Fachhochschule Köln
Cologne University of Applied Sciences
Campus Gummersbach
Fakultät für Informatik und Ingenieurwissenschaften

Fachhochschule Dortmund
University of Applied Sciences and Arts
Fachbereich Informatik

Verbundstudiengang Wirtschaftsinformatik

Abschlussarbeit

zur Erlangung

des Mastergrades

Master of Science

in der Fachrichtung Informatik

„Erweiterung eines Data Warehouse mit Big-Data-Quellen am Beispiel Twitter“

Erstprüfer:	Prof. Dr. Heide Faeskorn-Woyke
Zweitprüfer:	Wolfgang Rütter
vorgelegt am:	23.06.2015
von cand.	Martin Frisch
aus	Friedlandstr. 9
	51067 Köln
Tel.:	01786945773
Email:	martin.frisch@smail.fh-koeln.de
Matr.-Nr.:	11065815

Zusammenfassung

Im Zusammenhang mit dem Begriff Big Data können nicht nur immer größere Datenmengen verarbeitet werden, sondern auch neue Arten von Datenquellen genutzt werden. Insbesondere Web 2.0-Inhalte bieten dabei vielfältige Potenziale. So können beispielsweise mit Hilfe einer Sentiment-Analyse Meinungen und Stimmungen zu Produkten und Unternehmen in sozialen Netzwerken beobachtet werden. Diese Informationen sind für sich gesehen bereits wertvoll für viele Unternehmen. Jedoch ist eine effiziente Analyse und Auswertung der Informationen nur in Kombination mit weiteren Unternehmensdaten möglich, die typischerweise in einem Data Warehouse liegen. Diese Arbeit diskutiert die Unterschiede, Möglichkeiten und Herausforderungen diese Kombination zu realisieren. Veranschaulicht wird dies durch einen Show-Case, der eine Ende-zu-Ende-Umsetzung am Beispiel der Fernsehsendung Tatort zeigt. Dabei werden Zuschauerkommentare aus Twitter extrahiert, mit einer Sentiment-Analyse bewertet und schließlich in einem Data Warehouse ausgewertet. Dabei können klassische BI-Kennzahlen, wie beispielsweise Einschaltquoten, Folgen pro Ermittler etc. den Ergebnissen der Sentiment-Analyse gegenübergestellt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
BI	Business Intelligence
BSC	Balanced Scorecard
CAP	Consistency, Availability, Partition Tolerance
CIF	Corporate Information Factory
DDL	Data Definition Language
DWH	Data Warehouse
ER	Entity Relationship
ERP	Enterprise Ressource Planning
ETL	Extrahieren Transformieren Laden
EVA	Eingabe, Verarbeitung, Ausgabe
GFS	Google Filesystem
HDFS	Hadoop Distributed Filesystem
HOLAP	Hybrid Online Analytical Processing
IDC	International Data Corporation
IT	Informationstechnologie
JSON	JavaScript Object Notation
KB	Kilobyte
KDD	Knowledge Discovery in Databases
MB	Megabyte
MOLAP	Multidimensional Online Analytical Processing
NLP	Natural Language Processing
NoSQL	Not Only Structured Query Language
OAUTH	Open Authorization
OBIEE	Oracle Business Intelligence Enterprise Edition
ODI	Oracle Data Integrator
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
PMI	Pointwise Mutual Information
RDBMS	Relationales Datenbankmanagementsystem
REST	Representational State Transfer
ROLAP	Relational Online Analytical Processing
SerDe	Serializer/Deserializer
SPOF	Single Point of Failure
SPOT	Single Point of Truth
SQL	Structured Query Language
UML	Unified Modelling Language
URL	Uniform Resource Locator
UTC	Universal Time Coordonné (Koordinierte Weltzeit)
XML	Extensible Markup Language
YARN	Yet Another Resource Negotiator

1 Einleitung

Bereits seit Anfang der 1990er Jahre existiert der Begriff Data Warehouse (DWH).[1] Seither gilt dieses als wesentliche Komponente zur Datenbereitstellung in Applikationen zur Entscheidungsunterstützung. In der Vergangenheit nutzte es dabei vor allem wohlstrukturierte Daten aus operativen, transaktionalen Quellsystemen. Allerdings existierten neben diesen innerhalb und außerhalb von Unternehmen ebenfalls umfangreiche Datenmengen weniger strukturierter Daten (z.B. Texte, Bilder, Videos). Insbesondere durch das Web 2.0 stehen bereits in digitaler Form eine Vielzahl von Kommentaren und Meinungen zur Verfügung, die durch Nutzer und Kunden in Foren, sozialen Netzwerken oder Blogs erstellt werden. Die Speicherung, Verarbeitung und Auswertung solcher Daten bedeutet für Unternehmen sowohl neue Potentiale als auch neue Herausforderungen. Unter dem Begriff Big Data haben sich einige Methoden, Techniken und Werkzeuge angesammelt, die beim Bewältigen dieser Herausforderungen unterstützen können. Einige von diesen werden in dieser Arbeit in ein Data Warehouse integriert und genutzt, um zusätzliche Potenziale für die Entscheidungsunterstützung zu generieren.

Diese Arbeit gliedert sich inklusive dieser Einleitung in sechs Kapitel. Im zweiten Kapitel werden relevante Eigenschaften und Begrifflichkeiten zu Daten geklärt. Kapitel 3 stellt die wesentlichen Aspekte eines klassischen Data Warehouse vor. Herausforderungen, die dazugehörigen Methoden, Techniken und exemplarische Werkzeuge, die notwendig sind, um die Potentiale von Big Data zu nutzen, werden in Kapitel 4 thematisiert. Dieses schließt mit Möglichkeiten ab, solche in eine Data-Warehouse-Umgebung zu integrieren. Praktisch wird dies im fünften Kapitel an einem Fallbeispiel gezeigt. Bei diesem wird ein Data Warehouse zur Fernsehsendung Tatort aufgebaut und am Beispiel Twitter um eine Web 2.0-Quelle erweitert. Ziel dabei ist die Stimmung der Twitter-Gemeinde zur Serie mit im Data Warehouse auszuwerten. Ein Fazit sowie ein Ausblick zu dieser Arbeit werden schließlich in Kapitel 6 gegeben.

---

[1] Vgl. Kemper et al. 2010, S. 12.

2 Die Welt der Daten

Dieses Kapitel stellt aus verschiedenen Blickwinkeln dar, was Daten sind. Dazu findet in Abschnitt 2.1 eine Abgrenzung von den Begriffen Informationen und Wissen statt. Verschiedene Dimensionen von Daten werden in Abschnitt 2.2 vorgestellt. In Abschnitt 2.3 werden gängige Datenmodelle betrachtet.

2.1 Daten, Informationen, Wissen

Zwischen den Begriffen Daten, Information und Wissen besteht ein hierarchischer Zusammenhang, der in Abbildung 1 dargestellt wird.

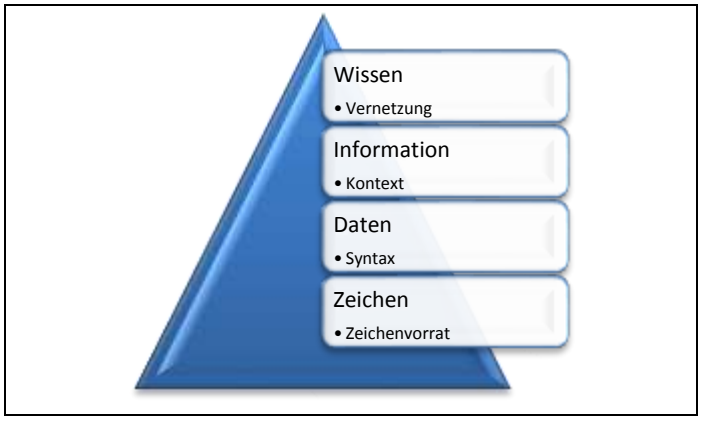

Abbildung 1: Begriffshierarchie Zeichen, Daten, Information, Wissen[2]

Auf der untersten Ebene befindet sich der zur Verfügung stehende Zeichenvorrat. Dieser setzt sich üblicherweise aus Ziffern, Buchstaben und weiteren Satz- oder Sonderzeichen zusammen. Werden Zeichen in einer bestimmten Syntax angeordnet (z.B. 0,99) ergeben sich Daten. Um aus solchen Daten eine Information zu gewinnen, müssen diese einem bestimmten Kontext zugeordnet werden. Zum Beispiel 0,99 könnte dieser lauten, dass es der Kilogrammpreis in Euro Tomaten ist. Wissen wiederum entsteht, wenn mehrere Informationen miteinander verknüpft werden. Bezogen auf das Beispiel könnte eine weitere Information lauten, dass in der vorherigen Woche der Preis für ein Kilogramm Tomaten 1,99 € war und sich daraus das Wissen ableiten, dass der aktuelle Preis von 0,99 € günstig ist.

[2] In Anlehnung an Krcmar 2015, S. 4.

2.2 Dimensionen von Daten

In diesem Abschnitt werden wichtige Eigenschaften bzw. Dimensionen von Daten vorgestellt und mögliche Ausprägungen innerhalb dieser unterschieden.

Die erste Dimension ist der **Strukturierungsgrad**. Zu unterscheiden sind dabei strukturierte, semi-strukturierte und unstrukturierte bzw. polystrukturierte Daten. Für strukturierte Daten ist vollständig bekannt, wie diese angeordnet sind, welche Bedeutung diese haben und in welcher Verbindung sie zu anderen Daten stehen. Solche Daten werden üblicherweise in einer einheitlichen Form in relationalen Datenbanken abgelegt. Zu den unstrukturierten Daten zählen dagegen Freitexte, Bilder, Videos oder Audiodaten. Für diese ist die Struktur nicht bekannt. Dies muss nicht bedeuten, dass sie keine haben. Sie ist allerdings auf einer höheren Abstraktionsebene zu finden.[3] So ist ein Fließtext in Kapiteln, Abschnitten und Sätzen gegliedert. Die Sätze erhalten wiederum durch grammatikalische Regeln der jeweiligen Sprache eine gewisse Struktur. Daher ist der Begriff unstrukturiert nicht ganz unumstritten und es ist in der Literatur auch der Begriff polystrukturiert für solche Daten zu finden.[4] Dennoch ist festzuhalten, dass diesen Daten zunächst keine Schemata, Metadaten oder Glossare beiliegen, die diese beschreiben. Zudem ist es nicht leicht solche Daten mit den Standardfunktionen einer Datenbank zu verarbeiten.[5] Zusätzlich existieren auch semi-strukturierte Daten, die sowohl Eigenschaften von strukturierten als auch unstrukturierten Daten haben. Sie zeichnen sich dadurch aus, dass eine gewisse Struktur vorliegt oder die Daten mit erklärenden Zusatzinformationen angereichet sind, aber auch flexible Bestandteile vorkommen. Beispiele sind XML- (Extensible Markup Language), JSON- (JavaScript Object Notation), Log- oder Sensordaten.[6]

Eine weitere Unterscheidung von Daten ist über ihren **Verwendungszweck** möglich. Dabei können operative von dispositiven Daten unterschieden werden. Operative Daten dienen zur Unterstützung der wertschöpfenden Geschäftsprozesse

---

[3] Vgl. Spies 2012.

[4] Vgl. Lang 2012, S. 143.

[5] Vgl. Inmon et al. 2008, S. 35.

[6] Vgl. Vorhies 2013.

eines Unternehmens und sind üblicherweise transaktionsorientiert.[7] Ein Beispiel können Daten zu einer Bestellung (Artikel, Menge, etc.) in einem Logistikunternehmen sein. Aufgrund ihres Transaktionsbezuges werden operative Daten auch als OLTP-Daten (Online Transaction Processing) bezeichnet. Dispositive Daten dienen dagegen der Unterstützung des Managements bei der Entscheidungsfindung. Sie werden von sogenannten OLAP-Anwendungen (Online Analytical Processing) genutzt. Typischerweise werden diese für eine Berichtserstellung dediziert vorgehalten, angereichert, bereinigt und verdichtet.[8]

Als Gegenstück zum Verwendungszweck können Daten auch nach ihrer **Herkunft** unterschieden werden (Vgl. Abbildung 2).

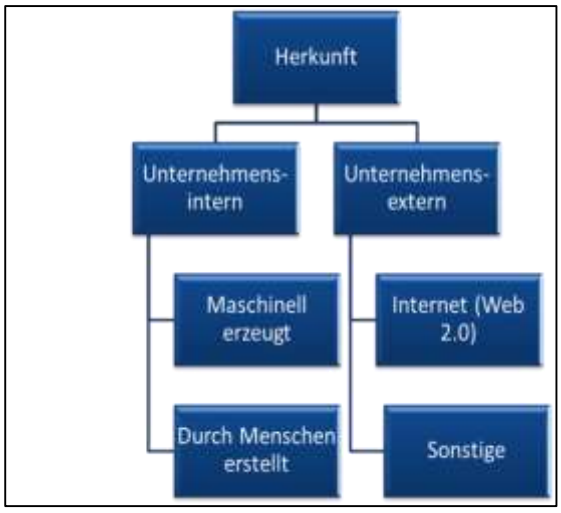

Abbildung 2: Datenherkunft

Zunächst kann dabei unterschieden werden, ob die Daten im Unternehmen selbst oder außerhalb von diesem erstellt werden. Bei den unternehmensinternen Daten kann nochmals zwischen maschinell- und menschengenerierten Daten unterschieden werden. Ein Beispiel für erstere sind Logdateien, die von Maschinen, Anlagen oder Geräten verschiedenster Art erzeugt werden (z.B. Produktionsmaschinen oder Telekommunikationsanlagen). In diesem Zusammenhang ist das Thema „Internet der Dinge“ bzw. „Internet of Things“ (IoT) zu erwähnen, welches für die Kommunikation mit und zwischen Geräten bzw. Maschinen steht. Es wird angenommen, dass durch das IoT die Anzahl der kommunizierenden Maschinen und

[7] Vgl. Kemper / Finger 2010, S. 160.

[8] Vgl. Manhart 2008.

somit auch das Volumen der auf diese Weise erzeugten Daten in den kommenden Jahren stark ansteigen wird.[9] Zu den durch Menschen erstellten Daten zählen neben verschiedenen Arten von Dokumenten auch E-Mails, Chat-Protokolle, unternehmensinterne Foren, Wissensmanagement-Plattformen etc. Unternehmensexterne Daten können zwar vielfältiger Herkunft sein, aber gerade im Zusammenhang mit der Web 2.0-Entwicklung sind an dieser Stelle durch Nutzer generierte Inhalte im Internet hervorzuheben. Diese stammen vor allem aus sozialen Netzwerken, Internetforen, Blogs und Bewertungsportalen. Alle haben gemeinsam, dass Nutzer dort von ihren Erfahrungen, Meinungen und Neigungen berichten. Andere unternehmensexterne Daten können frei verfügbare oder gekaufte Daten verschiedenster Art sein (z.B. geographische Daten, Wetterdaten, usw.).

2.3 Datenmodelle

Datenmodelle bilden die Grundlage einer Datenbank. Sie legen die Modellierungskonstrukte fest, mit denen ein Ausschnitt aus der realen Welt in der Datenbank abgebildet werden kann.[10] Dabei können mit einem Datenmodell Objekte, ihre Eigenschaften und Beziehungen zu weiteren Objekten beschrieben werden.[11] Zu unterscheiden sind konzeptionelle, logische und physische Datenmodelle. Erstere sind implementierungsunabhängig und beschreiben meist mit Hilfe einer grafischen Modellierung, wie z.B. ER-Diagrammen (Entity-Relationship-Diagrammen) oder UML (Unified Modelling Language), die Wirklichkeit. Um diese in logische Datenmodelle zu überführen, müssen die Regeln des zugrundeliegenden Datenbanksystems berücksichtig werden. Im Gegensatz zum physischen Datenmodell ist das logische jedoch weitestgehend herstellerunabhängig. Details zur physischen Speicherung in der Datenbank und zur Optimierung des Zugriffs (z.B. spezielle Indexstrukturen oder Partitionierungsmöglichkeiten) werden somit erst im physischen Datenmodell berücksichtigt. Im Folgenden wird das logische Datenmodell fokussiert. Dabei werden einige Beispiele und deren Besonderheiten betrachtet.

---

[9] Vgl. Ladner 2014.

[10] Vgl. Kemper / Eickler 2013, S. 25.

[11] Vgl. Hahne 2010, S. 230.

