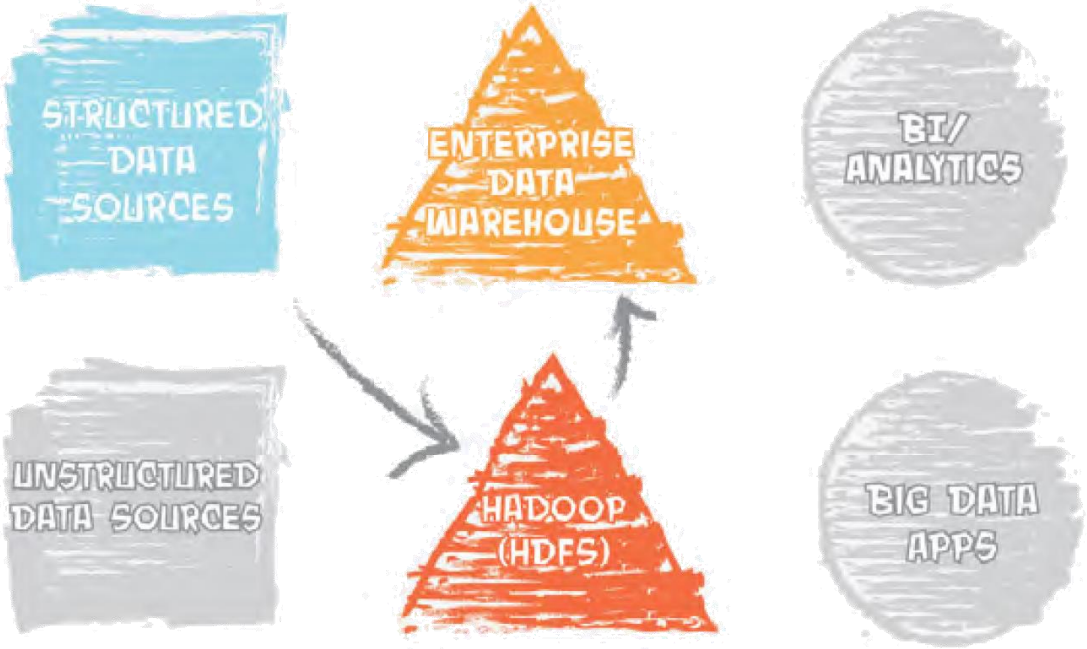

Abbildung 57: Szenario 1 – Hadoop als Staging Area für transaktionale Daten (Tamara Dull 2015, S. 8)

Da sich in dem abgebildeten Szenario die Struktur der Schicht für die spezialisierte Datenbereitstellung funktional nicht zu dem Stand vor der Hadoop-Integration unterscheidet, braucht der Zugriff auf die Daten nicht näher betrachtet werden. Für diesen können wie zuvor die bereits existierenden klassischen BI-Tools eingesetzt werden.

Das nächste Szenario weist auf den ersten Blick keine großen Unterschiede zum ersten Szenario auf.

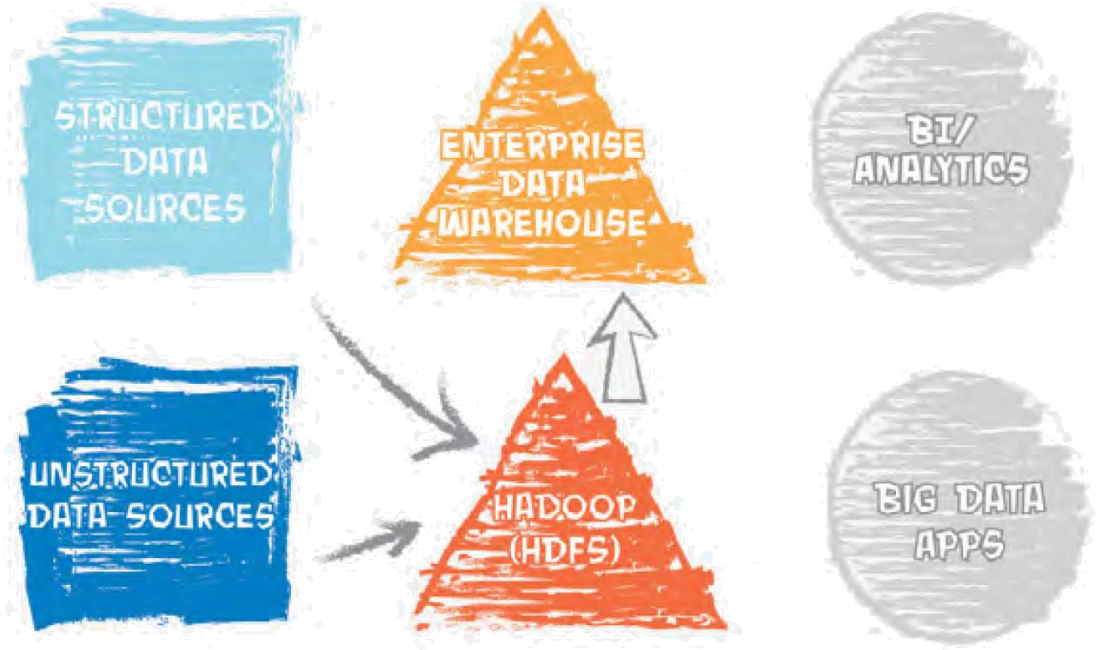

Abbildung 58: Szenario 2 – Hadoop als Staging Area für transaktionale Daten und zentrale Datenbereitstellung für unstrukturierte Daten (Tamara Dull 2015, S. 11)

Zunächst ist lediglich ein Verbinder hinzugekommen, der nun neben den strukturierten Daten auch die unstrukturierten Daten als Quelle für das Data-Warehouse-System aufnimmt. Diese unscheinbar wirkende Veränderung kann auf ein Data Warehouse jedoch komplexe Auswirkungen haben. Der erste offensichtliche Faktor, der die Komplexität erhöht, ist die Vielzahl verschiedener Formate und Dateitypen, die nun als potenzielle unstrukturierte Datenquellen zur Verfügung stehen. (Tamara Dull 2015, S. 11)

Der zweite, vermutlich weniger offensichtliche Faktor ist das potenzielle Datenvolumen. Waren die bislang vorhandenen strukturierten Datenquellen in ihrem Volumen noch überschaubar und damit effektiv zu kontrollieren, kommt mit den unstrukturierten Daten ein nur schwer zu kalkulierender Faktor hinzu. Einer IDC-Studie aus dem Jahr 2011 zufolge wachsen unstrukturierte Daten wesentlich schneller als strukturierte und sollen so bis zum Ende des aktuellen Jahrzehnts bereits 90 % aller verfügbaren Daten ausmachen. (Intel IT Center 2012, S. 1)