Eines der bekanntesten Vertreter ist das **relationale Datenmodell**. In diesem werden die Daten und ihre Beziehungen in Tabellen gespeichert. Dabei heißen die Tabellen Relationen, ihre Zeilen sind Tupel und Spalten entsprechen ihren Attributen.[12] Das relationale Datenmodell basiert auf der relationalen Algebra, die von Codd bereits 1970 entwickelt wurde.[13] Der Datenzugriff geschieht über die Abfragesprache SQL (Structured Query Language). Diese gilt als Programmiersprache der vierten Generation und wird von allen gängigen relationalen Datenbankmanagementsystemen (RDBMS) unterstützt.[14]

Angelehnt an die stark verbreiteten objektorientierten Programmiersprachen existieren auch **objektorientierte** und **objektrelationale** Datenmodelle. Bei objektorientierten Datenmodellen werden die Daten als Objekte zusammen mit ihren Methoden (mögliche Operationen) und Attributen gespeichert. Dabei können analog zu objektorientierten Programmiersprachen eigene Objekte definiert werden. Beim objektrelationalen Datenmodell wird das relationale Datenmodell um Konzepte aus dem objektorientierten Datenmodell erweitert.[15]

Insbesondere zur Auswertung und Speicherung von dispositiven Daten kommen **multidimensionale Datenmodelle** in OLAP-Anwendungen zum Einsatz.[16] Dabei werden die Daten in einer Würfelstruktur gespeichert. Zu unterscheiden sind in einem Würfel Kennzahlen (Fakten) und Dimensionen. Dimensionen bilden die Achsen des Würfels und basieren typischerweise auf Stammdaten (z.B. Zeit, Region, Produkt). An den Schnittpunkten der Dimensionen befinden sich die Faktenwerte (z.B. Umsätze). Ein multidimensionales Datenmodell ist dabei nicht durch die drei Dimensionen eines Würfels begrenzt, sondern kann auch über weitere verfügen. Ein komplett multidimensionales Datenmodell wird auch als MOLAP (Multidimensional Online Analytical Processing) bezeichnet. Zudem existiert unter dem Namen ROLAP (Relational Online Analytical Processing) ein alternativer Ansatz, der es ermöglicht multidimensionale Strukturen in einer relationalen Datenbank zu speichern. Eine Mischform aus ROLAP und MOLAP, in

---

[12] Vgl. Faeskorn-Woyke et al. 2007, S. 32.

[13] Vgl. Saake et al. 2013, S. 85.

[14] Vgl. Saake et al. 2013, S. 211.

[15] Vgl. Faeskorn-Woyke et al. 2007, S. 32f.

[16] Vgl. Hahne 2010, S. 255.

der ein Teil der Daten in einer multidimensionalen Speicherstruktur abgelegt ist und der andere in einer relationalen, nennt sich HOLAP (Hybrid Online Analytical Processing). Eine sinnvolle Möglichkeit ist es aggregierte Daten, die häufig abgefragt werden, in MOLAP und die Detaildaten in ROLAP abzulegen.

Weitere Datenmodelle können als **Schlüssel-Wert-Modelle** zusammengefasst werden. Dabei handelt es sich um eine generische Speicherstruktur, die lediglich aus einem Schlüssel und dem zu speichernden Wert besteht. Zu beachten ist, dass beide Bestandteile komplexer sein können als ein primitiver Datentyp. Ein Beispiel für solch einen Ansatz ist das Datenmodell Entität-Attribut-Wert bzw. Entity-Attribute-Value (EAV). Dabei werden alle Daten in Tabellen und einer Entität-Attribut-Kombination als Schlüssel abgelegt. Die Entität verweist dabei auf das eigentliche Objekt oder die Person, auf die sich der Eintrag bezieht. Dies könnte in einer Kundentabelle beispielsweise eine eindeutige Kundennummer sein. Attribut ist der Name der Eigenschaft oder des Parameters (z.B. das Alter des Kunden), zu dem der eigentliche Wert (z.B. 30 für das Kundenalter) gespeichert wird.[17]

Neben den genannten existieren eine ganze Reihe weiterer Ansätze, die hier nicht weiter vertieft werden. Zu diesen zählen beispielsweise die beiden älteren Ansätze Netzwerkdatenmodell und das hierarchische Datenmodell oder auch Ansätze, die sich auf bestimmte Anwendungsfälle spezialisiert haben (z.B. dokumentenorientierte Datenmodelle oder Datenmodelle zur Abbildung von Beziehungen in Graphen).

[17] Vgl. Nadkarni 2015.

3 Klassisches Data Warehouse

In diesem Kapitel wird das klassische Data Warehouse betrachtet. Dabei wird in Abschnitt 3.1 zunächst eine Erläuterung und Abgrenzung des Begriffes DWH gegeben. In Abschnitt 3.2 werden anhand von möglichen Architekturen die Komponenten zur Datenspeicherung und -verarbeitung innerhalb eines DWH betrachtet. Zuletzt wird in Abschnitt 3.3 auf die Analyse der Daten aus einem DWH mit Möglichkeiten aus dem Data Mining eingegangen.

3.1 Abgrenzung

Um den Begriff Data Warehouse zu erläutern wird mit einer Definition nach Inmon begonnen, mit der er Anfang der 1990er den Begriff prägte:

„A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support management´s decisions."[18]

Bei dieser Definition sind fünf Eigenschaften zu erkennen. Im Gegensatz zu operativen Anwendungen dient ein DWH nicht der Unterstützung der primären Geschäftsprozesse, sondern der Erstellung von **fachorientierten** (subject-oriented) bzw. themenorientierten Auswertungen zur **Entscheidungsunterstützung** (support management´s decisions). Aus verschiedenen Quellsystemen werden Daten in einem DWH **integriert** (integrated) und dort **nicht-flüchtig** (nonvolatile) gespeichert. Daten, die sich in einem DWH befinden, werden nicht mehr gelöscht. Stattdessen werden Daten in einem DWH **historisiert** (time variant). Dies ermöglicht Vergleiche über die Zeit und Beobachtungen von Entwicklungen.

Somit werden in einem Data Warehouse meist dispositive Daten vorgehalten, die das Management bei der strategischen Entscheidungsfindung unterstützen. Dabei ist das Data Warehouse alleine nur die Datenhaltungskomponente. Wird es darüber hinaus von Komponenten ergänzt, welche neben der Datenhaltung auch Aufgaben zur Datenintegration und Steuerung der Datenflüsse umfassen, ist die Rede von einem Data-Warehouse-System.[19] Noch weitreichender ist der Begriff Business Intelligence (BI). Dieser steht für unterschiedliche Technologien und Kon-

---

[18] Inmon 2005, S. 29.

[19] Vgl. Navrade 2008, S. 14.

zepte, bei denen relevante Daten gesammelt, aufbereitet, analysiert und zur Entscheidungsunterstützung präsentiert werden.[20] Somit kann das Data-Warehouse-System als Speicher- und Datenaufbereitungskomponente innerhalb eines BI-Systems betrachtet werden. Jedoch umfasst das BI-System darüber hinaus noch weitere Komponenten zur Analyse und Präsentation der Daten.

3.2 Architekturen

Für Data-Warehouse-Systeme existieren verschiedene Architekturansätze. Jedoch werden meist ähnliche Komponenten verwendet. Anhand eines gängigen Referenzmodells wird zunächst in Abschnitt 3.2.1 ein Überblick über die typischen Komponenten eines Data-Warehouse-Systems gegeben. Diese Komponenten werden dann in den darauf folgenden Abschnitten 3.2.2 bis 3.2.5 im Detail betrachtet und verschiedene alternative Ansätze bzw. Auffassungen diskutiert.

3.2.1 Referenzmodell

In Abbildung 3 ist ein Referenzmodell für Data-Warehouse-Systeme dargestellt. In diesem ist eine Verteilung auf die vier Bereiche externe Systeme, Integrationsbereich, Auswertebereich und Verwaltungsbereich zu erkennen. Innerhalb der verschiedenen Bereiche befinden sich verschiedene Speicher- und Aufgabenkomponenten. Verknüpft sind diese durch drei verschiedene Verbindungstypen (Datenflüsse, Metadatenflüsse und Kontrollflüsse).

[20] Vgl. Gluchowski et al. 2008, S. 93.

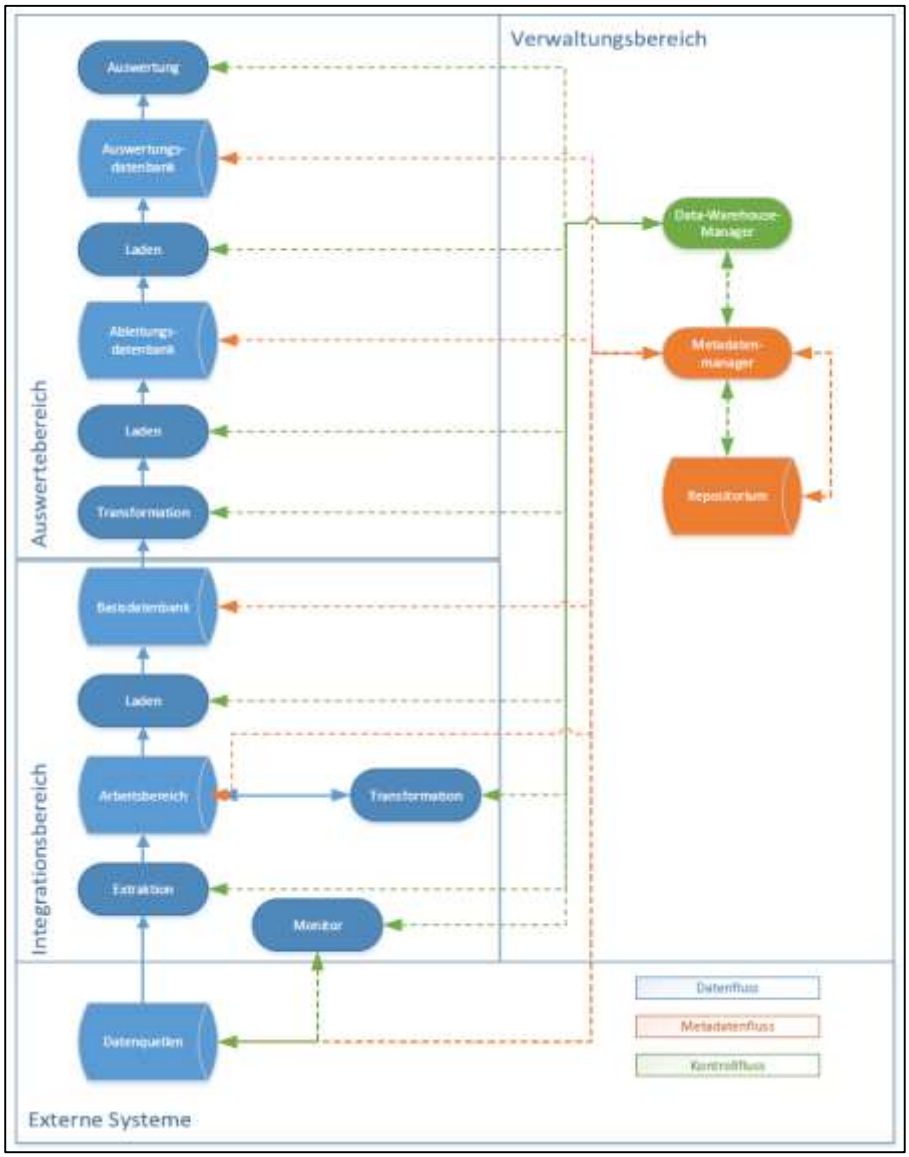

Abbildung 3: Referenzarchitektur für Data-Warehouse-Systeme[21]

Der Datenfluss beginnt mit der Datenextraktion aus meist unterschiedlichen externen Systemen. Daher sind diese auch mitdargestellt, obwohl sie grundsätzlich außerhalb vom eigentlichen Data-Warehouse-System liegen und keine direkte Komponente sind.[22] Im Integrationsbereich werden die Daten harmonisiert und zusammengeführt. Weitere Aufbereitungen, die auf spätere Auswertungen abzielen, finden im Auswertebereich statt. Die komplette Ausführung wird vom Verwaltungsbereich gesteuert und überwacht.

---

[21] Bauer / Günzel 2013, S. 42.

[22] Vgl. Bauer / Günzel 2013, S. 45.

3.2.2 Verwaltungsbereich

Der Verwaltungsbereich übernimmt sämtliche administrative Aufgaben in einem Data-Warehouse-System. Dabei werden diese auf den Data-Warehouse-Manager und den Metadaten-Manager verteilt.

Der Data-Warehouse-Manager übernimmt die Steuerung und Koordination der Komponenten, die Aufgaben im System erfüllen. Dementsprechend ist dieser auch mit diesen über einen Kontrollfluss verknüpft. Eine wesentliche Aufgabe ist dabei das Initiieren der Lade- und Verarbeitungsprozesse.[23] Typischerweise werden diese an bestimmten Zeitpunkten oder durch Ereignisse gestartet. Beispielsweise können zeitpunktgesteuert Ladeprozesse täglich, wöchentlich oder monatlich zu einer bestimmten Uhrzeit stattfinden. Ein relevantes Ereignis kann hingegen der Abschluss einer Verarbeitung im Vorsystem oder eine gewisse Anzahl an Änderungen in den Quelldaten sein. Zudem ist auch ein manueller Start der Verarbeitungen möglich.

Dagegen ist der Metadaten-Manager für die Verwaltung der Metadaten verantwortlich. Diese stellen sowohl betriebswirtschaftlich-semantische als auch technisch-strukturelle Beschreibungen der eigentlichen Daten im Data Warehouse zur Verfügung.[24] Erstere sind dabei inhaltliche Informationen (z.B. die Berechnungsvorschrift für eine bestimmte Kennzahl). Zu den technisch-strukturellen Beschreibungen zählen dagegen Tabellenstrukturen, Datentypen oder auch welche Prozesse welche Daten verarbeiten. Aufgabe des Metadaten-Managers ist es diese Metadaten im Repositorium abzulegen, abzufragen und bei Bedarf bereitzustellen.

3.2.3 Externe Systeme

Im Bereich der externen Systeme befinden sich die Datenquellen. Wie bereits erwähnt, stellen diese aus Datenflusssicht den Ausgangspunkt eines Data-Warehouse-Systems dar. Typischerweise existieren Schnittstellen zu unterschiedlichen operativen Systemen. Dabei kann es sich in einfachen Fällen um gleichartige Systeme aus unterschiedlichen Abteilungen oder Regionen handeln. So kann beispielsweise in einem internationalen Unternehmen jedes Land über ein eigenes

---

[23] Vgl. Bauer / Günzel 2013, S. 44.

[24] Vgl. Bange 2010, S. 133.

ERP-System (Enterprise Ressource Planning) verfügen, welche alle vom gleichen Typ sind und somit auch einheitliche Schnittstellen anbieten. Dagegen können die Quellsysteme auch sehr heterogen sein. Diese Heterogenität kann technisch, strukturell, syntaktisch und semantisch sein.[25] So können in den Quellsystemen unterschiedliche (Datenbank-) Technologien eingesetzt werden, die ebenfalls zu unterschiedlichen Eigenschaften (Protokolle, Abfragesprachen, Formate, etc.) innerhalb der Schnittstellen führen. Eine unterschiedliche Struktur, kann sich beispielsweise durch die Verwendung unterschiedlicher Datenmodelle (Vgl. Abschnitt 2.3) ergeben. Selbst wenn der gleiche Datenmodellierungstyp (z.B. dritte Normalform) verwendet wird, können unterschiedliche Systeme ebenfalls einen unterschiedlichen Fokus haben. So wird eine Kundentabelle in einem Buchhaltungssystem und einem Warenwirtschaftssystem unterschiedlich aufgebaut sein. Ein Beispiel für syntaktische Unterschiede ist die Verwendung von unterschiedlichen Zeichensätzen. Die semantische Heterogenität macht sich dadurch bemerkbar, dass Begrifflichkeiten in verschiedenen Systemen nicht die gleiche Bedeutung haben. Dies kann zu Synonymen und Homonymen führen. Synonyme können in der Form auftreten, dass unterschiedliche Ausprägungen die gleiche Bedeutung haben. So kann in der einen Datenquelle das Geschlecht für Personen die Ausprägungen „m“ oder „w“ haben und in einer anderen „männlich“ oder „weiblich“. Homonyme können durch gleiche Attributnamen mit unterschiedlicher Bedeutung in verschiedenen Datenquellen entstehen.

3.2.4 Integrationsbereich

Innerhalb des Integrationsbereiches werden die Daten aus allen Quellsystemen bereinigt, zusammengeführt und schließlich in der Basisdatenbank gespeichert. Für diese sollte ein weitestgehend normalisiertes Datenmodell genutzt werden.[26] Dementsprechend eignen sich relationale Datenbanken zur Umsetzung. Aus Sicht der späteren Auswertungen ist diese Basisdatenbank der einzige Datenlieferant und für die Quellsysteme das einzige gemeinsame Ziel. Da typischerweise mehrere Quellen und mehrere Auswertungen vorhanden sind, werden Architekturen mit solch einer zentralen Basisdatenbank auch als Hub-and-Spoke- bzw. Nabe-

[25] Vgl. Rossak 2013, S. 28.

[26] Vgl. Bauer / Günzel 2013, S. 199.