Das zweite Szenario behandelt damit also nicht nur den Punkt Datenvielfalt, sondern den gleichzeitig mit diesem einhergehenden Punkt Datenvolumen und wirft somit eine sehr entscheidende Frage eines jeden Big-Data-/Data-Warehouse-Projekts auf. Diese Frage lautet, ob die Analyse wirklich aller Daten einen Mehrwert bietet und folglich auch alle Daten erfasst und gespeichert werden müssen, oder ob es auch bei unstrukturierten Daten wichtigere und weniger wichtige Daten gibt. Sollte die Antwort Ja lauten und mit Hadoop ein Werkzeug zur Bewältigung aller damit verbundenen Probleme vorhanden sein, muss dennoch ein zentraler Aspekt bedacht werden. Ha-

doop und das damit verbundene Ökosystem können vielleicht all diese Probleme lösen, jedoch tun sie es zum aktuellen Zeitpunkt keinesfalls allein. Hardware zur Speicherung der Datenmengen muss zur Verfügung gestellt, ETL-Prozesse entwickelt und Auswertungen zur Verfügung gestellt werden. Alle diese Punkte kosten Zeit und Geld und widersprechen daher in Teilen dem zunächst vielleicht romantischen Gedanken von Open Source, Schemafreiheit und unendlicher Skalierbarkeit.

Die Integration von Hadoop kann also dabei helfen, bestehende Probleme in einem Data-Warehouse-System zu lösen und gleichzeitig neue Potenziale und Möglichkeiten bereitstellen. Gleichwohl sollte dieser Schritt nicht von jetzt auf gleich gegangen, sondern wohl überlegt und gut geplant werden. Auch wenn alle Daten im HDFS gespeichert werden können, so muss es noch lange nicht sinnvoll sein, diese auch alle mittels aufwendig zu erstellender ETL-Prozesse innerhalb von Hadoop zu transformieren und ins Data Warehouse zu übertragen.

Das zweite Szenario weist zum ersten Szenario einen weiteren Unterschied auf, welcher der Vollständigkeit halber erwähnt werden soll. Der Pfeil, der vom Hadoop-Cluster zum klassischen Data-Warehouse-System zeigt, weicht von der Form des Pfeils in der ersten Darstellung ab. Dahinter verbirgt sich der Umstand, dass das Hadoop-System nun nicht nur als Staging Area für das Data Warehouse fungiert, sondern aktiv Transformationen ausführt. Diese Transformationen überführen die unstrukturierten Daten in eine zur Analyse im klassischen Data Warehouse geeignete Form. Wie in den Grundlagen zu Hadoop, Hive und Pig ausführlich beschrieben, sind diese hierfür besonders prädestiniert.

Das nächste Szenario wirkt auf den ersten Blick wie ein radikaler Bruch mit den bestehenden Strukturen und verleitet zu der Annahme, dass Hadoop als die zentrale Komponente in einem Data Warehouse fungieren soll. Dies ist jedoch nicht die Intention, die mit dem dritten Ansatz verfolgt werden soll. Stattdessen handelt es sich bei diesem um wohl einen der populärsten Ansätze, in dem Hadoop als zentrales Datenarchiv eingesetzt werden soll. In vielen Data-Warehouse-Umgebungen ist es üblich, Daten nach einer gewissen Zeit aus dem DWH zu entfernen, um somit Speicherplatz wieder freizugeben und die Kosten für den Betrieb des Systems auf einem stabilen Niveau zu halten. Dieses regelmäßige Löschen kann beispielsweise nach drei, fünf oder auch erst nach zehn Jahren erfolgen. In dem aktuellen Szenario werden die Daten jedoch regelmäßig in das Hadoop-Cluster repliziert, wo sie auf der Commodity-Hardware langfristig archiviert werden können. Demzufolge wird neben der Staging Area und dem Operational Data Storage eine Kopie bzw. Archiv des zentralen Data Warehouse im Hadoop-Cluster aufgebaut, welches die Daten auch zur langfristigen Analyse zur Verfügung stellt.

Bei der Betrachtung dieses Szenarios ist es wichtig, dieses in Kombination mit den Szenarios eins und zwei zu analysieren. Es bietet mithin die Möglichkeit, die zentralen

Data-Warehouse-Daten langfristig über die gewohnten BI-Tools analysieren zu können und aus diesen unter Verwendung von Big-Data-Analysen neue Erkenntnisse zu gewinnen. (Tamara Dull 2015, S. 13)

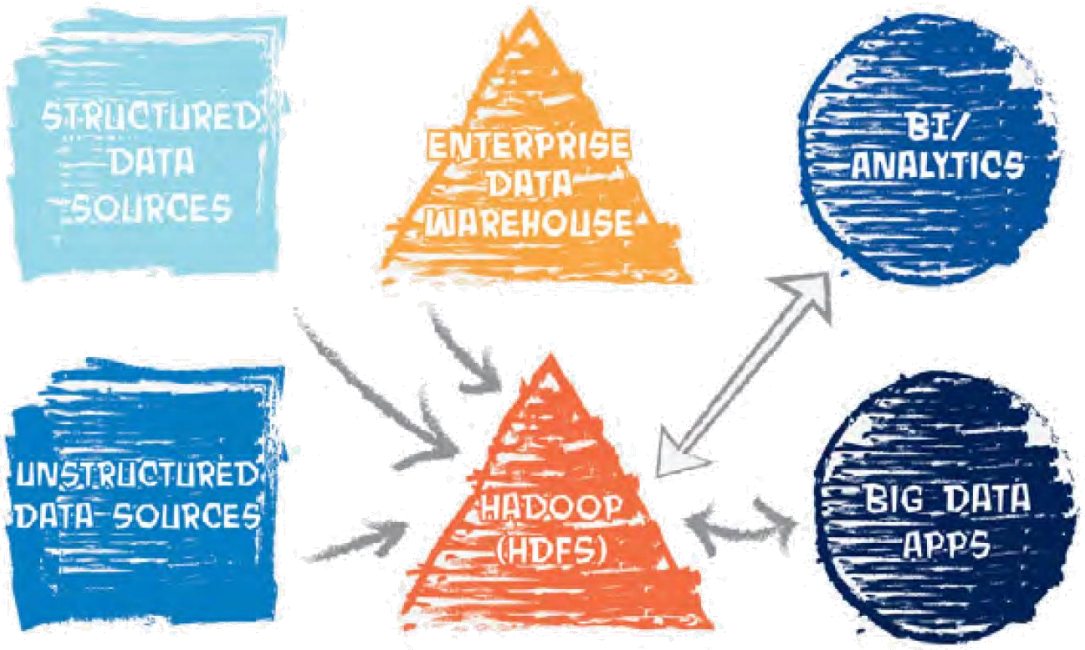

Abbildung 59: Szenario 3 – Hadoop als Archiv (Tamara Dull 2015, S. 13)