Speiche-Architektur bezeichnet.[27] Dabei entspricht die Basisdatenbank der Nabe und sowohl die Quellen als auch die Auswertungen den Speichen. Weitere Bezeichnungen für die Basisdatenbank sind auch Core (Data Warehouse) oder „Single Point of Truth“ (SPOT).[28] In der Vergangenheit war die Verwendung solch einer zentralen, normalisierten Komponente nicht ganz unumstritten. Inmon hatte bereits anfangs in seiner Architektur, die CIF (Corporate Information factory) genannt wird, solch eine Komponente fest verankert.[29] Dagegen wurde beispielsweise in Kimball´s Data-Mart-Bus-Architektur auf diese verzichtet und stattdessen die Daten direkt in ein dimensionales Datenmodell für Auswertungen geladen.[30] Auch in der Praxis sind Architekturen ohne eine zentrale Basisdatenbank zu finden.[31] Gründe sind insbesondere ein hoher Aufwand für die Erstellung und Wartung dieser. Dabei wird der Nutzen gerade von den Endbenutzern nicht immer direkt wahrgenommen. Hinzu kommt, dass die Daten einmal mehr kopiert werden müssen, was zu längeren Laufzeiten und einem höheren Speicherplatzbedarf führt. Es können aber durch den Nabe-Speiche-Ansatz auch Redundanzen in der Verarbeitung vermieden werden. Durch das normalisierte Datenmodell kann zudem die Datenqualität hinsichtlich der referentiellen Integrität gesichert werden. Ebenso kann durch die Eigenschaft als SPOT das Risiko von Inkonsistenten in den Auswertungen reduziert werden. Gerade im Data Warehouse bzw. in BI-Projekten ist die Qualität der Daten von besonderer Bedeutung, da diese einen wesentlichen Einfluss auf die Akzeptanz durch die Anwender hat.[32] Aus diesen Gründen wird die Basisdatenbank als eine essentielle Komponente in einem DWH gesehen. Auch Kimball lenkt inzwischen ein, dass solch eine Komponente eingesetzt werden kann und schlägt ebenfalls eine Architektur vor, die seine bisherige Architektur um eine zentrale Komponente erweitert.[33]

---

[27] Vgl. Kemper et al. 2010, S. 24.

[28] Vgl. Kemper et al. 2010, S. 227.

[29] Vgl. Inmon 2005, S. 16.

[30] Vgl. Kimball / Ross 2014, S. 19.

[31] Vgl. Bauer / Günzel 2013, S. 58.

[32] Vgl. Apel et al. 2015, S. 19.

[33] Vgl. Kimball / Ross, S. 29f.

Die Beladung der Basisdatenbank geschieht in den drei Schritten Extrahieren, Transformieren und Laden. Beim Extrahieren spielt der Monitor eine wichtige Rolle. Dessen Aufgabe ist es Datenveränderungen zu erkennen.[34] Im Idealfall kann der Monitor diese über entsprechende Loginformationen oder technische Zeitstempel ausfindig machen. Noch komfortabler ist es, wenn die geänderten Daten direkt aus gesonderten Tabellen abgegriffen werden können, in denen alle Änderungen durch Replikationsdienste gespeichert werden. Eine schlechtere Alternative zu den Replikationsdiensten sind Datenbanktrigger, da bei diesen die Gefahr besteht, dass sich die Performanz im operativen System verschlechtert. Im ungünstigsten Fall steht hingegen keine dieser Möglichkeiten zur Verfügung und die Daten müssen mit den bereits im DWH vorhandenen Daten abgeglichen werden. Unabhängig von der eingesetzten Monitortechnik befördert die Extraktionskomponente schließlich die Daten zunächst in einen Arbeitsbereich, der auch Stage genannt wird. Dort finden durch die Transformationskomponente verschiedene Datenqualitätsprüfungen, Bereinigungen, Anreicherungen und Harmonisierungen der Daten statt. Harmonisierungen sind für die Integration verschiedener Datenquellen unverzichtbar. Ein Aspekt ist dabei beispielsweise die Bereinigung von Dubletten, Homonymen und Synonymen. Beim dritten Schritt, dem Laden, gilt es die neuen Daten mit bereits in der Basisdatenbank vorhandenen Daten zu integrieren. Auch hierzu existieren hinsichtlich Ladestrategie und Historisierung der Daten verschiedene Ansätze. So können beispielsweise bei Änderungen die Daten überschrieben oder jeweils neue Datensätze mit einem entsprechenden Gültigkeitsmerkmal eingefügt werden.

Bei allen drei Schritten können Datenintegrationswerkzeuge zur Unterstützung eingesetzt werden. Zu den führenden Herstellern zählen dabei nach Gartners magischem Quadranten für Datenintegrationswerkzeuge aus dem Jahr 2014 Informatica, IBM, SAP, Oracle und SAS.[35] In Abhängigkeit der Reihenfolge der drei zuvor genannten Schritte lassen sich die beiden Ansätze ETL (Extrahieren, Transformieren, Laden) und ELT (Extrahieren, Laden, Transformieren) unterscheiden. Im Referenzmodell ist der klassische ETL-Ansatz zu sehen. Dabei werden die

---

[34] Vgl. Bauer / Günzel 2013, S. 54.

[35] Vgl. Beyer / Thoo 2014.

Daten zunächst in den Arbeitsbereich geladen, dort transformiert und letztendlich in die Basisdatenbank übertragen. Beim ELT-Ansatz geschieht die Transformation der Daten nicht im Arbeitsbereich, sondern beim Laden in die Zieldatenbank.[36] Dieser Ansatz ist insbesondere dann sinnvoll, wenn die Zieldatenbank sehr leistungsfähig ist.[37] Daher ist dieser Ansatz auch bei Datenintegrationswerkzeugen großer Datenbankhersteller wie z.B. Oracle beliebt.

3.2.5 Auswertebereich

Schließlich ist der letzte Bereich der Auswertebereich. Dieser wird auch als Mart bezeichnet und hat zum Ziel die Daten für konkrete Auswertungen aufzubereiten. Dementsprechend sollte das Datenmodell in erster Linie an den Auswertebedürfnissen der Anwender ausgerichtet sein.[38] Häufig wird daher nicht die dritte Normalform verwendet, da bei dieser die Abfrageperformanz bei Auswertungen zu schlecht und die Komplexität zu hoch werden.[39] Stattdessen sind (multi-)dimensionale Modellierungsansätze weit verbreitet.[40] In einer multidimensionalen Datenbank werden dabei OLAP-Würfel verwendet, die bereits im Abschnitt 2.3 angesprochen wurden. In einer relationalen Datenbank hat sich das Star-Schema als Standard etabliert.[41] Beide Ansätze haben gemeinsam, dass Kennzahlen als Fakten und Stammdaten als Dimensionen dargestellt werden. Im Star-Schema sind Fakten und Dimensionen Tabellen, die über Fremdschlüsselbeziehungen miteinander verbunden sind. Ein Beispiel für ein Star-Schema ist in Abbildung 4 zu sehen. Bei diesem ist in der Mitte eine Faktentabelle Umsatz, die von den drei Dimensionen Zeit, Region und Produkt umgeben ist. Wie in dem Beispiel zu sehen ist, sind die Dimensionen über 1:n-Beziehungen mit der Faktentabelle verbunden. Aus dieser sternförmigen Anordnung resultiert auch der Name Star-Schema.

---

[36] Vgl. Rossak 2013, S. 41.

[37] Vgl. Speare 2015.

[38] Vgl. Bauer / Günzel 2013, S. 64.

[39] Vgl. Kimball / Ross 2013, S. 8.

[40] Vgl. Kimball / Ross 2013, S. 7.

[41] Vgl. Hahne 2014, S. 94.

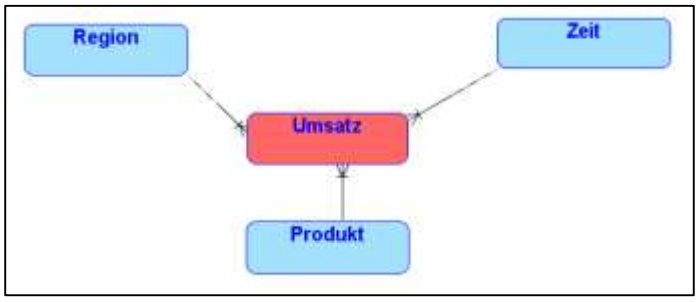

Abbildung 4: Beispiel Star-Schema

Dimensionen können auch Hierarchien sein, die mehrere Levels umfassen und die Daten auf unterschiedlichen Granularitäten beschreiben. Beispielsweise kann in einer Zeitdimension jeweils ein Level für Monat, Quartal und Jahr existieren. Im klassischen Star-Schema befinden sich alle Levels zu einer Dimension denormalisiert in einer einzigen Dimensionsstabelle. Werden solche hierarchischen Dimensionstabellen normalisiert und für jeden Level eine eigene Tabelle modelliert, führt das zu einer Abwandlung vom Star-Schema, die sich Snowflake-Schema nennt.[42] Dadurch können redundante Daten und somit auch Speicherplatz eingespart werden. Allerdings wird bei Abfragen eine größere Anzahl an Joins benötigt, was eine höhere Komplexität und schlechtere Performanz zur Folge hat. Ein konsequentes Snowflake-Schema wird daher selten eingesetzt. Jedoch können Mischformen, in denen nur bestimmte Dimensionen normalisiert werden, von beiden Ansätzen profitieren. So könnten insbesondere nur Level in separaten Tabellen gehalten werden, die selten abgefragt werden.

Im Referenzmodell sind für den Auswertebereich zwei Komponenten vorgesehen. Zum einen eine Ableitungsdatenbank und zum anderen eine Auswertungsdatenbank. Die damit verbundene Komponente „Auswertung“ kann im Referenzmodell als eine Schnittstelle zu entsprechenden Analyse- und Präsentations-Werkzeugen gesehen werden. Die Ableitungsdatenbank hat einen zentralen Charakter und die Auswertungsdatenbank einen dezentralen. Aus Sicht der Nabe-Speiche-Architektur verlängert erstere die Nabe und letztere entspricht den Speichen.[43] Ebenfalls sind auch wieder Transformations- und Ladekomponenten im Datenfluss zur Ableitungsdatenbank zu erkennen. Da die Datenqualität bereits durch die

---

[42] Vgl. Müller / Lenz 2013, S. 63.

[43] Vgl. Bauer / Günzel 2013, S. 66f.

Integrationsschicht sichergestellt wird, finden hier weniger Bereinigungen statt. Stattdessen wird die Struktur vom normalisierten Datenmodell in ein dimensionales Datenmodell überführt. Außerdem können weitere Transformationen an dieser Stelle Anreicherungen, wie z.B. Vorberechnungen von bestimmten Kennzahlen, oder Aggregationen sein. Allerdings wird empfohlen auch im dimensionalen Datenmodell die Daten auf der detailliertesten Aggregationsstufe vorzuhalten, um bei den eigentlichen Auswertungen mehr Flexibilität zu haben.[44]

In der Auswertungsdatenbank können die Daten individuell für bestimmte Abteilungen oder Nutzergruppen bereitgestellt werden. Diese dezentrale Aufbereitung kann sowohl aus organisatorischen als auch aus technischen Gründen zweckmäßig sein.[45] Aus organisatorischer Sicht erleichtert es beispielsweise die Vergabe von Berechtigungen. Ebenfalls kann es sinnvoll sein, wenn bestimmte Aufbereitungen nur für eine einzelne Auswertung/Abteilung angewendet werden sollen. Aus technischer Sicht kann eine dezentrale Lösung einfacher skaliert werden. Allerdings existieren auch hier diverse Ansätze, die von diesem Idealbild abweichen, um Aufwand und Laufzeiten zu reduzieren. So kann in einem kleinen Data Warehouse eine zentrale Mart-Komponente ausreichen. In einem anderen können rein dezentrale Data-Marts von Vorteil sein. Möglich sind auch Mischformen, welche nicht ganz so stringent wie die Referenzarchitektur sind. Dies bedeutet, dass sowohl Daten aus der zentralen Komponente direkt an die Auswertewerkzeuge übergeben werden als auch Daten aus der Basisdatenbank direkt in eine der dezentralen Komponenten geladen werden können.

3.3 Data Mining

Im vorangegangen Abschnitt wurde die Datenspeicherung und -aufbereitung innerhalb des Data-Warehouse-Systems betrachtet. An dieser Stelle wird das Data Mining als Möglichkeit zur Analyse- und Präsentation der bereitgestellten Daten betrachtet. Dazu werden in Abschnitt 3.3.1 zunächst weitere Analyse- und Präsentationsmöglichkeiten vorgestellt und vom Data Mining abgegrenzt. Abschnitt 3.3.2 zeigt typische Data-Mining-Problemstellungen und Abschnitt 3.3.3 Verfahren zu deren Lösung.

---

[44] Vgl. Ross / Kimball 2013, S. 9.

[45] Vgl. Bauer / Günzel 2013, S. 67.

3.3.1 Abgrenzung

Anwendungen zur Analyse und Präsentation der Daten aus dem Data Warehouse erweitern dieses zu einem BI-System (vgl. Abschnitt 3.1). Ziel ist es die Daten aus dem DWH zu visualisieren sowie Informationen und Wissen zu generieren. Dabei können verschiedene Klassen von Anwendungen hinsichtlich der Analyse-Möglichkeiten (Flexibilität) und der Komplexität unterschieden werden (Vgl. Abbildung 5).

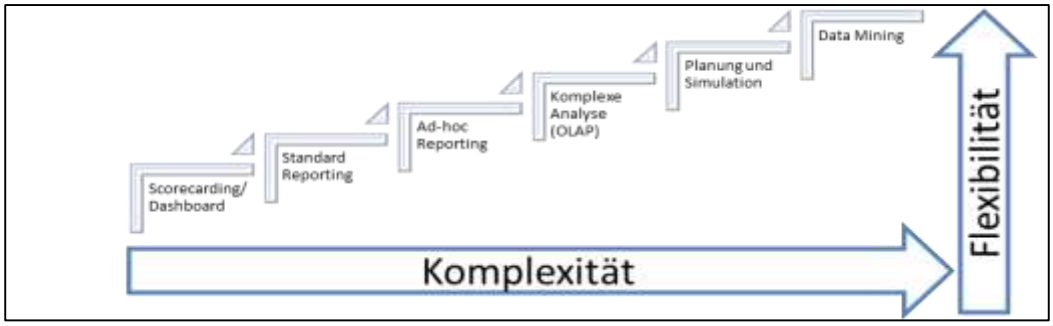

Abbildung 5: Klassen von Business-Intelligence-Anwendungen[46]

Die Ansätze auf der ersten Stufe **Scorecarding/Dashboard** zielen auf eine möglichst intuitive Bedienung und Erfassbarkeit der Informationen durch die Nutzer ab. In einem Dashboard, welches auch als Cockpit bezeichnet wird, wird dabei ein Überblick über meist aggregierte Kennzahlen gegeben. Typichweise werden dafür verschiedene Darstellungstypen miteinander kombiniert, wie in dem Beispiel in Abbildung 6 zu sehen ist.

Abbildung 6: Beispiel Dashboard[47]

Bevorzugt werden Dashboards in Web-Browsern angezeigt.[48] Etwas weitergehender ist der Ansatz der Scorecards, wie z.B. der Balanced Scorecard (BSC). Bei

[46] In Anlehnung an Bange 2010, S. 141.

[47] Proctor 2008.

dieser werden nicht nur einzelne aggregierte Kennzahlen, sondern auch die Ursache-Wirkungs-Beziehungen zwischen diesen Kennzahlen dargestellt.[49]

Die nächste Klasse ist das **Standard Reporting**. Dabei werden fertige Berichte erstellt und verteilt.[50] Ein häufiges Einsatzgebiet ist das betriebliche Berichtswesen. Im Standard Reporting geschieht die Erstellung und Verteilung der Berichte vorzugsweise durch Mitarbeiter der IT-Abteilung, da diese über die entsprechenden Fertigkeiten verfügen, um die Werkzeuge zur Berichterstellung zu bedienen.[51] Die Anwender benötigen daher keine besonderen Kenntnisse über das Werkzeug.

Dagegen erlaubt das **Ad-hoc Reporting** auch den Anwendern Berichte zu erstellen und zu modifizieren. Dafür brauchen diese entsprechende Kenntnisse oder Schulungen. Allerdings sind solche Werkzeuge darauf ausgelegt ohne tiefere technische Kenntnisse im Bereich der Datenbanken und zugehörigen Abfragesprachen (z.B. SQL) flexibel Berichte erstellen zu können.[52] Daher steht im Fokus, dass die Anwender schnell und einfach ihre eigenen Berichte zusammenstellen können und nicht ein hoher Funktionsumfang.[53]

Wird ein größerer Funktionsumfang benötigt, können **komplexere Analysen** mit Hilfe von OLAP-Operationen auf einem multidimensionalen Datenbestand durchgeführt werden. Dabei hat der Anwender die Möglichkeit durch aggregierte Datenbestände in verschiedene Richtungen intuitiv zu navigieren. Dort kann sowohl zwischen verschiedenen Aggregationsebenen gewechselt werden als auch innerhalb einer Ebene auf bestimmte Dimensionselemente gefiltert werden.[54] Solche Analysen zeichnen sich durch eine hohe Interaktion mit den Anwendern aus. Diese haben die Möglichkeit die Datenbestände explorativ zu untersuchen. Ebenso

---

[48] Vgl. Gluchowski 2010, S. 278f.