Dieses Szenario wird häufig auch als sogenanntes Multi-Temperature-Datamanagement bezeichnet. Hierbei werden die verschiedenen Datenkategorien definiert, zum einen Daten, die häufig gelesen und geschrieben werden. Dies sind die heißen Daten bzw. Hot-Data. An zweiter Stelle stehen die warmen Daten. Dies sind Daten, auf die nicht häufig zugegriffen wird. An letzter Stelle stehen die kalten Daten. Dieses sind Daten, auf die nur sporadisch zugegriffen wird. Ziel des Multi-Temperature-Datamanagements ist es, Daten eines Data Warehouse automatisch der entsprechenden Zone zuzuweisen und diese dort zu speichern. Über dieses Verfahren kann der kleine Teil der Daten, der häufig verwendet wird, in den Speichersystemen vorgehalten werden, die sehr schnelle Antwortzeiten auf Abfragen erzielen, jedoch die höchsten Speicherkosten aufweisen. Der selten verwendete Teil der Daten wird in die Systeme mit den geringsten Speicherkosten verschoben, die allerdings häufig hohe Antwortzeiten und damit eine schlechte Performance aufweisen. (SAP)

Bei der Betrachtung des Punktes Datenauswertung im Abschnitt der technischen Überprüfung hat sich gezeigt, dass Apache Hive wesentlich schlechtere Antwortzeiten als beispielsweise ein klassisches RDBMS oder moderne Ansätze wie Apache Spark (mit Ausnahme von komplexen Anfragen) hat. Die Hot-Data-Schicht wäre somit beispielsweise als eine In-Memory-Datenbank wie die SAP Hana zu sehen. Auf dieses folgt ein Apache-Spark-Cluster, das ebenfalls zu großen Teilen im Hauptspeicher ausgeführt wird. Letztendlich werden alle Daten in einem Hadoop-Cluster archiviert,

in dem die Daten mittels Apache Hive verwaltet und strukturiert zur Analyse bereitgestellt werden.

Die nachfolgende Abbildung zeigt eine solche Multi-Temperatur-Datenverwaltung schematisch im Kontext der In-Memory-Datenbank SAP Hana.

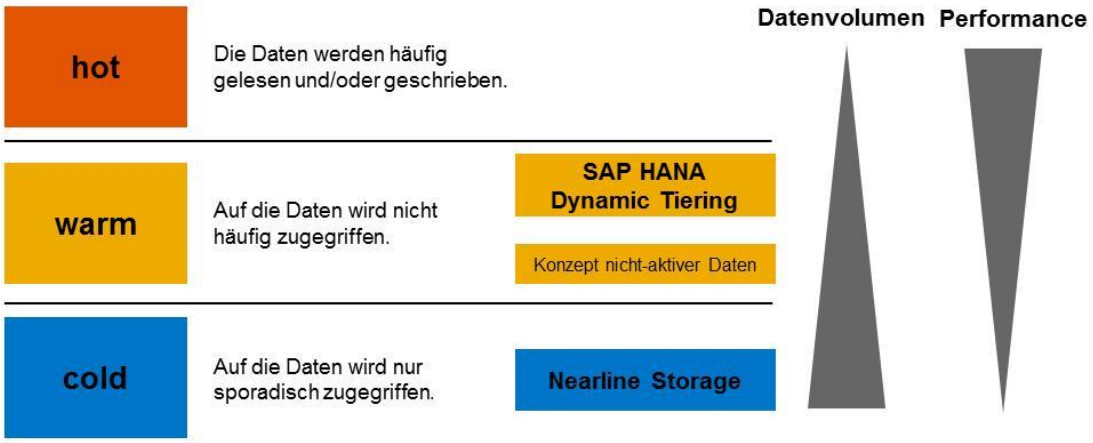

Abbildung 60: Multi-Temperature-Datenverwaltung (SAP)

Die zentrale Herausforderung bei der Realisierung einer Multi-Temperature-Datenverwaltung ist es, die einzelnen Temperaturbereiche nahtlos sowie dynamisch ineinander zu integrieren, ohne dass dem Anwender ersichtlich wird, dass im Hintergrund ein Technologiewechsel stattfindet und er womöglich aktiv verschiedene Auswertungen verwenden muss, die auf die einzelnen Temperaturbereiche zugreifen. (SAP)

Das vierte Szenario wirkt auf den ersten Blick sehr umfangreich und komplex. Bei näherer Betrachtung stellt sich jedoch heraus, dass es das Szenario ist, das auf den Status quo bezüglich der Stellung des klassischen DWH-Systems die geringsten Auswirkungen hat. Wird zunächst nur die obere Hälfte des folgenden Schaubildes betrachtet und dieses mit dem Schaubild einer klassischen Data-Warehouse-Architektur verglichen, so stellt sich heraus, dass diese identisch sind. (Tamara Dull 2015, S. 14)

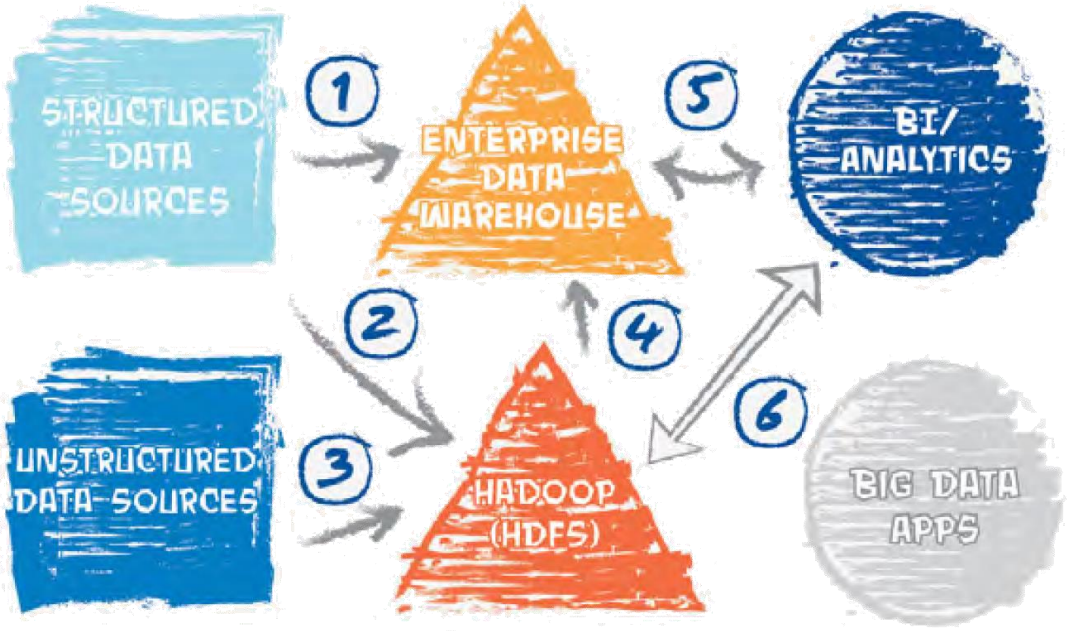

Abbildung 61: Szenario 4 – DWH als zentrale Komponente (Tamara Dull 2015, S. 14)

Abbildung 62: Klassischer Data-Warehouse-Aufbau (Tamara Dull 2015, S. 5)

Dies bedeutet, dass das bestehende Data-Warehouse-System in seiner Struktur zunächst nicht angepasst werden muss. Die vorhandenen ETL-Prozesse können weiterhin ausgeführt werden, und auch für die Anwender in der Datenauswertung ändert sich vorerst nichts. Das bestehende Data-Warehouse-System behält seine Stellung als führendes System.

Auf dem Schaubild sind die transaktionalen Systeme jedoch auch wie die unstrukturierten Datenquellen an das Hadoop-System angeschlossen, welches wiederum eine Verbindung zum Data Warehouse sowie zu den klassischen BI-Tools hat. Bei den strukturierten Daten, die nun dennoch ins Hadoop-Cluster übertragen werden, handelt es sich um jene Daten, die aktuell nicht im DWH benötigt oder dort mit anderen unstrukturierten Daten kombiniert werden sollen. Wie bereits in Szenario 2 werden die Ergebnisse der Transformationen der unstrukturierten Daten in das zentrale Data Warehouse geladen. Von dort aus können diese mit den gewohnten BI-Tools im Kontext der klassischen transaktionalen Daten analysiert und ausgewertet werden. Alternativ kann auch mit den klassischen Werkzeugen direkt auf das Hadoop-Cluster zu-

gegriffen werden, um dort erweiterte Analysen zu erstellen. Wichtig ist, dass in diesem Zusammenhang bedacht wird, dass die Daten im Hadoop-Cluster nicht derselben Datenqualität wie dem Data Warehouse entsprechen müssen. (Tamara Dull 2015, S. 14)

Auch wenn der vierte Ansatz auf dem Schaubild unübersichtlich und komplex wirkt, dürfte seine Implementierung wohl mit am einfachsten durchzuführen sein. Bestehende ETL-Prozesse müssen nicht editiert und das zentrale Data Warehouse kann wie gewohnt weiter verwendet werden. Parallel dazu wird ein eigenständiges Hadoop-System aufgebaut, das über eigene ETL-Prozesse verfügt, die größtenteils unabhängig vom klassischen Data Warehouse sind. Dieses Hadoop-System wird anschließend lediglich als weitere Quelle an das bestehende Data-Warehouse-System angeschlossen, die unstrukturierte Daten in Analyse-optimierter Struktur anbietet.

Dieses Szenario kann durchaus praktikabel sein, jedoch muss bei einem solchen Vorgehen stets abgewogen werden, welche Nachteile es mit sich bringt. Zum einen wird das zentrale Data-Warehouse-System nicht entlastet, da keine Daten aktiv in das Hadoop-Cluster ausgelagert werden. Stattdessen bildet das Hadoop-System lediglich eine weitere Quelle, die zusätzlich in das zentrale System integriert werden muss. Ein weiterer Ausbau des bestehenden Systems wird damit unausweichlich und führt somit mittelfristig zu noch höheren Betriebs- und Wartungskosten. Des Weiteren birgt das eigenständige Hadoop-System die Gefahr, dass neben dem zentralen DWH ein weiteres Schatten-Data-Warehouse abseits geltender Data-Governance-Richtlinien aufgebaut wird.

Bei der Wahl dieses Ansatzes muss also in der Projektplanung genau evaluiert werden, welche Risiken durch das gewählte Vorgehen auftreten könnten und wie diese frühzeitig zu umgehen oder vollständig zu verhindern sind. Sollte dies nicht durchgeführt werden, birgt der Ansatz die Gefahr, zusätzliche Komplexität mit in die Data-Warehouse-Systemlandschaft zu bringen. Diese zusätzliche Komplexität zu kontrollieren und zu steuern, bindet zusätzliche Ressourcen und widerspricht damit dem initialen Ansatz, das gesamte System zu verschlanken und effizienter zu betreiben.

Der fünfte und damit letzte Ansatz ist der konsequenteste Ansatz bei der Einführung von Hadoop in ein Data-Warehouse-System, hierdurch jedoch gleichzeitig auch der aufwendigste.

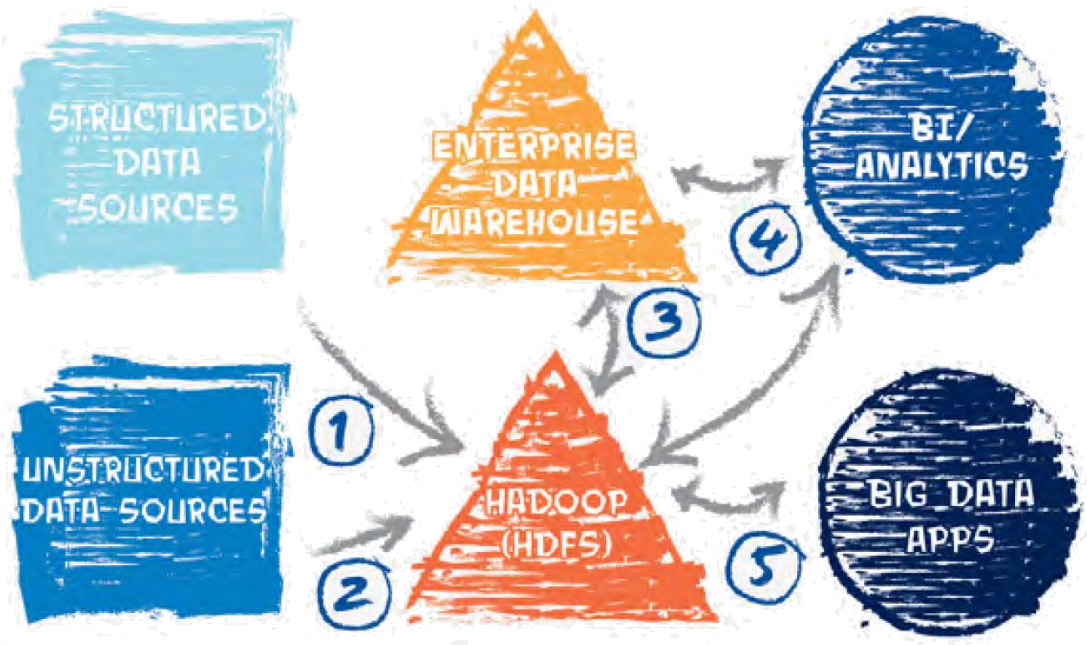

Abbildung 63: Szenario 5 – Hadoop als zentrale Plattform im Data Warehouse (Tamara Dull 2015, S. 16)