[49] Vgl. Bange 2010, S. 143.

[50] Vgl. Bauer / Günzel 2013, S. 77.

[51] Vgl. Bange 2010, S. 145.

[52] Vgl. Bange 2010, S. 146.

[53] Vgl. Bauer / Günzel 2013, S. 77.

[54] Vgl. Bauer / Günzel 2013, S. 75.

wird für diese Art von Analysen auch stärker die Nutzung von mathematischen/statistischen Methoden unterstützt.[55]

Im Bereich **Planung und Simulation** wird die besondere Anforderung an die Anwendung gestellt, dass es ermöglicht werden muss Daten in die Datenbank zurückzuschreiben.[56] Dies erlaubt es den Anwendern Vergleiche von Ergebnissen unterschiedlicher Szenarien durchzuführen und geplante Daten zu einem späteren Zeitpunkt oder in anderen Anwendungen weiter zu verwenden. Zur Durchführung der Planung oder Simulation stehen dem Anwender Möglichkeiten zur Verfügung, um entsprechende Parameter, Verteilungen, Regeln und auch konkrete Dateneingaben für eine Planung oder ein Szenario festzulegen. Um den hohen Anforderungen an die Performanz gerecht zu werden, kommt meistens ebenfalls eine multidimensionale Datenbank zum Einsatz.[57]

Zuletzt ist das **Data Mining** die komplexeste Klasse für Analysen im BI-Umfeld. Bei dieser ist das Ziel möglichst automatisiert mit Hilfe von Datenanalyse-Verfahren Muster in meist sehr großen und hochdimensionalen Datenbeständen aufzudecken.[58] Dabei existiert eine große Vielfalt an möglichen Problemstellungen und Verfahren. Verwandt zum Data Mining ist der Begriff KDD (Knowledge Discovery in Databases), der auch als Synonym zum Data Mining zu finden ist.[59] Allerdings handelt es sich dabei um einen Prozess, der neben dem Data-Mining auch Vor- und Nachbereitungen der Daten umfasst.[60]

3.3.2 Problemstellungen

Viele der typischen Problemstellungen lassen sich in die Gruppen Klassifikation, Regression, Bildung von Clustern, Erkennung von Abhängigkeiten (Assoziationsregeln) und Entdeckung von Abweichungen einordnen.[61] Die Aufgaben der letzten drei Gruppen zielen darauf ab tiefere Erkenntnisse und Beschreibungen zu den

---

[55] Vgl. Bange 2010, S. 147f.

[56] Vgl. Bange 2010, S. 151.

[57] Vgl. Bange 2010, S. 152.

[58] Vgl. Müller / Lenz 2013, S. 75.

[59] Vgl. Düsing 2010, S. 282.

[60] Vgl. Tan et al. 2014, S. 3.

[61] Vgl. Bauer / Günzel 2013, S. 131.

Daten zu erhalten.[62] Die ersten beiden Gruppen können darüber hinaus auch Aufgaben zur Vorhersage der Zukunft enthalten.

Bei der **Klassifikation** ist das Ziel Beobachtungen in bereits vorgegebene Kategorien einzuordnen. Ein Beispiel ist die Vorhersage, ob ein Kunde eines Mobilfunkanbieters ein potentieller Kündigungskandidat ist oder nicht. In dem Beispiel wird somit versucht einen Schluss für die Zukunft zu ziehen. Ein anderer Anwendungsfall zur Klassifikation kann die Spam-Erkennung von E-Mails sein. Bei der Klassifikation Spam-E-Mail oder nicht ist es nicht das Ziel die Zukunft vorherzusagen, sondern die E-Mail näher zu beschreiben.

Mit der **Regression** können ebenfalls Aussagen über die Zukunft gemacht werden. Der Unterschied zur Klassifikation liegt darin, dass bei der Regression kontinuierliche Werte vorhergesagt werden und keine diskreten, die bei der Klassifikation durch die Kategorien vorgegeben sind.[63] Beispiele können das Vorhersagen von Verkaufszahlen in einem Supermarkt oder des Verschleißes einer Maschine sein.

Bei der **Bildung von Clustern** werden die Beobachtungen ähnlich wie bei der Klassifikation verschiedenen Kategorien oder Gruppen zugeordnet. Allerdings sind diese Gruppen vorher noch nicht bekannt. Ziel bei der Gruppenbildung ist, dass Beobachtungen innerhalb einer Gruppe möglichst ähnlich sind, sich aber möglichst stark von anderen Gruppen unterscheiden.[64] Ein Beispiel ist die Segmentierung von Kunden für das Marketing.[65]

Bei der **Erkennung von Abhängigkeiten** werden Muster innerhalb von Beobachtungen gesucht. Genutzt wird dies zum Beispiel bei der Warenkorbanalyse. Mit dieser kann herausgefunden werden, welche Produkte besonders häufig in Kombination gekauft werden. Dabei konnten bereits auch unerwartete Kombinationen, wie Pampers und Bier, aufgedeckt werden.[66] Dies kann dann wiederum bei der Warenpräsentation berücksichtigt werden.

---

[62] Vgl. Tan et al. 2014, S. 7.

[63] Vgl. Tan et al. 2014, S. 8.

[64] Vgl. Tan et al. 2014, S. 9f.

[65] Vgl. Bauer / Günzel 2013, S. 131.

[66] Vgl. Aegerter 2014.

Die letzte Kategorie ist die **Aufdeckung von Ausreißern** und Anomalien. Hier wird gezielt nach Beobachtungen gesucht, die sich von der Mehrheit unterscheiden. Ein typisches Einsatzgebiet ist die Betrugserkennung. Am Beispiel der Betrugserkennung für Kreditkarten wird deutlich, dass neben der Erkennung von potentiellen Betrugsversuchen ebenfalls wichtig ist, dass nicht zu oft ein Fehlalarm auftritt.[67]

3.3.3 Verfahren

Zum Lösen der Aufgaben und Problemstellungen aus dem vorangegangen Abschnitt nutzt das Data Mining Ansätze aus vielen anderen Gebieten (z.B. Statistik, Künstliche Intelligenz oder maschinelles Lernen).[68] Daraus ergibt sich eine Vielzahl an Verfahren, die genutzt werden können. Im Folgenden werden zwei dieser Ansätze exemplarisch näher beschrieben. Dies sind zum einen Entscheidungsbäume, die bei Klassifikations- und Regressionsaufgaben eingesetzt werden können und zum anderen der k-means-Algorithmus, der bei der Bildung von Clustern genutzt wird.[69]

Entscheidungsbäume zählen zu den Verfahren des überwachten maschinellen Lernens. Bei diesen muss im Vorhinein eine Datenbasis mit bereits bekannten Ergebnissen existieren. Diese wird in Trainings- und Testdaten aufgeteilt. Aus den Trainingsdaten leitet der Algorithmus Entscheidungsregeln ab, die auf andere Datenbestände angewendet werden können. In Abbildung 7 ist ein möglicher Entscheidungsbaum zu sehen, ob ein Mobilfunkkunde ein potentieller Kündigungskandidat ist oder nicht.

---

[67] Vgl. Tan et al. 2014, S. 11.

[68] Vgl. Tan et al. 2014, S. 6.

[69] Vgl. Chamoni et al. 2010, S. 333.

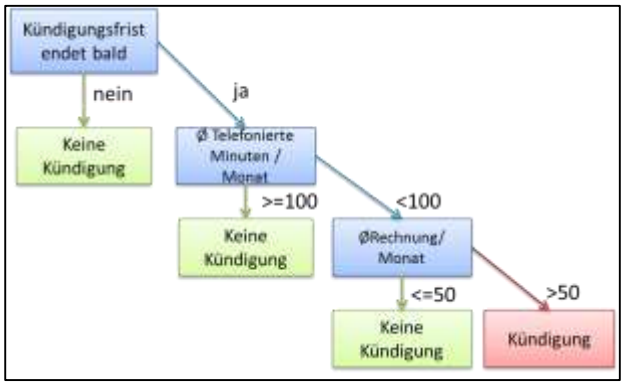

Abbildung 7: Beispiel Entscheidungsbaum

Mit den Testdaten können diese evaluiert werden. Die Anwendung eines Entscheidungsbaumes gestaltet sich dabei als einfacher als der Aufbau.

Für die Prüfung von einem weiteren Kunden muss lediglich der entsprechende Pfad im Baum entlang gegangen werden. An den inneren Knoten und dem Wurzelknoten werden verschiedene Testbedingungen zu den Attributen des Kunden geprüft. Die Kanten stellen die möglichen Optionen dar. Die Blätter zeigen schließlich die Klasse, in die der Kunde eingeordnet wird (Kündigung oder keine Kündigung). Zum Aufbau solch eines Baumes existieren verschiedene Algorithmen, die die unterschiedlichsten Entscheidungsbäume aus einem Trainingsdatenbestand erzeugen können.[70] Viele dieser Entscheidungsbaumalgorithmen basieren auf dem sogenannten Hunt´s Algorithmus.[71] Bei diesem handelt es sich um einen rekursiven Algorithmus, der den Trainingsdatenbestand so oft weiterzerlegt bis jeder Teildatenbestand eindeutig einer Klasse zugeordnet werden kann. Herausforderungen liegen dabei zum einen in der Wahl der geeigneten Testbedingungen, an denen die Datenmengen zerlegt werden, und zum anderen in einer Vermeidung einer Überanpassung an die Trainingsdaten.[72]

Clusterverfahren gehören im Gegensatz zu den Entscheidungsbäumen zu den unüberwachten Algorithmen des maschinellen Lernens.[73] Somit liegen keine Erwartungen an die Ergebnisse des Verfahrens in Form von Test- oder Trainingsdaten vor. Es gibt verschiedene Arten von Clusterverfahren. Häufig wird dabei zwi-

[70] Vgl. Tan et al. 2014, S. 151.

[71] Vgl. Tan et al. 2014, S. 152.

[72] Vgl. Tan et al. 2014, S. 155.

[73] Vgl. Chamoni et al. 2010, S. 343.

schen hierarchischem und partitionierendem Clustering unterschieden. Beim hierarchischen Clustering wird eine Baumstruktur an Clustern aufgebaut. Die Blätter entsprechen dabei den einzelnen Datenobjekten bzw. Beobachtungen. In den Knoten werden diese zu Clustern zusammengefasst. Dies kann auf mehreren Levels geschehen. So können Cluster einer Ebene zu einem größeren Cluster in der übergeordneten Ebene vereint werden. Der Wurzelknoten enthält schließlich alle Datenobjekte.[74] Beim partitionierenden Clustering ist dagegen jedes Datenobjekt genau einem Cluster zugeordnet. Zwischen den Clustern besteht keine Verbindung. Ein Beispiel für ein partitionierendes Clusterverfahren ist k-means. Anhand dessen wird im Folgenden der wesentliche Ansatz solcher Verfahren gezeigt. Bei diesem Verfahren gilt es k Cluster zu bilden. In jedem dieser Cluster wird ein Datenobjekt als Mittelpunkt festgelegt, der dieses repräsentiert. Dieser wird auch als Centroid bezeichnet.[75] Ziel ist es, dass alle Datenobjekte eines Clusters zu dem Centroiden ihres Clusters mehr Ähnlichkeit haben als zu allen anderen Centroiden. Der eigentliche Algorithmus ist in Abbildung 8 dargestellt.

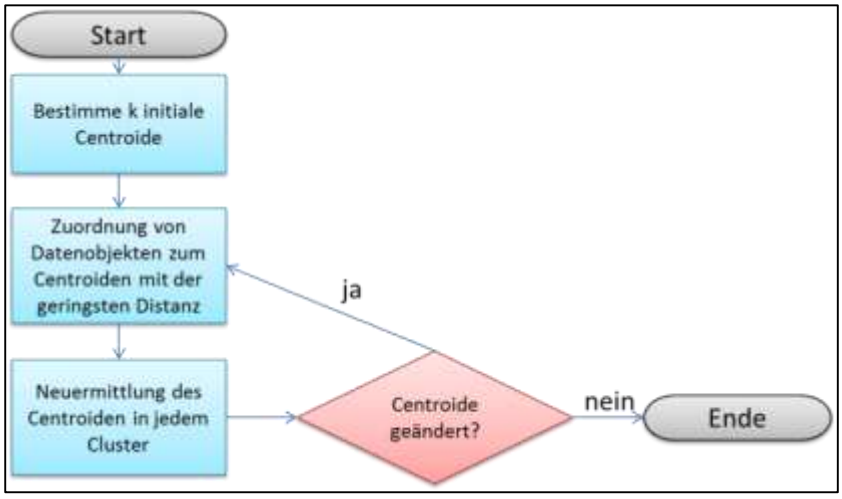

Abbildung 8: k-means Algorithmus

Zunächst werden k Centroide initiiert. Danach folgen zwei sich abwechselnd wiederholende Schritte. Im ersten werden die restlichen Datenobjekte den Centroiden zugeordnet. Dabei wird über ein Ähnlichkeits- oder Distanzmaß ermittelt zu welchem Centroiden die größte Ähnlichkeit bzw. geringste Distanz besteht. Da durch das Hinzufügen der Datenobjekte die ursprünglichen Centroiden womöglich nicht

[74] Vgl. Müller / Lenz 2013, S. 88.

[75] Vgl. Müller / Lenz 2013, S. 86f.

mehr die Mittelpunkte in ihrem Cluster bilden, müssen im zweiten Schritt diese neu bestimmt werden. Danach können wieder die übrigen Datenobjekte den Centroiden neu zugeordnet werden. Der Algorithmus ist beendet, wenn sich die Centroiden nicht mehr ändern.

4 Big Data

Dieses Kapitel diskutiert Ansätze aus dem Bereich Big Data. In Abschnitt 4.1 wird dazu zunächst anhand der Herausforderungen, die mit Big Data einhergehen, diskutiert, was Big Data ist. Einen Überblick zu den Datenverarbeitungsansätzen im Umfeld von Big Data wird in Abschnitt 4.2 vorgestellt. Typische Datenquellen, die durch Big Data an Popularität gewonnen haben, werden in Abschnitt 4.3 vorgestellt. Durch den Fokus dieser Arbeit werden dabei insbesondere soziale Netzwerke betrachtet. Analyse-Ansätze für Daten aus solch sozialen Netzwerken sind Bestandteil von Abschnitt 4.4. Schließlich wird in Abschnitt 4.5 diskutiert wie eine Architektur aussehen kann, um das klassische Data Warehouse aus Kapitel 3 mit Big-Data-Ansätzen zu kombinieren.

4.1 Herausforderungen

Big Data ist ein Begriff, der sehr vielseitig und auch unterschiedlich genutzt wird. Eine einheitliche Definition liegt nicht vor. Feststeht allerdings, dass Big Data mehr als die Verarbeitung von großen Datenmengen ist. Das Datenvolumen ist nur eine der Herausforderungen von Big Data, aber es gehören noch weitere dazu. Häufig werden diese anhand von Begriffen, die mit dem Buchstaben V beginnen, umschrieben. Den Ursprung bilden dabei die drei durch Gartner definierten V´s Volume, Variety und Velocity.[76]

Volume steht dabei für das Datenvolumen. Günstiger Speicher ermöglicht es mehr Daten vorzuhalten. Gleichzeitig wächst das Datenvolumen rapide. Eine Studie von IDC (International Data Corporation) zeigt, dass sich das weltweite Datenvolumen alle zwei Jahre verdoppeln wird.[77] Jedoch muss dieses größere Datenvolumen nicht nur gespeichert, sondern auch verarbeitet und ausgewertet werden. Dabei können größere Datenmengen die Analyseergebnisse verbessern. Auch wo in der Vergangenheit möglicherweise nur eine Stichprobe oder ein Ausschnitt der Daten ausgewertet werden konnte, sollen durch Big Data Möglichkeiten zur Verfügung stehen, die Gesamtheit der Daten zu betrachten.

---

[76] Vgl. Frampton 2015, S. 1.

[77] Vgl. IDC 2014.

Mit **Variety** wird Bezug auf die Vielfalt der Daten genommen, die ausgewertet werden sollen. In einem klassischen Data Warehouse sind es typischerweise strukturierte Daten. Jedoch stehen Unternehmen auch eine Vielzahl an semi- und unstrukturierten Daten zur Verfügung (Texte aus E-Mails, Foren, Blogs, Pressemitteilungen, etc.), die eine hohe Relevanz haben.[78] Um diese auszuwerten zu können, reichen die klassischen Ansätze nicht mehr aus.

Velocity drückt aus, dass auch durch die beiden zuvor genannten Punkte immer mehr Daten immer schneller in die Unternehmen fließen und ein Bedarf besteht, diese ebenfalls möglichst schnell zu verarbeiten.[79] So müssen aus Daten in Echtzeit Informationen generiert werden, um die Entscheidungsfindung zu unterstützen.