Die grundlegende Idee hinter diesem Ansatz ist einfach und schnell erläutert. Ziel des Szenarios ist es, alle ETL-Prozesse vollständig in der Hadoop-Plattform durchzuführen. Dies soll heißen, dass das bestehende Data-Warehouse-System vollständig von allen Quellsystemen abgekoppelt wird und alle bestehenden ETL-Prozesse deaktiviert werden. Im nächsten Schritt werden alle ETL-Prozesse in der Hadoop-Umgebung neu erstellt oder auf das neue Framework angepasst. Hadoop wird damit zur zentralen primären Datenquelle, in der alle Transformationen und Datenbereinigungen durchgeführt werden. Auf diesen Datenbestand kann mittels klassischer BI-Tools oder Big-Data-Apps zugegriffen werden. Die zentrale Hadoop-Plattform würde somit als eigenständiges Data-Warehouse-System fungieren, das ohne weitere klassische Data-Warehouse-Komponenten auskäme. (Tamara Dull 2015, S. 15)

Der Grund, weshalb diese jedoch weiterhin in dem Szenario enthalten ist, findet sich im Punkt Datenauswertung in der technologischen Überprüfung. In diesem wurde festgestellt, dass Hadoop zwar auf Auslesen durch klassische BI-Tools vorbereitet ist, die Antwortzeiten auf diese Anfragen jedoch weit über denjenigen von klassischen Data-Warehouse-Systemen liegen. Aus diesem Grund bietet es sich an, die Reporting-optimierten Daten weiterhin in das klassische Data-Warehouse-System zu schreiben. Aus diesem können dann die meisten aller Analyseanfragen schnell beantwortet werden. Benötigt ein Analyst für seine Auswertungen weitere Daten, kann er hierzu direkt auf das Hadoop-System gehen und in diesem die Reporting-optimierten Daten mit weiteren Daten anreichern oder diese mittels komplexer Analyseverfahren aus dem Bereich der Big-Data-Apps auswerten.

Dieses Szenario bietet somit zum einen alle Vorteile, die Hadoop mit sich bringt, und kann die bestehende Infrastruktur weiter effizient ausnutzen, indem diese nur noch als Datenhaltungskomponente für die spezialisierte Datenbereitstellung verwendet wird. Das bestehende System muss nicht erweitert werden, da alle anderen Aufgaben künftig vom grenzenlos skalierbaren Hadoop-Cluster übernommen werden.
Junge Technologien wie Apache Spark und Apache Kylin bieten zudem die Aussicht, dass künftig auch Auswertungen auf Hadoop im Bereich von wenigen Sekunden ausgeführt werden können. Mit dieser Entwicklung scheint es potenziell möglich, dass die bestehenden spezialisierten Systeme mittelfristig vollständig abgelöst werden können und das dadurch frei werdende Humankapital sich auf die Weiterentwicklung des Hadoop-Systems konzentrieren kann. Darüber hinaus sind für den Betrieb von klassischen Data-Warehouse-Systemen in der Regel Lizenzen und Wartungsgebühren erforderlich, die künftig nicht mehr gezahlt werden müssten. Das letzte Argument ist jedoch mit Vorsicht zu betrachten, da für den Aufbau eines Hadoop-Systems initial ein hoher zeitlicher sowie monetärer Aufwand einzukalkulieren ist, der bei der Gesamtrechnung betrachtet werden muss. Dieser Effekt wird zudem noch verschärft, da sich in der jüngeren Vergangenheit zeigt, dass Spezialisten im Bereich neuer Big-Data-Technologien die aktuell mit am besten bezahlten Fachkräfte im Bereich der Informationstechnologie sind.

6 Schlussbetrachtung

Zum Abschluss der Arbeit stellt sich die Frage, welche Erkenntnisse sich hinsichtlich der Zielsetzung in der Einleitung der Abhandlung ergeben haben. Eines der definierten Ziele war es, die Arbeit zu schreiben, die ich gerne selbst du Beginn meiner Recherche gelesen hätte. Es nun also zu fragen, ob mir dies in der vorliegenden Arbeit gelungen ist.

Zur Beantwortung dieser Frage muss einleitend noch einmal kurz die Intention hinter diesem Wunsch aufgezeigt werden. Als ein Business Intelligence Consultant, der von Berufswegen her ein besonderes Interesse an Data-Warehouse-Systemen hat, stellte sich mir die Frage, wie sich klassische Ansätze mit dem Hype-Thema Big Data und den damit verbundenen Technologien verbinden lassen. Bei der Einarbeitung in diese beiden Thematiken wurde schnell die Problematik hinter dieser Fragestellung ersichtlich. Der Begriff „Big Data“ selbst hat sich als nicht klar definierbar erwiesen. Es existieren viele verschieden Auffassungen, die häufig vom Hintergrund des jeweiligen Personen- und Anwenderkreis beeinflusst sind und deren persönliche Auffassung widerspiegeln. Ein weiteres Problem stellte die technologische Basis für den Begriff „Big Data“ dar. Konnte ich mich zu Beginn schnell auf Apache Hadoop als Software-Framework zur Evaluierung entscheiden, stellte sich der Umfang des gesamten Ökosystems rund um Apache Hadoop schnell als eine wesentliche Herausforderung heraus. Es erwies sich als äußerst schwierig, die vielen einzelnen Komponenten voneinander abzugrenzen und diesen konkreten fachlichen Bezügen zuzuordnen. Da Menschen neue Dinge häufig schneller verstehen und begreifen können, wenn sie diese mit bereits bekannten Dingen assoziieren können, entschloss ich mich dazu, das ganze Thema aus einer Data-Warehouse-Sicht aufzurollen.

Um die gewählte Perspektive auch für Leser verständlich zu machen, die bislang keinen Bezug zur Data-Warehouse-Thematik hatten, ist diese zunächst in den theoretischen Grundlagen ausgiebig vermittelt worden. Anschließend sind die Themen Big Data und Apache Hadoop mit dem Fokus auf ebenjenes Anwendungsszenario präsentiert worden. Hierbei stellte sich insbesondere bei der Projektion des Begriffs „Big Data“ heraus, dass dieser besser allgemein und möglichst abstrakt betrachtet werden kann, als diesen bewusst in eine Data-Warehouse-Schublade zu stecken. Anhand von Beispielen wurde versucht, zu verdeutlichen, dass Big Data zu Recht ein Thema ist, zu dem viele verschiedene Auffassungen und Ansichten bestehen und je nach Kontext definiert werden muss, was Big Data letztlich für den Einzelnen ist. Eben einer dieser Kontexte kann die Anwendung von Technologien wie Apache Hadoop in einem Data-Warehouse-Umfeld sein.