Neben diesen drei gängigen V´s sind auch noch weitere „V"-Charakteristika bzw. Herausforderungen für Big Data in der Literatur und in der Praxis zu finden. Zu diesen zählen Value, Veracity, Variability/Variance, Viability.[80] **Value** beschreibt dabei die Nutzensicht. So sollen mit Big Data neue Mehrwerte bis hin zu komplett neuen Geschäftsmodellen für die Unternehmen ermöglicht werden. Insbesondere durch IBM wird die Eigenschaft **Veracity** hervorgehoben, die die Vertrauenswürdigkeit bzw. Richtigkeit der Daten betrifft. Durch die Hinzunahme unsicherer Datenquellen (z.B. Soziale Netzwerke, öffentliche Blogs) besteht ein erhöhtes Risiko von falschen Daten. **Variability** bzw. **Variance** beziehen sich ebenfalls auf die Qualität der Daten. Diese betreffen allerdings weniger die Eingangsdaten als die Ergebnisse. So existieren sowohl in der Analyse als auch bei der Interpretation der Resultate variable Bestandteile, die das Ergebnis beeinflussen können.[81] **Viability** berücksichtigt die Realisierbarkeit des Vorhabens mit den Daten. So soll einfach und günstig prüfbar sein, ob bestimmte Daten relevant für einen Sachverhalt sind, bevor diese aufwändig gesammelt und verarbeitet werden.[82]

---

[78] Vgl. Rath 2014.

[79] Vgl. Freiknecht 2014, S. 11.

[80] Vgl. Grimes 2013.

[81] Vgl. Hopkins / Evelson 2011, S. 3ff.

[82] Vgl. Biehn 2013.

Zusammenfassend kann somit für Big Data festgehalten werden, dass dies Daten sind, die aufgrund ihres Volumens, ihrer Komplexität und der Entstehungsgeschwindigkeit nicht mehr mit traditionellen Technologen verarbeitet und ausgewertet werden können.[83]

4.2 Datenspeicher

In diesem Abschnitt werden alternative Ansätze zur relationalen Datenbank beschrieben, die bei Bewältigung der durch Big Data entstanden Herausforderungen bei der Datenspeicherung und -verarbeitung unterstützen können. Dies geschieht anhand von NoSQL (Not only SQL) in Abschnitt 4.2.1 und Hadoop in Abschnitt 4.2.2.

4.2.1 NoSQL-Datenbanken

Unter dem Begriff NoSQL sind einige nicht-relationale Datenbanktechnologien entstanden, die bei der Bewältigung von speziellen Herausforderungen auch im Big-Data-Umfeld unterstützen können. In diesem Abschnitt werden zunächst einige Konzepte dieser Technologien vorgestellt (vgl. Abschnitt 4.2.1.1) und im Anschluss ein Überblick über mögliche Typen (vgl. Abschnitt 4.2.1.2) gegeben.

4.2.1.1 Konzepte

NoSQL-Datenbanken können sehr vielseitig sein und für unterschiedlichste Anwendungsfälle entwickelt werden. Dennoch existieren einige Konzepte und typische Eigenschaften, die für viele NoSQL-Datenbanken gelten und diese auch von relationalen Datenbanken abgrenzen.

Einer der wesentlichen Unterschiede ist dabei, dass NoSQL-Datenbanken jeweils **Spezialisten** für eine ziemlich konkrete Art von Anwendungsfällen sind. Im Vergleich werden relationale Datenbanken eher als Generalisten für eine Vielzahl von Problemstellungen eingesetzt. Allerdings sind diese nicht immer die beste Möglichkeit, da das Problem und auch die Daten zunächst in eine strukturierte relationale Form gebracht werden müssen, was nicht immer möglich ist. NoSQL-Datenbanken erlauben dagegen auch andere Strukturen. Sie werden auch als schemafrei bezeichnet.[84] Allerdings darf dieser Begriff nicht falsch verstanden

[83] Vgl. Krishnan 2013, S. 5.

[84] Vgl. Freiknecht 2014, S. 189.

werden. Auch in den meisten NoSQL-Datenbanken existieren Schemata und Strukturen zum Ablegen und Abfragen von Daten. Allerdings sind diese flexibler als bei relationalen Datenbanken.

Bezüglich der **Hardware** sind in NoSQL-Datenbanken zwei wesentliche Eigenschaften zu erkennen. Zum einen wird meist eine horizontale Skalierbarkeit (scale-out) verwendet, bei der das Gesamtsystem über die Hinzunahme weiterer Knoten skaliert werden kann.[85] Dem gegenüber steht die vertikale Skalierung (scale-up), die häufig in relationalen Datenbankmanagementsystemen Anwendung findet und durch die Aufwertung der Hardware skaliert.[86] Zum anderen geschieht diese horizontale Skalierung typischerweise durch günstigere Standardhardware.[87]

Eine weitere Eigenschaft von NoSQL-Datenbanken, welche den Einsatz vergleichsweise kostengünstig macht, ist **Open Source**. Gerade durch Unternehmen wie Google, Yahoo oder auch einige soziale Netzwerke (z.B. Facebook) wurden viele Ansätze in Form von Open Source Software veröffentlicht und haben somit ein rasantes Wachstum dieser Technologien vorangetrieben.[88]

Die Eigenschaften zum **Transaktionsverhalten** werden in relationalen Datenbanken typischerweise mit dem Akronym ACID (Atomicity, Consistency, Isolation, Durability) umschrieben.[89] Somit ist dort eine Transaktion atomar, hinterlässt einen konsistenten Datenstand, kann nicht durch parallele Transaktionen beeinflusst werden und hat eine dauerhafte Wirkung. In NoSQL-Datenbanken kommt dagegen das CAP-Theorem zum Einsatz (vgl. Abbildung 9). Dieses besagt, dass in einem verteilten System die drei Eigenschaften Konsistenz, Verfügbarkeit und Partitionstoleranz konkurrierend zueinander stehen.[90] Es kann somit nicht gleichzeitig sichergestellt werden, dass alle Anfragen (performant) beantwortet werden, alle Replikate ständig konsistent sind und das System auch bei Ausfällen einzelner Knoten fehlerfrei weiterarbeiten kann.

---

[85] Vgl. Mohanty et al. 2013, S. 75f.

[86] Vgl. Baron 2013, S. 161.

[87] Vgl. Mohanty et al. 2013, S. 93.

[88] Vgl. Baron 2013, S. 88.

[89] Vgl. Kempler / Eickler 2013, S. 299.

[90] Vgl. Zerbes 2014.

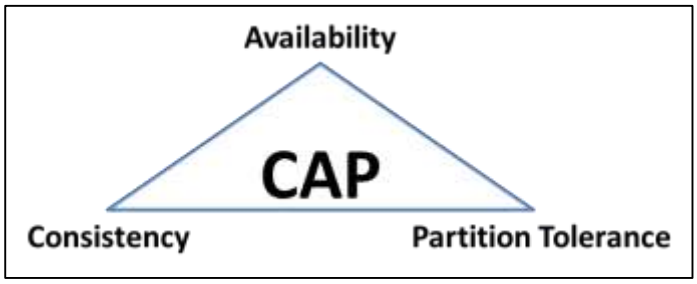

Abbildung 9: CAP-Theorem

Konkrete Anwendung findet das CAP-Theorem beispielsweise durch das BASE-Prinzip (Basically Available, Soft state, Eventual consistency). Bei diesem wird keine Konsistenz garantiert.[91]

Zuletzt soll noch das **MapReduce**-Programmiermodell als ein Konzept zur Datenverarbeitung betrachtet wurden. Es wurde von Google Inc. zur effizienten Websuche entwickelt und im Jahr 2004 veröffentlicht.[92] Sein Ziel ist es die Entwicklung von Anwendungen mit parallelen und hochskalierbaren Datenverarbeitungen zu unterstützen.[93] Dabei dient es zur Batchverarbeitung in verteilten Systemen. Den Entwickler unterstützt es, indem es Aufgaben zur Parallelisierung, Fehlertoleranz und Datenverteilung übernimmt.[94] Zu Entwickeln ist dann lediglich die eigentliche Berechnungslogik in zwei Funktionen - Map und Reduce. Den Ablauf solch eines Programmes zeigt Abbildung 10 am Beispiel der Berechnung der Durchschnittsnote von Abschlussarbeiten von Studenten.

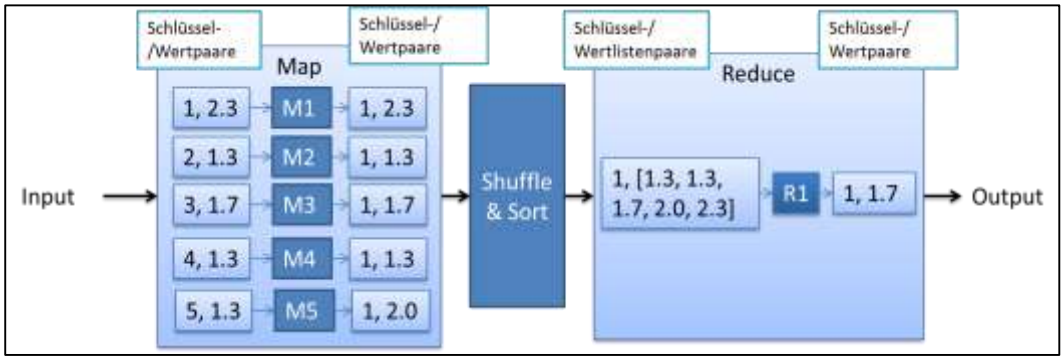

Abbildung 10: MapReduce-Programmiermodell[95]

---

[91] Vgl. Zerbes 2014.

[92] Vgl. Dean / Ghemawat 2004, S. 10f.

[93] Vgl. White 2015, S. 19.

[94] Vgl. Dean / Ghemawat 2004, S. 1.

[95] In Anlehnung an White 2015, S. 24.

Die Map-Komponente, die aus mehreren parallelen Prozessen bestehen kann, erhält die Eingangsdaten als Schlüssel-Wertpaare. In diesem Beispiel ist dies eine aufsteigende Nummer als Schlüssel für den jeweiligen Studenten und die Note als Wert. Innerhalb der Map-Komponenten können Schlüssel-Wert-Paare herausgefiltert, transformiert oder in mehrere neue Schlüssel-Wert-Paare zerlegt werden. Im Beispiel erhalten die Noten einen neuen Schlüssel mit dem fixen Wert von 1. Der Reducer nimmt die Ergebnisse auf und verdichtet es zum Endergebnis, welches ebenfalls als Schlüssel-Wertpaar ausgegeben wird. Analog zu den Map-Prozessen besteht die Möglichkeit, dass mehrere Reducer-Prozesse parallel arbeiten. Allerdings sind es typischerweise weniger als die Map-Prozesse. Ist die Aufgabenstellung beispielsweise eine Aggregation wird es maximal nur so viele Reducer-Prozesse geben, wie das Gruppierungskriterium zulässt. Da im aktuellen Beispiel gar keine Gruppierungsregel vorliegt, sondern die Durchschnittsnote über alle Noten gebildet werden soll, existiert dort nur ein Reducer-Prozess. Zuletzt ist die Komponente Shuffle & Sort zu erwähnen, die sich in der Mitte befindet. Diese sortiert die Daten nach dem Schlüssel und fast die Schlüssel-Wertpaare zu Schlüssel-Wertlistenpaaren zusammen, in denen jeder Schlüssel nur noch einmal vorkommt.

4.2.1.2 Typen

Da sehr unterschiedliche NoSQL-Datenbanken existieren, ist eine Kategorisierung nach Typen sinnvoll, die sich zum einen in der Art der Daten, die sie vorhalten, unterscheiden und zum anderen in der Struktur, in der die Daten logisch und physisch gespeichert werden.

Der erste Vertreter sind **Schlüssel-Wert-Datenbanken**, deren Datenmodell bereits in Abschnitt 2.3 betrachtet wurde. Es existiert ein eindeutiger Schlüssel, der einen bestimmten Wert identifiziert. Somit können diese mit einer Hashtabelle verglichen werden.[96] Dabei müssen sowohl Schlüssel als auch Wert nicht zwangsweise atomar sein. Ein Vertreter von Schlüssel-Wert-Datenbanken ist Amazon Dynamo.

Bei **Spaltenorientierten Datenbanken** geschieht die physische Speicherung neuer Einträge nicht zeilenweise, sondern spaltenweise. Dies bringt im Vergleich zu

[96] Vgl. Freiknecht 2014, S. 191.

relationalen Datenbanken Vorteile für Auswertungen auf Spaltenlevel (z.B. Aggregationen).[97] Soll umgekehrt jedoch ein komplettes Tupel abgefragt werden, ist der Ansatz von Nachteil. Eine Spalte kann über einen Schlüssel identifiziert werden und besteht typischerweise aus drei Teilen, dem Namen, dem Wert und einem Zeitstempel. Mehrere Spalten, die häufig gemeinsam verwendet werden, können auch zu einer Spaltenfamilie zusammengefasst werden.[98] Beispiele sind SAP Hana oder Cassandra.

Dokumentenorientierte Datenbanken können als Sonderform der Schlüssel-Wert-Datenbanken betrachtet werden, die sich darauf spezialisiert haben, dass die Werte komplette Dokumente (z.B. Office-Dateien) sind.[99] Beispiele sind Mongo DB und Couch DB.

In-Memory-Datenbank halten die gesamten Daten während des Betriebes flüchtig im Arbeitsspeicher und nicht auf der Festplatte. Dies ermöglicht signifikant schnellere Zugriffszeiten. Dafür ist der Preis von flüchtigem Arbeitsspeicher teurer als für Festplattenspeicher. Ein weiterer Nachteil des flüchtigen Speicherns ist das Risiko von Datenverlusten bei einem Systemabsturz.[100] Beispiele sind Oracle Times Ten oder SAP Hana.

Zuletzt seien an dieser Stelle noch die **Graphen-Datenbanken** erwähnt. Diese eignen sich zur Abbildung von komplexen Beziehungen in Baumstrukturen. Somit können diese beispielsweise auch besonders gut zum Abbilden von Beziehungen zwischen Nutzern in einem sozialen Netzwerk genutzt werden.[101] Ein Beispiel ist Neo4j.

4.2.2 Hadoop

Hadoop ist ein Open Source Rahmenwerk der Apache Software Foundation für sichere, skalierbare und verteilte Berechnungen.[102] Der Kern von Hadoop wird durch das Hadoop-Distributed-Filesystem (HDFS), dem MapReduce-Algorithmus

---

[97] Vgl. Freiknecht 2014, S. 192.

[98] Vgl. Krishnan 2013, S. 89.

[99] Vgl. Freiknecht 2014, S. 192.

[100] Vgl. Freiknecht 2014, S. 192.

[101] Vgl. Krishnan 2013, S. 97.

[102] Vgl. Apache Hadoop 2015.

und seit Hadoop 2 durch YARN (Yet Another Resource Negotiator) gebildet. Um diesen Kern herum hat sich eine Vielzahl an Projekten zur Entwicklung von Werkzeugen mit speziellen Funktionen angesammelt.[103] Abbildung 11 zeigt eine Übersicht der wesentlichen Projekte. Diese werden in den folgenden Abschnitten vorgestellt. Dabei werden in den Abschnitten 4.2.2.1 und 4.2.2.2 die Kernkomponenten ausführlich erläutert und in Abschnitt 4.2.2.3 ein Überblick zu den übrigen Komponenten gegeben. Da es insbesondere bei den Kernkomponenten einige Veränderungen zwischen Hadoop Version 1 und Hadoop Version 2 gibt, wird in den nachfolgenden Beschreibungen bei Unterschieden die Versionsnummer V1 bzw. V2 mit angegeben. Ist keine Versionsnummer angegeben beziehen sich die Ausführungen auf beide Versionen.

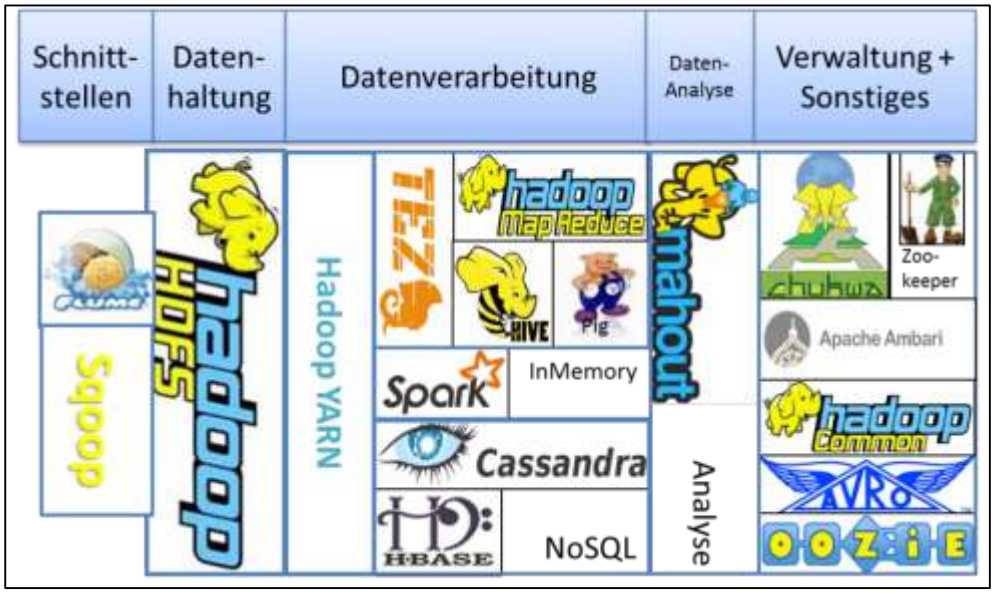

Abbildung 11: Hadoop Ökosystem

4.2.2.1 Hadoop Distributed Filesystem

Zur Datenhaltung nutzt Hadoop das HDFS, welches seinen Ursprung in den Konzepten vom Google-Filesystem (GFS) hat.[104] Wesentliche Eigenschaften dieses Dateisystems sind eine gute Skalierbarkeit, verteilte Datenhaltung, hohe Verfügbarkeit und effiziente Speicherung großer Dateien. Die gute Skalierbarkeit wird durch eine horizontale Skalierung auf Standardhardware erreicht. Zur Sicherstellung der Verfügbarkeit der Daten werden diese redundant auf unterschiedlichen

[103] Vgl. Prajapati 2013, S. 6f.

[104] Vgl. Prajapati 2013, S. 28.

Knoten vorgehalten. Dabei werden alle Daten standardmäßig dreifach repliziert. Beim Ausfall eines Knotens können die Daten von einem der anderen beiden Knoten verwendet werden. Durch eine standardmäßige Blockgröße von 64 Megabyte (MB), die noch vergrößert werden kann, können große Dateien effizient gespeichert werden. Im Vergleich sind in gängigen Dateisystemen Blockgrößen in Bereichen von wenigen Kilobyte (KB) üblich.

Im Betrieb vom HDFS sind drei Komponenten hervorzuheben und zwar der Datanode, Namenode und Secondary Namenode. Vom **Datanode** existieren typischerweise mehrere. Auf diesen werden die eigentlichen Daten in Blöcken abgelegt.[105] Vom **Namenode** ist dagegen in Hadoop V1 immer genau einer vorhanden. Dieser stellt Management- und Kontrollservices zur Verwaltung des Dateisystems bereit. Dazu gehört insbesondere auch die Vorhaltung von zugehörigen Metadaten. So speichert dieser genau ab, welche Dateien bzw. Blöcke auf welchem Datanode liegen. Diese Informationen werden ihm aktiv durch die Datanodes in regelmäßigen Abständen mitgeteilt. Auf diese Weise erkennt der Namenode ebenfalls, wenn ein Datanode ausgefallen ist. Datenabfragen kann er in solch einem Fall an einen der anderen Datanodes weiterleiten, auf dem ein entsprechendes Replikat zur Verfügung steht. So hat ein Ausfall von einem Datanode typischerweise keine Auswirkungen auf den Client. Problematischer dagegen ist ein Ausfall vom Namenode selber, da dieser der einzige Prozess ist, der weiß welche Daten sich auf welchem Datanode befinden. Ohne diesen können die Anfragen des Clients nicht verarbeitet werden. Daher wird dieser auch SPOF (Single Point of Failure) genannt.[106] Um das Risiko eines Ausfalls zu reduzieren, ist für diesen auch der Einsatz hochwertigerer Hardware sinnvoll. Der Schaden bei einem Ausfall kann des Weiteren durch einen **Seconday Namenode** reduziert werden. Dabei handelt es sich nicht um einen Standby-Namenode, der bei einem Ausfall automatisch einspringt. Dennoch verfügt dieser aufgrund seiner eigentlichen Aufgabe immer über einen nahezu aktuellen Stand der Metadaten, den er im Notfall einem neuen Namenode bereitstellen kann.[107] Seine primäre Aufgabe besteht jedoch darin, den Namenode in regelmäßigen Abständen beim Aufräumen der Metadaten

[105] Vgl. Venner 2009, S. 6.

[106] Vgl. White 2015, S. 48.

[107] Vgl. White 2015, S. 47.

zu unterstützen. Denn dieser vereint im laufenden Betrieb die von den Datanodes gelieferten Daten nicht mit den vorhandenen Metadaten. Stattdessen werden diese nur in einem flüchtigen Speicher vorgehalten und erst beim Hoch- und Runterfahren in den persistenten Metadaten integriert und in einem neuen Snapshot abgelegt. Bei einer langen Betriebszeit kann diese flüchtige Datenmenge sehr groß werden. Um einen Überlauf zu vermeiden, werden diese regelmäßig zum Seconday Namenode kopiert und dort in einem neuen Snapshot aufbereitet. Anschließend wird dieser zum Namenode zurückgeschickt.

Das Zusammenspiel dieser drei Komponenten bei einem Schreibvorgang im HDFS zeigt Abbildung 12. Der Client möchte die Datei „eineDatei.txt“ auf dem HDFS ablegen. Die Datei wird in Blöcke aufgeteilt, die direkt zu einem Datanode übertragen und von dort aus auf zwei weitere repliziert werden. Es ist zu erkennen, dass kein Datenfluss über den Namenode stattfindet. Zum Namenode werden nur die zugehörigen Metadaten übertragen, die er sich vom Secondary Namenode aufräumen lassen kann.

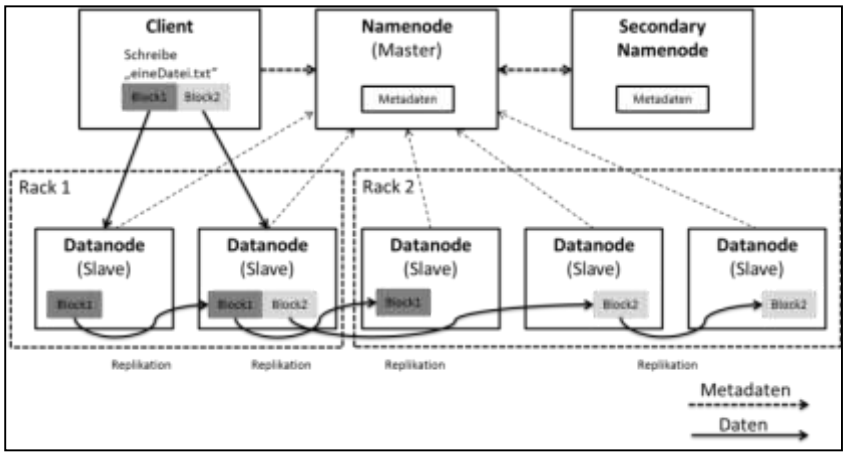

Abbildung 12: Hadoop-Distributed-Filesystem[108]

In Hadoop V2 sind zwei wesentliche Änderungen für das HDFS. Zum einen ist nun die Möglichkeit gegeben, den Namenode horizontal zu skalieren.[109] Vorher konnte bei besonders großen Clustern dieser zum Flaschenhals werden. Zum anderen wird der Eigenschaft vom Namenode als SPOF zu gelten entgegengewirkt. Es besteht die Möglichkeit eine höhere Verfügbarkeit durch die Hinzunahme von

[108] Frisch 2014, S. 4.

[109] Vgl. White 2015, S. 48.

einem Standby Namenode zu erreichen. Dieser kann bei einem Ausfall ohne größere Verzögerungen die Aufgaben vom eigentlichen Namenode fortführen.[110]

4.2.2.2 Von MapReduce zu YARN

Hinsichtlich der Datenverarbeitung ist durch den Übergang von Hadoop V1 zu Hadoop V2 die neue Komponente YARN hinzugekommen.

In Hadoop V1 ist **MapReduce** das gesetzte Programmiermodell zur Datenverarbeitung. Dabei wird die Verarbeitung analog zum HDFS durch einen Master- und einen oder mehreren Slave-Prozessen durchgeführt. Das Pendant zum Namenode ist der Jobtracker und zum Datanode der Tasktracker. Somit werden Steuer- und Verwaltungsaufgaben vom Jobtracker übernommen. Dieser nimmt den Job vom Client entgegen, verteilt die Aufgaben an die Tasktracker und kontrolliert bzw. überwacht die Ausführung.[111] Die Tasktracker führen die eigentlichen Map- oder Reduce-Aufgaben aus. In einem HDFS ist es üblich, dass ein Datanode ebenfalls Tasktracker-Services bereitstellt und der Namenode entsprechend Jobtracker-Services.[112] Dadurch können die Daten dort verarbeitet werden, wo sie sich bereits befinden, was den Netzwerkverkehr reduziert.

YARN bietet in Hadoop V2 einige grundlegende Änderungen, die auch als MapReduce 2 oder Next Generation MapReduce bezeichnet werden. Hadoop V1 ist durch die Fixierung auf MapReduce auf eine Batch-Verarbeitung begrenzt und mit dem Jobtracker als alleinigen zentralen Knoten zur Verwaltung und Überwachung aller Verarbeitungen auch in der Skalierbarkeit selbst eingeschränkt.[113] Dem wirken insbesondere zwei Aspekte von YARN entgegen. Zum einen werden die Komponenten Job- und Tasktracker durch den Ressource Manager, dem Node Manager und dem Application Master ersetzt (vgl. Abbildung 13). Dabei werden die beiden primären Aufgaben des Jobtrackers, Koordination von Jobs und Ausführungsüberwachung, neu verteilt. Der Ressource Manager übernimmt die Koordination aller Jobs im Sinne einer globalen und zentralen Ressourcenverteilung. Lokal befindet sich auf jedem Knoten ein Node Manager. Dieser überwacht den

---

[110] Vgl. White 2015, S. 49.

[111] Vgl. Venner 2009, S. 5.

[112] Vgl. Venner 2009, S. 6.

[113] Vgl. Frampton 2015, S. 13.

Status des zugehörigen Knotens und teilt diesen regelmäßig dem Ressource Manager mit. Die Ressourcen je Knoten werden als logische Pakete, die als Container bezeichnet werden, angeboten.[114] Ein besonderer Container ist der Application Master, der auch als Container 0 bezeichnet wird.[115] Dieser existiert einmal je Applikation und überwacht die Ausführung dieser.[116] Typischerweise verfügt er alleine nicht über ausreichend Ressourcen, um die gesamte Applikation auszuführen. Daher ist eine seiner wesentlichen Aufgaben die Anfrage bzw. das Aushandeln von zusätzlichen Ressourcen beim Ressource Manager.

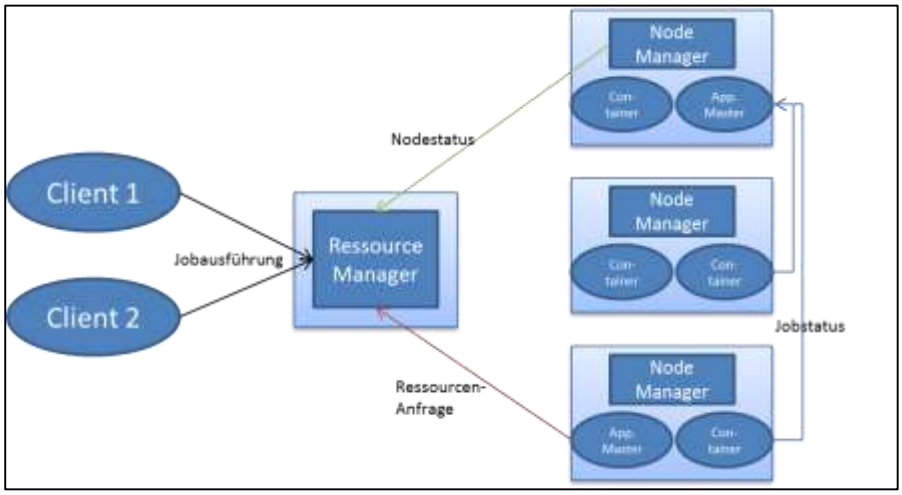

Abbildung 13: YARN[117]

In Abbildung 13 kann beispielsweise angenommen werden, dass Client1 und Client2 jeweils eine Jobausführung auslösen. Der Ressource Manager legt jeweils einen Container als Application Master fest. Der obere Application Master hat bereits auf Anfrage zwei weitere Container als zusätzliche Ressourcen zugeteilt bekommen. Durch die Dezentralisierung der Ausführungsüberwachung kann die Skalierbarkeit nochmals erhöht werden. Die Gefahr zum Flaschenhals zu werden, ist für den Ressource Manager im Vergleich zum Jobtracker wesentlich geringer.

Der zweite Aspekt von YARN ist, dass weiter vom zu verwendenden Programmiermodell abstrahiert wird. Für jeden Application Master kann ein anderes verwendet werden. Dabei ist das klassische MapReduce nur eine der Möglichkei-

---

[114] Vgl. Murthy et al. 2014, S. 49.

[115] Vgl. Murthy et al. 2014, S. 44.

[116] Vgl. Murthy et al. 2014, S. 38.

[117] In Anlehnung an Murthy et al. 2014, S. 39.

ten.[118] Andere Möglichkeiten, die auch in Abbildung 11 vorhanden sind, sind beispielsweise TEZ oder SPARK.

TEZ erlaubt es Applikationen, deren Aufgaben in Form von gerichteten azyklischen Graphen dargestellt werden können, effizient auszuführen. Ein traditioneller MapReduce-Job besteht immer aus einer Map-Aufgabe und einer Reduce-Aufgabe. Müssen mehrere aufeinanderfolgende Reduce-Aufgaben jeweils auf die Ergebnisse des Vorgängers zugreifen, muss immer ein neuer MapReduce-Job gestartet werden. Dies ist nicht sonderlich effizient, da zum einen jeder dieser Jobs initialisiert werden muss und zum anderen jeder obligatorisch eine Map-Aufgabe ausführen muss. TEZ erlaubt es in solchen Fällen mehrere Reduce-Aufgaben hintereinander zu schalten und in einem einzigen Job auszuführen.[119]

Spark ist eine Engine zur Datenverarbeitung mit einer In-Memory-Datenhaltung, die 10-40-mal schneller als MapReduce sein kann.[120] Im Gegensatz zu MapReduce werden Zwischenergebnisse nicht im HDFS persistiert, sondern im Arbeitsspeicher behalten. Verarbeitungen können in erster Linie mit der Programmiersprache Scala entwickelt werden. Es existieren aber ebenfalls Schnittstellen zu gängigen Programmiersprachen, wie z.B. Java.

4.2.2.3 Weitere Komponenten

Im Folgenden wird ein Überblick zu den übrigen Komponenten des Hadoop-Ökosystems aus Abbildung 11 gegeben. Dabei werden Flume und Sqoop als Schnittstellenwerkzeuge vorgestellt. Als Möglichkeiten zur Datenverarbeitung werden Pig, Hive, HBase und Cassandra beschrieben. Mit Mahout wird ein Analysewerkzeug für statistische Auswertungen gezeigt. Schließlich stellen die übrigen Werkzeuge (Chukwa, Zookeeper, Ambari, Oozie, Avro) weitere Hilfsmittel insbesondere auch zur Steuerung und Verwaltung zur Verfügung.

Flume ist ein Werkzeug zum Bewegen von großen Datenmengen. Dabei kann es genutzt werden, um Streaming-Daten in Echtzeit zu verarbeiten. Dies geschieht mit sogenannten Agenten. Ein Agent besteht dabei immer aus drei Komponenten (vgl. Abbildung 14). In der *Source* werden eingehende Daten, die bei Flume auch

---

[118] Vgl. Murthy et al. 2014, S. 38.

[119] Vgl. Apache TEZ 2015.

[120] Vgl. Freiknecht 2014, S. 176.

Events genannt werden, verarbeitet. Dies können zum Beispiel Log-Daten eines Webservers sein. Die Komponente *Sink* schreibt diese Daten wiederum in das Ziel, was im Hadoop-Umfeld typischerweise das HDFS ist. Jedoch kann Flume auch losgelöst von Hadoop eingesetzt werden. Zwischen *Source* und *Sink* findet sich noch ein Zwischenspeicher, der *Channel* genannt wird. Im einfachsten Fall ist dieser ein interner Speicher von Flume. Es ist aber ebenfalls eine Speicherung in einem Dateisystem oder einer externen Datenbank möglich.

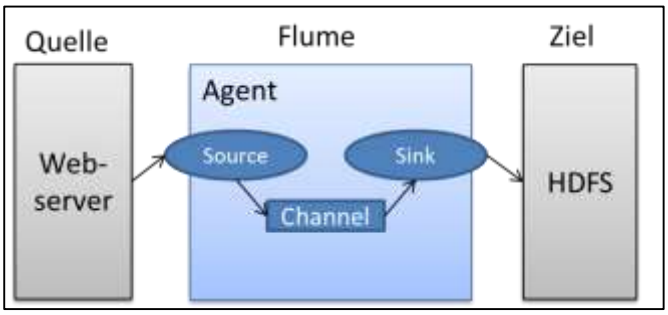

Abbildung 14: Flume[121]

Sqoop dient ebenfalls dem Bewegen von großen Datenmengen. Allerdings ist Sqoop auf den Austausch von Daten zwischen Hadoop und strukturierten Datenspeichern (z.B. relationale Datenbanken) spezialisiert. Daher hat es auch seinen Namen, der sich aus der Verknüpfung der Datenbanksprache **SQ**L mit Had**oop** ableitet.