Bei der anschließenden Betrachtung der technologischen Grundlagen zu Apache Hadoop zeigte sich, dass die Data-Warehouse-Perspektive dort deutlich einfacher und vor allen Dingen sinnhafter angewendet werden konnte. Die bewusste Konzentrierung auf Komponenten, wie sie im Data-Warehouse-Umfeld eingesetzt werden, vereinfachte die Auswahl der näher zu betrachtenden Module stark. Hierdurch lassen sich Projekte wie Apache Hive oder Pig klaren Data-Warehouse-bezogenen Anwendungen zuordnen. Durch die Projektion von einzelnen Komponenten auf konkrete Aufgaben innerhalb eines Data Warehouse ergibt sich eine Struktur in der Gesamtbetrachtung, die dabei hilft, einzelne Projekte korrekt einzuordnen. Durch die Fokussierung auf Data-Warehouse-bezogene Anwendungen konnte somit ein besseres Verständnis für das gesamte Hadoop-Ökosystem geschaffen werden.

Die anschließende technologische Überprüfung der einzelnen Komponenten auf die Aufgaben Datenbewirtschaftung, Datenhaltung und Datenauswertung bestätigte, was bereits in der Einleitung erwartet wurde, und zwar, dass Apache Hadoop prinzipiell als Data-Warehouse-System eingesetzt werden kann. Hierbei wurde insbesondere auf die Punkte eingegangen, die für Personen, die aus der klassischen relationalen Welt kommen, besonders interessant sein könnten. Künftig wäre es interessant, zu verfolgen, wie sich zum einen die Integration von ACID-Transaktionen in Apache Hive (ab Version 0.14) auswirkt und sich zum anderen mit der Weiterentwicklung von Apache Kylin und Apache Spark in Antwortzeiten von Auswertungen entwickeln, sodass diese sich auch für Ad-hoc-Analysen eignen können.

Die abschließende Betrachtung von architektonischen Ansätzen für den Einsatz von Apache Hadoop in Data-Warehouse-Systemen ist als Denkanstoß zu betrachten, wie sich Hadoop fachlich sinnvoll in bestehende Systeme integrieren ließe. Bis zu diesem Punkt der Arbeit ist nur wenig auf die Fragestellung eingegangen worden, weshalb sich im Data-Warehouse-Umfeld überhaupt mit der Integration von Hadoop beschäftigt werden sollte und welche Potenziale sich hierdurch auch für bereits bestehende Systeme ergeben können. Dieser Teil ist besonders wichtig hinsichtlich der initialen Zielstellung der Arbeit. Ein potenzieller Leser soll die verschiedenen Ansätze auf seine aktuelle Situation projizieren und für sich entscheiden, wie Apache Hadoop sein Data-Warehouse-System unterstützen könnte. Das zuvor vermittelte technologische Grundwissen hilft dabei, die einzelnen Szenarien zu verstehen, und vermittelt eine Vorstellung davon, wie eine konkrete Implementierung aussehen könnte.

Abschließend kann gesagt werden, dass Apache Hadoop längst den Status eines Nischenprodukts für spezielle Anwendungsfälle hinter sich gelassen hat. Die Anzahl der vielen verschiedenen, aktiv weiterentwickelten Projekte im Ökosystem zeigt, dass die grundlegende Technologie für viele Aufgaben interessant ist. Auch die Tatsache,

dass die Anbieter von klassischen Data-Warehouse-Systemen ihre Systeme in Richtung Apache Hadoop öffnen, veranschaulicht, dass die aktuelle Entwicklung in der IT-Branche sehr ernst genommen und entsprechend darauf reagiert wird. Zuletzt hat beispielsweise die SAP mit SAP Hana Vora einen aktiven Beitrag zur Weiterentwicklung von Apache Spark geleistet. Bei Vora handelt es sich um eine Erweiterung der Spark-Ausführungs-Engine, die Welten der SAP-In-Memory-Datenbank SAP Hana und Apache Hadoop über Spark miteinander verbindet.

7 Literaturverzeichnis

Antje Höll (2005): Data Warehousing. Speichermodelle für Data-Warehouse-Strukturen. Seminar. Friedrich-Schiller-Universität Jena, Jena. Institut für Informatik. Online verfügbar unter http://www.informatik.uni-jena.de/dbis/lehre/ss2005/sem_dwh/mat/Hoell_Speichermodelle_text.pdf.

Apache Software Foundation (2013): Apache Hadoop Releases. Apache Software Foundation. Online verfügbar unter http://hadoop.apache.org/releases.html#15+October%2C+2013%3A+Release+2.2.0+available, zuletzt aktualisiert am 13.02.2016, zuletzt geprüft am 13.04.2016.

Apache Software Foundation / Cloudera Inc. (2012): A New Generation of Data Transfer Tools for Hadoop: Sqoop 2. Online verfügbar unter https://www.youtube.com/watch?v=xzU3HL4ZYI0, zuletzt geprüft am 14.03.2016.

Bauer, Andreas; Günzel, Holger (Hg.) (2013): Data-Warehouse-Systeme. Architektur, Entwicklung, Anwendung. 4., überarb. und erw. Aufl. Heidelberg: dpunkt-Verl.

BITKOM: Big Data im Praxiseinsatz - Szenarien, Beispiele, Effekte 2012. Online verfügbar unter http://www.bitkom.org/files/documents/BITKOM_LF_big_data_2012_online(1).pdf, zuletzt geprüft am 07.08.2014.

BITKOM (2014): Big Data-Technologien - Wissen für Entscheider. Online verfügbar unter http://www.bitkom.org/files/documents/BITKOM_Leitfaden_Big-Data-Technologien-Wissen_fuer_Entscheider_Febr_2014.pdf, zuletzt geprüft am 08.08.2014.

BITKOM-Arbeitskreis Big Data (2015): Big Data und Geschäftsmodell - Innovationen in der Praxis: 40+ Beispiele. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. Berlin. Online verfügbar unter https://www.bitkom.org/Publikationen/2015/Leitfaden/Big-Data-und-Geschaeftsmodell-Innovationen/151229-Big-Data-und-GM-Innovationen.pdf, zuletzt aktualisiert am 25.02.2016, zuletzt geprüft am 25.02.2016.

C. Bange (2006): Analytische Informationssysteme. Business Intelligence-Technologien und -Anwendungen. Werkzeuge für analytische Informationssysteme. 3. Auflage. Heidelberg: Springer, zuletzt geprüft am 23.11.2015.

Devlin, Barry (1997): Data warehouse. From architecture to implementation. Reading, Mass: Addison-Wesley.

Do, Hong Hai; Stöhr, Thomas; Rahm, Erhard; Müller, Robert; Dern, Gernot (2000): Evaluierung von Data Warehouse-Werkzeugen. In: Reinhard Jung und Robert Win-

ter (Hg.): Data Warehousing 2000. Methoden, Anwendungen, Strategien. Heidelberg: Physica-Verlag HD, S. 43–57. Online verfügbar unter http://dbs.uni-leipzig.de/file/Eval-DW2000.pdf, zuletzt geprüft am 20.03.2016.