Pig ermöglicht die Datenverarbeitung in Hadoop mit einer höheren Programmiersprache und zwar der Datenflusssprache Pig Latin. Die Datenflüsse werden automatisch in MapReduce-Programme kompiliert und später auch als solche ausgeführt.[122] Der Entwickler benötigt dabei keine Kenntnisse von dem Programmiermodell.

Mit **Hive** können ebenfalls MapReduce-Programme erzeugt werden ohne das Programmiermodell zu kennen. Dafür wird anstatt einer Datenflusssprache eine SQL-ähnliche Abfragesprache mit dem Namen HiveQL verwendet. Außerdem können in Hive Tabellenstrukturen angelegt werden. Daher wird es auch als das Hadoop DWH bezeichnet. Allerdings handelt es sich bei Hive-Tabellen lediglich

---

[121] In Anlehnung an Apache Flume 2014.

[122] Vgl. Gates 2011, S. 5.

um eine Metadatenschicht über Daten, die sich im HDFS befinden. Dadurch ist beispielsweise das Anlegen von Indizes oder ein Update auf Einzelsatzebene nicht möglich. Im Vergleich zu relationalen Datenbanken hat Hive genauso wie Pig selbst bei einfachen Abfragen aufgrund von MapReduce hohe Latenzzeiten.[123] Mit YARN können beide jedoch auch TEZ verwenden, was die Laufzeiten verbessert.[124] Hive und Pig schließen sich gegenseitig nicht aus, sondern können auch gut kombiniert eingesetzt werden, da beide ihre jeweiligen Stärken haben.[125]

HBase (Hadoop Data**base**) ist eine spaltenorientierte NoSQL-Datenbank, die Hadoop zur Verfügung stellt. Sie ist in Anlehnung an Google´s BigTable entworfen und im Gegensatz zur MapReduce-Batch-Verarbeitung vorwiegend für Realtime-Anwendungen gedacht.[126]

Cassandra ist eine von Facebook erstellte NoSQL-Datenbank, die ebenfalls wie HBase spaltenorientiert ist. Zwischen diesen beiden Datenbanken bestehen noch weitere Gemeinsamkeiten.[127] Beispielsweise basiert das Datenmodell ebenfalls auf Google´s BigTable. Allerdings existieren auch Unterschiede. So ist Amazon Dynamo noch eine weitere Vorlage für Cassandra, der insbesondere das verteilte Design nachgebaut ist.[128] Ein weiterer Unterschied ist, dass Hbase speziell für Hadoop und das HDFS entwickelt ist. Für Cassandra ist dagegen die Integration in Hadoop nur eine der Einsatzmöglichkeiten.

Bei **Mahout** handelt es sich um eine Bibliothek mit skalierbaren Algorithmen für maschinelles Lernen und statistische Berechnungen.[129] Diese können auch eingesetzt werden, um die in Abschnitt 3.3 vorgestellten Data-Mining-Aufgaben zu lösen.

---

[123] Vgl. Frisch 2014, S. 25.

[124] Vgl. Murthy et al. 2014, S. 242.

[125] Vgl. Freiknecht 2014, S. 318.

[126] Vgl. Apache HBase 2015.

[127] Vgl. Grehan 2014.

[128] Vgl. Hewett 2011, S. 14.

[129] Vgl. Apache Mahout 2015.

Chukwa dient dazu, Daten zum Überwachen von großen, verteilten Systemen zu sammeln, aufbereiten und visualisieren.[130]

Ambari bietet ebenfalls Möglichkeiten, um ein Hadoop-Cluster einfach überwachen und verwalten zu können. Auch bei der Einrichtung eines Hadoop-Clusters bzw. neuer Knoten kann Ambari genutzt werden. Für die Zukunft scheint es auch im Vergleich zu Chukwa das gesetzte Überwachsungswerkzeug zu sein, da Chukwa seit 2012 keine nennenswerten Erweiterungen erhalten hat.[131]

Mit Hilfe von **ZooKeeper** können Konfigurationsinformationen zentral gepflegt werden. Es koordiniert und synchronisiert die Teilnehmer in verteilten Anwendungen.[132]

Avro ist ein Datenserialisierungs-Rahmenwerk, welches Hadoop nutzt, um Daten persistent zu schreiben oder auch zwischen Knoten oder Services auszutauschen. Der Vorteil liegt in der Unabhängigkeit von der Programmiersprache der datenverarbeitenden Anwendung.[133]

Oozie ist ein Workflow-Scheduler zur Planung und Ausführung von Hadoop-Jobs.[134] Es kann genutzt werden, um Prozessketten, die aus Hadoop-Jobs bestehen, zu erstellen und auch eine automatische Ausführung einzuplanen.

4.3 Soziale Netzwerke als Quelle

Durch die Möglichkeit größere Datenmengen und eine größere Vielfalt von Datentypen immer schneller verarbeiten zu können, werden durch Big Data auch neue Datenquellen für Analysen relevant. Als ein Beispiel für solch eine Datenquelle werden in diesem Abschnitt soziale Netzwerke mit dem Fokus auf das Beispiel Twitter vorgestellt. Abschnitt 4.3.1 hebt dazu zunächst die Besonderheiten von Daten aus sozialen Netzwerken im Vergleich zu Datenquellen für klassische Datenauswertungen hervor. Dabei wird auch diskutiert, welche Chancen und Schwierigkeiten solche Daten mit sich bringen. In Abschnitt 4.3.2 wird ein allge-

---

[130] Vgl. Apache Chukwa 2015.

[131] Vgl. Freiknecht 2014, S. 182.

[132] Vgl. Apache Zookeeper 2015.

[133] Vgl. White 2015, S. 345.

[134] Vgl. Apache Oozie 2014.

meiner Überblick über Twitter gegeben. Eine technische Sicht auf die von Twitter angebotene Schnittstelle zur Datenextraktion liefert Abschnitt 4.3.3.

4.3.1 Chancen und Schwierigkeiten

Die Auswertung von Daten aus sozialen Netzwerken eröffnet Unternehmen Chancen, aber bringt ebenfalls einige Schwierigkeiten mit sich.

Insbesondere dem Marketing stehen durch Daten aus sozialen Netzwerken große Datenmengen zur Verfügung, deren Analyse Potential hat, den Kunden sowie den Markt besser zu durchdringen.[135] So können beispielsweise bereits vor Verkaufsstart eines neuen Produktes anhand der Nachrichten in sozialen Netzwerken Aussagen zum Erfolg gemacht werden. Zu finden sind solche Analysen insbesondere für prominente Produkte von bekannten Herstellern wie z.B. aktuell die Apple Watch.[136] Dabei kann bereits ein Indikator sein, wie viel die Nutzer sozialer Netzwerke sich mit diesem Produkt beschäftigen bzw. wie oft es erwähnt wird.

Aber auch zu weniger prominenten Produkten und Unternehmen existiert eine Vielzahl an frei verfügbaren Meinungen in sozialen Netzwerken. Mit diesen kann die klassische Markforschung modernisiert bzw. erweitert werden. Es können Informationen zu Interessen, Trends, Wettbewerbern und Meinungen aus den sozialen Netzwerken abgeleitet werden. Im Gegensatz zum klassischen Fragebogen kann eine Auswertung sozialer Netzwerke Vor- und Nachteile mit sich bringen. So sind die Daten in sozialen Netzwerken weniger strukturiert als in einem Fragebogen mit Auswahlmöglichkeiten und nicht passend auf die jeweiligen Fragen zugeschnitten. Dies erhöht den Aufwand bei der Auswertung. Insbesondere weil Freitexte auch unbrauchbare oder nicht relevante Bestandteile haben können, die es von den relevanten zu trennen gilt.[137] Umgekehrt bietet dies auch Chancen. So können Fragen beantwortet und Erkenntnissen gewonnen werden, die wichtig sind, aber im Fragebogen nicht abgedeckt werden. Die Daten aus sozialen Netzwerken können einfacher, schneller und großflächiger abgegriffen werden. Ein weiterer Nutzenpunkt von Daten aus sozialen Netzwerken ist die Verknüpfung mit bereits im Unternehmen vorhandenen Daten. So können Daten aus sozialen

---

[135] Vgl. Bachmann et al. 2014, S. 329.

[136] Vgl. Kummerfeld 2015.

[137] Vgl. Inmon et al. 2008, S. 38.

Netzwerken zusammen mit anderen Daten genutzt werden, um eine 360-Grad-Kundensicht zu erstellen, die wiederum zur Personalisierung von Marketing-Maßnahmen eingesetzt werden kann.[138] Eine Erkenntnis, die nicht Bestandteil einer klassischen Stammdatenerfassung ist, kann dabei der Stand in der sozialen Umgebung sein (z.B. Größe des Bekanntenkreises). Dabei muss allerdings berücksichtigt werden, dass nicht jeder soziale Netzwerke nutzt und auch nicht jeder gleich intensiv. Dementsprechend muss für die Übertragbarkeit auf die Realität und bei Verallgemeinerungen eine gewisse Vorsicht geboten sein. Bei Verallgemeinerungen muss geprüft werden, inwieweit die Autoren in den sozialen Netzwerken auch repräsentativ für die zu betrachtende Kundengruppe sind.[139] Weitere Herausforderungen entstehen dadurch, dass die Daten unsicher sind. Sowohl Inhalte bzw. Nachrichten selbst als auch gesamte Profile können unecht bzw. manipuliert sein.[140] Es gibt keinerlei Prüfungen, dass der Inhalt von Nachrichten auch korrekt ist. Einige soziale Netzwerke (z.B. Twitter) erlauben Avatare als Profile. Das erlaubt den Nutzern Nachrichten anonym zu veröffentlichen. Auch dies ist zwiespältig zu sehen. Durch den Schutz der Anonymität werden möglicherweise offener Meinungen preisgegeben. Dagegen vereinfacht dies auch Falschmeldungen durch böswillige Nutzer.[141]

4.3.2 Überblick Twitter

Twitter ist ein sogenannter Mikroblogging-Dienst, welcher es seinen Nutzern ermöglicht über Kurznachrichten mit einer maximalen Länge von 140 Zeichen zu kommunizieren. Dies entspricht in etwa der Länge für einen Gedanken oder eine Idee.[142] Die Kurznachrichten werden bei Twitter als Tweets bezeichnet und sind öffentlich. Jeder Nutzer hat eine sogenannte Timeline, in der Beiträge angezeigt werden. Es besteht die Möglichkeit, dass jeder Nutzer „Follower“ von jedem beliebigen anderen Nutzer sein kann und sich auf diese Weise dessen Beiträge abonniert. Ebenso ist es möglich, Tweets zu kommentieren oder zu teilen. Letzteres wird als Retweet bezeichnet. Zum Hervorheben von besonderen Begriffen

---

[138] Vgl. King 2013, S. 64.

[139] Vgl. Lütters / Egger 2013, S. 40.

[140] Vgl. Bachmann et al. 2014, S. 305.

[141] Vgl. Bin 2012, S. 7.

[142] Vgl. Russel 2014, S. 7.

können diese als Hashtag markiert werden. Dies geschieht mit einem vorangestellten Doppelkreuz („#“).

Neben Twitter existiert eine große Anzahl weiterer sozialer Netzwerke. Dabei zählen gemessen an der Anzahl an Besuchern im Dezember 2014 neben Twitter auch Facebook und Google Plus zu den drei beliebtesten sozialen Netzwerken in Deutschland. Bei Twitter ist dabei im Gegensatz zu Facebook eine positive Entwicklung zu erkennen.[143] Ein Vergleich dieser drei sozialen Netzwerke zeigt Tabelle 1.

Eigenschaft	Twitter	Facebook	Google Plus
Ziele / Fokus	Mikroblogging, Veröffentlichen	Freundschaften, Kommunizieren	Austausch, Kontakte knüpfen
Erscheinungsdatum	15.07.2006	04.02.2004	28.06.2011
Sichtbarkeit	Nahezu alles	Nur freigegebene Inhalte	Nur freigegebene Inhalte
Beziehungsmodell	Follower. Jeder kann jedem folgen (Unidirektional).	Freunde. Bidirektionale Zustimmung.	Kreise. Unidirektional.
Nachrichtendienst	Direktnachrichten (Private Nachrichten ebenfalls 140 Zeichen)	Facebook-Messenger für Chats	Chat mit Google Hangouts (Möglichkeit von Sprach und Videoanrufe)
Weitere Besonderheiten / Funktionen		Spiele, Applikationen	Integration mit anderen Google-Diensten

Tabelle 1: Vergleich Twitter, Facebook, Google+

Twitter eignet sich sehr gut als Quelle für Datenauswertungen. Insbesondere kommt dabei das Ziel, dass die Tweet-Autoren veröffentlichen wollen, zu gute. Dadurch ist auch der Großteil der Twitter-Inhalte für jeden frei verfügbar. Auch die Beschränkung auf 140 Zeichen, erleichtert eine Auswertung, da dadurch die Veröffentlichungen kurz und prägnant sind.[144] Allerdings können dadurch auch vermehrt Abkürzungen und Umgangssprache verwendet werden, die schwerer zu verstehen sind.

[143] Vgl. Schröder 2015.

[144] Vgl. Bing 2012, S. 9f.

4.3.3 Twitter Schnittstelle

Twitter bietet zwei Schnittstellen / API´s (Application Programming Interface) an, die genutzt werden können, um Tweets zu extrahieren. Dies sind zum einen die REST-API (Representational State Transfer) und zum anderen die Streaming-API. Um diese nutzen zu können, muss zunächst ein Entwickler-Zugang bei Twitter eingerichtet werden. Mit Hilfe von diesem können Applikationen registriert und Schlüssel für die Autorisierung generiert werden. Twitter nutzt zur Autorisierung das Protokoll OAuth (Open Authorization), welches als Standard für soziale Netzwerke gilt.[145]

Die Twitter-API stellt eine Vielzahl an Informationen über die gut dokumentierte Schnittstelle bereit. Diese werden dabei nach den vier Objekten Nutzer, Tweets, Entitäten und Orte gegliedert.[146] Nutzerobjekte enthalten zum einen Stammdaten, die Nutzer selber im Profil pflegen bzw. die aus diesem abgeleitet werden, wie zum Beispiel der Name, der Ort oder das Erstellungsdatum. Zum anderen enthalten sie aber auch Informationen, die sich aus Interaktionen bzw. den Beziehungen zu anderen Nutzern ergeben (z.B. Anzahl der Favoriten oder Follower). Tweet-Objekte umfassen neben dem eigentlichen Tweet-Text zahlreiche Metadaten. Zu diesen zählen beispielsweise das Datum, die Uhrzeit, der Ort oder die Sprache der Veröffentlichung des Tweets. Zusätzliche Metadaten werden als Entität-Objekt gespeichert. Dies können insbesondere Interpretationen bzw. Aufbereitungen von Twitter bezüglich des Tweet-Textes sein. Beispiele sind die verwendeten Hashtags, die genannten Nutzer (mit dem Zeichen @ kann auf andere Nutzer verwiesen werden) und auch verwendete URLs (Uniform Resource Locator). Das letzte Objekt sind Orte. Diese umfassen neben dem Namen, auch die Koordinaten, das Land und den Typ (z.B. Stadt, Stadtteil, Region).

Die **REST-API** bietet mit einer Such-API Möglichkeiten durch diese Informationen flexibel und intuitiv zu navigieren.[147] Dabei stehen die Daten aus den letzten fünf Tagen zur Verfügung. Diese können mit nahezu beliebigen Filterkriterien über Http-Get-Anfragen abgefragt werden. Möglich ist beispielsweise eine Filte-

---

[145] Vgl. Russel 2014, S. 13.

[146] Vgl. Twitter 2015.

[147] Vgl. Russel 2014, S. 12.

rung von Tweets zu einem bestimmten Hashtag, von einem bestimmten Nutzer oder mit einem bestimmten Veröffentlichungszeitpunkt. Das Ergebnis wird im JSON-Format ausgegeben. Viele Programmiersprachen haben bereits öffentliche Bibliotheken, um die API zu nutzen (z.B. Twitter4J für Java). Solche Bibliotheken stehen auch für Werkzeuge wie R oder Python zur Verfügung. Allerdings unterliegt die REST-API bestimmten Abfragebeschränkungen, die zu beachten sind. So sind beispielsweise nur 180 Anfragen je Viertelstunden möglich.[148] Für manuelle Anfragen erscheint dies zunächst viel. Werden die Anfragen jedoch automatisch durch eine Applikation generiert und abgesetzt, können diese schnell aufgebraucht sein.