Gabriel, Roland; Gluchowski, Peter; Pastwa, Alexander (2009): Data warehouse & Data Mining. Herdecke, Witten: W3L-Verl. (Informatik).

Gesellschaft für Informatik e.V. (2014): Big Data - GI - Gesellschaft für Informatik e.V. Online verfügbar unter http://www.gi.de/nc/service/informatiklexikon/detailansicht/article/big-data.html, zuletzt aktualisiert am 08.08.2014, zuletzt geprüft am 08.08.2014.

Goeken, Matthias (2006): Entwicklung von Data-Warehouse-Systemen. Anforderungsmanagement, Modellierung, Implementierung. 1. Aufl. Wiesbaden: Dt. Univ.-Verl. (Wirtschaftsinformatik).

Greg Phillips (2014): Four-step Strategy for Incremental Updates in Apache Hive. Unter Mitarbeit von Greg Phillips. Hortonworks Inc. Online verfügbar unter http://hortonworks.com/blog/four-step-strategy-incremental-updates-hive/, zuletzt geprüft am 20.03.2016.

H. Kemper:; R. Finger (2006): Analytische Informationssysteme. Business Intelligence-Technologien und -Anwendungen. Transformation operativer Daten. 3. Auflage. Heidelberg: Springer, zuletzt geprüft am 23.11.2015.

H. Mucksch (2006): Analytische Informationssysteme. Business Intelligence-Technologien und -Anwendungen. Architektur und Komponenten. 3. Auflage. Heidelberg: Springer, zuletzt geprüft am 23.11.2015.

Hans-Dieter Groffmann (1997): Das Data Warehouse Konzept. In: *HMD - Praxis Wirtschaftsinformatik* 195. Online verfügbar unter http://hmd.dpunkt.de/195/01.html.

Heise Medien GmbH & Co. KG (2015): Big Data. In: *iX Developer*.

Hortonworks: SQL to Hive Cheat Sheet. Online verfügbar unter http://hortonworks.com/wp-content/uploads/downloads/2013/08/Hortonworks.CheatSheet.SQLtoHive.pdf, zuletzt geprüft am 06.03.2015.

Hortonworks (2013): Introduction to MapReduce. Online verfügbar unter https://www.youtube.com/watch?v=ht3dNvdNDzI, zuletzt geprüft am 09.12.2015.

Hortonworks Inc. (2013): Hadoop Tutorial Apache Pig. Hortonworks Inc. Online verfügbar unter https://www.youtube.com/watch?v=PQb9I-8986s, zuletzt geprüft am 15.03.2016.

Hortonworks Inc. (2014): What is Hadoop? Open Enterprise Hadoop: The Ecosystem of Projects. Online verfügbar unter http://hortonworks.com/hadoop/, zuletzt geprüft am 01.12.2015.

IBM Analytics: Star and Snowflake Schemas. Online verfügbar unter http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/10g/r2/owb/owb10gr2_gs/owb/lesson3/starandsnowflake.htm, zuletzt geprüft am 19.05.2016.

IBM Big Data & Analytics: What is Pig? Online verfügbar unter https://www.youtube.com/watch?v=jxt5xpMFczs, zuletzt geprüft am 15.03.2016.

Inmon, W.H (1996): Building the data warehouse. 2. Auflage. New York: John Wiley & Sons. Online verfügbar unter http://tzeh.de/doc/ife-dw.pdf.

Inmon, W.H (2005): Building the data warehouse. 4th ed. Indianapolis: Wiley Publishing.

Intel IT Center (Hg.) (2012): Einführung in Big Data: Die Analyse unstrukturierter Daten. Ein Schnellkurs zum IT-Umfeld für Big Data und neuen Techniken. Online verfügbar unter http://www.intel.de/content/dam/www/public/emea/de/de/pdf/unstructured-data-analytics-paper.pdf, zuletzt geprüft am 30.04.2016.

Intricity: What is an ETL Tool? Online verfügbar unter https://www.youtube.com/watch?v=K_FCHYWGGug, zuletzt geprüft am 20.03.2016.

ITWissen. Mooresches Gesetz (2007). Online verfügbar unter http://www.itwissen.info/definition/lexikon/Mooresches-Gesetz-Moores-law.html, zuletzt aktualisiert am 08.08.2014, zuletzt geprüft am 08.08.2014.

Jesse Anderson (2013): Learn MapReduce with Playing Cards. Online verfügbar unter https://www.youtube.com/watch?v=bcjSe0xCHbE, zuletzt geprüft am 09.12.2015.

Kimball, Ralph; Ross, Margy (2013): The data warehouse toolkit. The definitive guide to dimensional modeling. 3 ed. Indianapolis, IN: Wiley. Online verfügbar unter http://lib.myilibrary.com?id=504435.

Kimble, Chris (2010): Different Types of Information System and the Pyramid Model. Online verfügbar unter http://www.chris-kimble.com/Courses/World_Med_MBA/Types-of-Information-System.html, zuletzt aktualisiert am 01.11.2010, zuletzt geprüft am 18.11.2015.

Kisker, Holger (2014): Big Data Meets Cloud. Online verfügbar unter http://blogs.forrester.com/holger_kisker/12-08-15-big_data_meets_cloud, zuletzt aktualisiert am 08.08.2014, zuletzt geprüft am 08.08.2014.

Lehner, Wolfgang (2003): Datenbanktechnologie für Data-Warehouse-Systeme. Konzepte und Methoden. 1. Aufl. Heidelberg: dpunkt-Verl. (dpunkt.lehrbuch).

Lynn Langit (2014): Hadoop MapReduce Fundamentals 2 of 5. Online verfügbar unter https://www.youtube.com/watch?v=pDGLe4CsrhY, zuletzt geprüft am 09.12.2015.

Martin Bayer (2013): Hadoop - der kleine Elefant für die großen Daten. Online verfügbar unter http://www.computerwoche.de/a/hadoop-der-kleine-elefant-fuer-die-grossen-daten,2507037, zuletzt geprüft am 01.12.2015.

Mucksch, H., Behme, W. (2000): Das Data Warehouse-Konzept. Architektur -- Datenmodelle -- Anwendungen. 4., vollständig überarbeitete und erweiterte Auflage. Wiesbaden: Gabler Verlag.

Owen O'Malley (2013): Basic Introduction to Apache Hadoop. Hortonworks Inc. Online verfügbar unter https://www.youtube.com/watch?v=OoEpfb6yga8, zuletzt geprüft am 01.12.2015.

Phil Radley (2015): Game of Elephants. data modeling zone. Hamburg, 28.09.2015, zuletzt geprüft am 10.11.2015.

Rajesh Kartha (2012): Big Data - An Introduction to Hive and HQL. IBM Analytics. Online verfügbar unter https://www.youtube.com/watch?v=ZSzl4mylPiw, zuletzt geprüft am 06.03.2016.

Sameer Farooqui (2013): Hadoop Tutorial: Intro to HDFS - YouTube. Online verfügbar unter https://www.youtube.com/watch?v=ziqx2hJY8Hg, zuletzt geprüft am 01.12.2015.

SAP: Multi-Temperature-Datenverwaltung. Online verfügbar unter https://help.sap.com/saphelp_nw74/helpdata/de/f6/99e81daef24dfa8414b4e104fd76b7/content.htm, zuletzt geprüft am 30.04.2016.

SAP (2015): Leveraging SAP, Hadoop, and Big Data to Redefine Business. Online verfügbar unter http://de.slideshare.net/Hadoop_Summit/leveraging-sap-hadoop-and-big-data-to-redefine-business-49578994, zuletzt geprüft am 06.03.2016.

SAP HANA Vora | Hadoop | In-Memory-Abfrage. Online verfügbar unter http://go.sap.com/germany/product/data-mgmt/hana-vora-hadoop.html, zuletzt geprüft am 06.03.2016.

SAP SE (2015a): Information Design Tool User Guide. SAP SE. Online verfügbar unter http://help.sap.com/businessobject/product_guides/sbo42/en/sbo42_info_design_tool_en.pdf, zuletzt geprüft am 02.05.2016.

SAP SE (2015b): SAP BusinessObjects Business Intelligence Suite Master Guide. SAP SE. Online verfügbar unter https://websmp202.sap-

ag.de/~sapidb/012002523100015754782015E/sbo42_master_en.pd, zuletzt geprüft am 02.05.2016.

Sébastien Jelsch: Apache Kylin: OLAP im Big-Data-Maßstab |. Hg. v. heise Developer. Heise Medien. Online verfügbar unter http://www.heise.de/developer/artikel/Apache-Kylin-OLAP-im-Big-Data-Massstab-2824878.html?view=print, zuletzt geprüft am 14.04.2016.

Tamara Dull (2015): Hadoop and the Enterprise Data Warehouse. A Non-Geek's Big Data Playbook. SAS Institute Inc. Online verfügbar unter http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/non-geeks-big-data-playbook-106947.pdf, zuletzt aktualisiert am 15.04.2016, zuletzt geprüft am 15.04.2016.

The Apache Software Foundation (2015): Welcome to Apache™ Hadoop®! Online verfügbar unter http://www.tpc.org/tpctc/tpctc2009/tpctc2009-17.pdf, zuletzt aktualisiert am 31.10.2015, zuletzt geprüft am 10.11.2015.

Tom Deutsch (2013): Why is Schema on Read So Useful? A primer on why flexibility—not scale—often drives big data adoption. IBM Big Data & Analytics. Online verfügbar unter http://www.ibmbigdatahub.com/blog/why-schema-read-so-useful, zuletzt aktualisiert am 06.03.2016, zuletzt geprüft am 06.03.2016.

Zeh, Thomas (2003): Data Warehousing als Organisationskonzept des Datenmanagements. Eine kritische Betrachtung der Data-Warehouse-Definition von Inmon. In: *Informatik - Forschung und Entwicklung* 18 (1), S. 32–38. DOI: 10.1007/s00450-003-0130-8.

Anhang

Anhang A – SQL-Abfragen

Einfacher Select

```
select  *
from    dimaccount
```

Einfaches Group By

```
SELECT dimaccount.valuetype,
                count(dimaccount.accountkey)
FROM dimaccount
GROUP BY dimaccount.valuetype
```

Einfacher Map Join

```
SELECT dimorganization.organizationname,
                dimcurrency.currencyname
FROM dimcurrency,
                dimorganization
WHERE (dimcurrency.currencykey = dimorganization.currencykey)
```

Komplexer Map Join

```
SELECT dimaccount.accountcodealternatekey,
                dimdate.monthnumberofyear,
                dimdepartmentgroup.departmentgroupname,
                dimscenario.scenarioname
FROM
        factfinance,
        dimdate,
        dimaccount,
        dimdepartmentgroup,
        dimscenario
WHERE
        ( dimdate.datekey=factfinance.datekey ) AND
        ( dimscenario.scenariokey=factfinance.scenariokey ) AND
        ( dimaccount.accountkey=factfinance.accountkey ) AND
        ( dimdepartmentgroup.departmentgroupkey=factfinance.departmentgroupkey
)
```

Komplexe Analyse Abfrage

```
SELECT
 dimdate.fiscalyear,
 dimdate.englishmonthname,
 dimorganization.organizationname,
 dimdepartmentgroup.departmentgroupname,
 sum(case when factfinance.scenariokey = 2 then factfinance.amount else cast(null as signed) end),
 sum(case when factfinance.scenariokey = 1 then factfinance.amount else cast(null as signed) end),
 dimcurrency.currencyname
FROM
 dimcurrency INNER JOIN dimorganization ON (dimcurrency.CURRENCYKEY=dimorganization.CURRENCYKEY)
 INNER JOIN factfinance ON (dimorganization.ORGANIZATIONKEY=factfinance.ORGANIZATIONKEY)
 INNER JOIN dimdepartmentgroup ON (dimdepartmentgroup.DEPARTMENTGROUPKEY=factfinance.DEPARTMENTGROUPKEY)
 INNER JOIN dimdate ON (dimdate.DATEKEY=factfinance.DATEKEY)
 INNER JOIN dimaccount ON (dimaccount.ACCOUNTKEY=factfinance.ACCOUNTKEY)

WHERE
 dimaccount.valuetype = 'Currency'
GROUP BY
 dimdate.fiscalyear,
 dimdate.englishmonthname,
 dimorganization.organizationname,
 dimdepartmentgroup.departmentgroupname,
 dimcurrency.currencyname,
 dimdate.monthnumberofyear
ORDER BY
 1,
 dimdate.monthnumberofyear
```

Anhang B - Prototypische Implementierung eines Apache Hadoop Clusters auf Basis von ARM Kleinstrechnern

Adobe Acrobat Document

Der Anhang findet sich in auf Grund des Umfanges in digitaler Form auf der beigefügten CD. Sollte die Ausarbeitung digital als PDF betrachtet werden, kann der Anhang direkt aus der Datei heraus geöffnet werden.

Anhang C - Installationsleitfaden Prototypische Installation eines Apache Hadoop Clusters

Adobe Acrobat Document

Der Anhang findet sich in auf Grund des Umfanges in digitaler Form auf der beigefügten CD. Sollte die Ausarbeitung digital als PDF betrachtet werden, kann der Anhang direkt aus der Datei heraus geöffnet werden.

Ehrenwörtliche Erklärung

Ich versichere, dass ich diese Projektarbeit selbstständig angefertigt, alle Hilfen und Hilfsmittel angegeben und alle wörtlich oder dem Sinne nach aus Veröffentlichungen oder anderen Quellen sowie auch aus dem Internet entnommenen Inhalte kenntlich gemacht habe.

Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird.

Düsseldorf, den 20.05.2016

Unterschrift