Einen anderen Ansatz hat die **Streaming-API**. Diese liefert in nahezu Echtzeit veröffentlichte Tweets. Dazu wird eine dauerhafte Http-Verbindung mit bestimmten Parametern aufgebaut und Twitter leitet passend dazu die Tweets weiter. Über die Parameter kann somit ebenfalls gefiltert werden, welche Tweets benötigt werden. Jedoch sind die Möglichkeiten dort eingeschränkter als bei der REST-API. Wichtig ist, dass auf Empfänger-Seite ein Dienst existieren muss, der die Tweets annehmen kann. Es kann aus drei unterschiedlichen Streams auf diese Weise mitgeschnitten werden. Dabei liefert der Public Stream eine Teilmenge aller öffentlichen Tweets. Daten und Ereignisse zum aktuell angemeldeten Nutzer können über den User Stream empfangen werden. Eine Erweiterung des User Streams ist der Site Stream. Dieser kann Daten zu beliebigen Nutzern enthalten. Jedoch befindet er sich laut Twitter-Dokumentation noch in einer Beta-Phase und ist daher nur ausgewählten Zugängen vorbehalten.[149]

4.4 Sentiment-Analyse in sozialen Netzwerken

Möglichkeiten um Informationen und Wissen aus Daten von sozialen Netzwerken zu generieren, lassen sich unter dem Begriff Social Media Mining, welches als Teilgebiet des Data Mining betrachtet werden kann, zusammenfassen. Eine der Möglichkeiten dabei ist die Sentiment-Analyse. Diese nutzt unter anderem Verfahren aus dem Text-Mining, Opinion-Mining und NLP (Natural Language Processing), um automatisiert die Stimmung eines Textes zu ermitteln. Im Folgenden

[148] Vgl. Twitter 2015.

[149] Vgl. Twitter 2015.

wird zunächst in Abschnitt 4.4.1 näher beschrieben, was ein Sentiment ist und anschließend werden mögliche Verfahren zur Ermittlung vorgestellt. Diese lassen sich grob in lexikonbasierte Ansätze (vgl. Abschnitt 4.4.2) und mathematische bzw. statistische Methoden (vgl. Abschnitt 4.4.3) unterteilen.

4.4.1 Ausdruck der Stimmung mit Sentiments

Ein Sentiment drückt die Neigung oder die Stimmung von jemandem zu einer Entität aus. Diese kann beispielsweise eine bestimmten Person, ein Ort, eine Sache, ein Thema, eine Organisation oder deren Eigenschaften sein.[150] Typischerweise ist diese Stimmung mit einer positiven oder negativen Wertung verknüpft. Hinsichtlich der Granularität kann ein Sentiment auf drei Ebenen bestimmt werden.[151] Die gröbste Ebene ist dabei das Dokument. Dabei wird ein Gesamtsentiment für ein komplettes Dokument ermittelt, was beispielsweise ein Artikel, eine Rezension oder ein Tweet sein kann. Detaillierter ist eine Auswertung auf Satzebene. Da aber auch innerhalb eines Satzes unterschiedliche und sogar gegensätzliche Sentiments möglich sind, ist noch feingranularer eine Angabe auf Entität- bzw. Aspektebene. Auf dieser Ebene lässt sich ein Sentiment mit dem Quintupel (h,g,s,p,c) beschreiben:[152]

Dabei steht h für den Holder bzw. **Halter** des Sentiments, was üblicherweise (z.B. bei einem Tweet) dem Autor entspricht.

Mit dem g (Goal) wird das **Zielobjekt** umschrieben. Dies entspricht der obengenannten Entität, auf die sich das Sentiment bezieht. Wie erwähnt kann dies eine Person, ein Ort oder eine andere beliebige Sache sein.

Das **Sentiment** (s) selbst ist auch ein Teil dieses Quintupels. In einem Text ist es meist ein Adjektiv, welches das Zielobjekt umschreibt.

Das **Ausmaß** bzw. Stärke eines Sentiments entspricht dem p.

---

[150] Vgl. Liu 2012, S, 1.

[151] Vgl. Liu 2012, S. 4.

[152] Vgl. Danneman / Heimann 2014, S. 44f.

Zuletzt bezieht sich c (cause) auf den **Ursprung** bzw. die Ursache eines Sentiments. Da dieser aber oft nicht offensichtlich ist, werden stattdessen auch Informationen zu Ort und Zeit der Entstehung genutzt.[153]

Es ist also zu erkennen, dass in den meisten Fällen ein Text alleine nicht ausreicht, um einen Sentiment vollumfänglich zu umschreiben. Insbesondere zur Ermittlung des Halters und des Ursprungs sind weitere Metadaten notwendig. An folgendem fiktiven Beispiel-Tweet werden die fünf Bestandteile nochmals verdeutlicht: „Ich finde den heutigen Tatort sehr gut".

In diesem Fall ist der Twitter-Nutzer, der den Tweet publiziert hat, der Halter. Der „heutige Tatort" ist das Zielobjekt. „Gut" bzw. positiv ist das Sentiment und die Stärke ist hoch, da das Sentiment noch mit dem Wort „sehr" verstärkt ist. Für Auswertungen ist es hilfreich diese zu quantifizieren (z.B. auf einer Skala von -1 bis 1). Zum Ursprung kann der Zeitpunkt und, wenn vorhanden, der Ort der Publizierung genutzt werden.

4.4.2 Lexikonbasierter Ansatz

Der lexikonbasierte Ansatz gilt als ein sehr einfacher und zugleich effizienter Ansatz für die Praxis.[154] Bei diesem wird der zu bewertende Text mit einem sogenannten Sentiment-Lexikon abgeglichen. Solch ein Lexikon enthält Wörter oder Phrasen, die eine bestimmte Neigung ausdrücken. Im einfachsten Fall ist diese binär in positiv oder negativ angegeben. Um schließlich ein Gesamtsentiment für einen Text oder Satz zu erhalten, können die Wörter des Textes mit solch einem Lexikon abgeglichen werden. Auch als Ergebnis kann in der einfachsten Variante eine binäre Aussage zwischen positiv und negativ in Abhängigkeit davon getroffen werden, ob mehr positive oder mehr negative Wörter gefunden werden können. Durch die Hinzunahme von Gewichtungen oder einer dritten neutralen Kategorie kann der Ansatz entsprechend genauere Ergebnisse liefern. Dies ist allerdings auch mit einer höheren Komplexität verbunden.

Die Stärke von diesem Ansatz liegt neben der einfachen Anwendung in einer hohen Transparenz. So können die Ergebnisse recht intuitiv nachvollzogen und

---

[153] Vgl. Danneman / Heimann 2014, S. 45.

[154] Vgl. Vinodhini / Chandrasekaran 2012, S. 285.

durch Änderungen in dem Sentiment-Lexikon beeinflusst werden. Allerdings bringt der Ansatz auch einige Schwierigkeiten mit sich, die im Wesentlichen darauf zurückzuführen sind, dass das Problem der Sentiment-Analyse zu komplex ist, um es komplett über ein Wörterbuch abzubilden.[155] So hat häufig der Kontext einen wesentlichen Einfluss auf ein Sentiment. Beispielsweise kann das Wort „langsam“ bei der Bewertung eines Läufers als sehr negativ ausgelegt werden. Bei der Beschreibung einer Melodie kann es dagegen auch positiv genutzt werden. In einem anderen Kontext kann es wiederum gar keine Stimmung oder Meinung ausdrücken. Ein ähnliches Problem stellen Homonyme dar, da diese in Abhängigkeit vom Kontext eine komplett andere Bedeutung haben können. Auch wie ein Wort benutzt wird, kann Einfluss darauf haben, ob ein Sentiment vorliegt oder nicht. So drückt der Satz „Das Wetter wird heute gut“ eine positive Stimmung aus. Mit den gleichen Worten kann allerdings auch eine Frage gebildet werden „Wird das Wetter heute gut?“, die kein Sentiment enthält. Eine weitere Schwierigkeit in der menschlichen Sprache sind Sarkasmus und Ironie, die ein Wörterbuch nicht erkennen kann.

Eine der Herausforderungen bei diesem Ansatz ist die Wahl bzw. Erstellung eines adäquaten Sentiment-Lexikons, da dieses wesentlichen Einfluss auf die Qualität der Analyse-Ergebnisse hat. Zunächst muss dabei entschieden werden, ob ein allgemeingültiges oder spezielles Lexikon für eine bestimmte Domäne verwendet werden soll.[156] Letzteres kann in gewissem Maße dem zuvor genannten Problem entgegenwirken, dass Wörter in unterschiedlichem Kontext ein unterschiedliches Sentiment haben können. Dafür wird möglicherweise nicht für jede Domäne ein vorgefertigtes verfügbar sein. Soll das Lexikon selbst erstellt werden, ist es wichtig einen geeigneten Ansatz zu wählen. Neben der Möglichkeit einer aufwändigen manuellen Erfassung und Pflege von Sentiment-Wörtern bilden insbesondere zwei Verfahren den Ursprung für Möglichkeiten, die bei einer automatisierten Erstellung unterstützen können. Bei beiden Ansätzen wird eine Basismenge an bereits bewerteten Wörtern benötigt. Der erste Ansatz nutzt gewöhnliche Wörterbücher, um diese Basismenge automatisiert zu erweitern. Dabei macht es sich zu Nutze,

---

[155] Vgl. Liu 2012, S. 5.

[156] Vgl. Danneman / Heimann 2014, S. 60.

dass diese häufig über eine Vielzahl von Synonymen und Antonymen verfügen.[157] Durch eine Aufnahme von diesen Synonymen als neue Wörter mit der gleichen Neigung und Antonymen entsprechend mit der entgegengesetzten Neigung, kann der Umfang des Sentiment-Lexikons sehr schnell erweitert werden. Verstärkt werden kann dieser Effekt durch eine mehrmalige bzw. rekursive Ausführung des Algorithmus, sodass beispielsweise auch die Synonyme von den Synonymen mit betrachtet werden. Der zweite Ansatz nutzt als Quelle zur Anreicherung einen Korpus mit Dokumenten. In diesem werden die bereits bekannten Sentiment-Wörter gesucht und umliegende Wörter anhand ihrer Verknüpfung bewertet. So wird beispielsweise davon ausgegangen, dass zwei Wörter, die mit „und" verbunden sind, die gleiche Neigung haben. Umgekehrt haben mit „aber" verknüpfte Wörter eine entgegengesetzte Neigung.[158] Da bei der Erstellung bereits ein Korpus speziell für diese Domäne genutzt werden kann, ist dieser Ansatz insbesondere bei der Erstellung eines domänenspezifischen Wörterbuches hilfreich.

Für die englische Sprache existieren mittlerweile viele kommerzielle und frei verfügbare Sentiment-Lexika. Für die deutsche Sprache sind dagegen nur wenige frei verfügbar zu finden. Eines ist SentiWS (SentimentWortschatz), welches über 3000 positiv und negativ bewertete sowie auf einer Skala von -1 bis 1 gewichtete Wörter verfügt.[159] Unter anderem werden bei diesem die beiden zuvor vorgestellten Ansätze kombiniert angewendet. Dabei basiert es auf drei Quellen.[160] Die ist zunächst ein in englischer Sprache bereits vorhandenes und teilautomatisiert übersetztes Sentiment-Lexikon (General Inquirer). Die zweite Quelle nutzt den korpusbasierten Ansatz. Jedoch in einer etwas anderen Variante als oben beschrieben. Es wird ein Korpus für Produktbewertungen, die bereits in positiv oder negativ klassifiziert sind, verwendet. Danach wird analysiert, welche Wörter besonders oft in positiven bzw. in negativen Produktbewertungen auftauchen und diese dem Sentiment-Lexikon angefügt. Die letzte Quelle ist ein deutschsprachiges Wörterbuch für Kollokationen. Dieses reichert die bereits vorhandenen Wörter zwar nicht wie zuvor beschrieben um Synonyme an, aber um weitere Wortformen (z.B.

---

[157] Vgl. Liu 2012, S. 80.

[158] Vgl. Liu 2012, S. 83f.

[159] Vgl. Remus et al. 2010, S. 1168.

[160] Vgl. Remus et al. 2010, S. 1169.

mit anderem Kasus, Numerus oder Genus). In einem zweiten Schritt werden die Wörter mit dem sogenannten PMI-Ansatz (Pointwise Mutual Information) gewichtet. Bei diesem wird aus der Wahrscheinlichkeit P, dass zwei Wörter w1 und w2 alleine oder gemeinsam auftreten, die semantische Assoziation quantifiziert. Dazu wird folgende Formel verwendet:

$$PMI(w1, w2) = log_2(\frac{P(w_1, w_2)}{P(w_1) \times P(w_2)})$$

Um die Wahrscheinlichkeiten zu ermitteln, wird nochmals eine Variante des zuvor vorgestellten korpusbasierten Ansatzes verwendet. Dabei werden einige wenige positive und negative Suchwörter festgelegt. Mit diesem Korpus werden für alle Wörter aus dem zuvor erstellten Lexikon (w1) und diesen Suchwörtern (w2) jeweils die Wahrscheinlichkeiten des Auftretens insgesamt und gemeinsam bestimmt. Die eigentliche Gewichtung für ein Wort aus dem Lexikon ergibt sich dann schließlich aus der Summe der PMI zu den positiven Suchwörtern abzüglich dieser Summe zu den negativen Suchwörtern.[161]

4.4.3 Statistische Verfahren zur Klassifizierung

Neben dem lexikonbasierten Ansatz zur Sentiment-Analyse ist auch die Nutzung von Verfahren aus der Statistik möglich. Dabei können auch die in Abschnitt 3.3.3 vorgestellten Verfahren genutzt werden. Die Sentiment-Analyse ist dabei ein Klassifizierungsproblem, welches ein Dokument in die Klassen positiv, negativ und optional neutral einordnet.[162] Vorwiegend eignen sich dazu überwachte Verfahren aus dem maschinellen Lernen (z.B. Naive Bayés oder Support Vector Machine).[163] Im Folgenden wird dazu beispielhaft eine Möglichkeit der Sentiment-Analyse mit dem Naive Bayés näher betrachtet. Mit Hilfe einer Menge an bereits bewerteten Dokumenten, die als Trainings- und Testdaten für den Algorithmus genutzt werden können, wird ein Modell mit konditionalen Wahrscheinlichkeiten abgeleitet wie es beispielsweise in Tabelle 2 zu sehen ist.

---

[161] Vgl. Remus et al. 2010, S. 1169.

[162] Vgl. Bing 2012, S. 24.

[163] Vgl. Vinodhini / Chandrasekaran 2012, S. 283.

Wort	Vorkommen in allen Dokumenten	Vorkommen in positiven Dokumenten	Wahrscheinlichkeit Zugehörigkeit zur Klasse positiv
Gut	15	12	80%
Schlecht	10	1	10%
freuen	8	5	62,5%

Tabelle 2: Naive Bayés Modelleerstellung

Dieses enthält Wahrscheinlichkeiten dafür, dass beim Vorkommen bestimmter Wörter oder Phrasen das Dokument zur Kategorie positiv oder negativ gehört.

Beim Naive Bayés wird die einfache Annahme getroffen, dass alle Eigenschaften gleichbedeutend und unabhängig voneinander sind.[164] Dadurch können selbst bei vielen Eigenschaften und großen Mengen an Trainingsdaten schnelle Ergebnisse erzielt werden.[165] Soll schließlich ein neues Dokument klassifiziert werden, vergleicht der Naive-Bayés-Klassifikator die Eigenschaften dieses Dokumentes mit dem Modell. Die Wahrscheinlichkeit einer Klassenzugehörigkeit ergibt sich dabei aus dem Produkt der jeweiligen konditionalen Wahrscheinlichkeiten der gefundenen Merkmale ($P(F_i|C_L)$), multipliziert mit der Wahrscheinlichkeit für das Vorkommen der Klasse selbst($P(C_L)$):[166]

$$P(C_L|F_1 \dots F_n) = P(C_L) \times \prod_{i=1}^{n} P(F_i|C_L)$$

Die Qualität der Ergebnisse hängt bei solch einem Ansatz im Wesentlichen von der Qualität und der Repräsentativität der Trainings- und Testdaten ab. Ein Vorteil dabei ist, dass neu bewertete Dokumente direkt auch wieder genutzt werden können, um die Datenbasis für die Trainingsdaten zu erweitern, was im Idealfall die Entscheidungen des Algorithmus verbessert. Jedoch entsteht dadurch auch das Problem, dass mit der Zeit diese immer intransparenter werden. Ein weiterer Nachteil ist, dass es zur Nutzung auf der Ebene einer einzelnen Entität bzw. eines

[164] Vgl. Lantz 2013, S. 95.

[165] Vgl. Danneman / Heimann 2014, S. 62.

[166] Vgl. Lantz 2013, S. 98.

Aspektes kaum geeignet ist. Typischerweise kommt es auf Dokumentebene zum Einsatz.

4.5 Integration von Big Data in das klassische Data Warehouse

Zum Abschluss dieses Kapitels diskutiert dieser Abschnitt Möglichkeiten die Ansätze aus diesem Kapitel mit den klassischen Ansätzen aus Kapitel 3 zu vereinen. Dazu werden in Abschnitt 4.5.1 Synergieeffekte und Herausforderungen aufgezeigt und in Abschnitt 4.5.2 mögliche Architekturen vorgestellt.

4.5.1 Synergieeffekte und Herausforderungen

Bei der Verknüpfung der Ansätze aus dem klassischen Daten Warehouse mit den Ansätzen aus diesem Kapitel entstehen sowohl aus inhaltlicher als auch aus technischer Sicht Chancen, die allerdings auch mit neuen Herausforderungen einhergehen.

Inhaltlich wurde bereits in Abschnitt 4.3.1 dargelegt, dass eine Verknüpfung der klassischen Unternehmensdaten mit Daten aus sozialen Netzwerken großes Potenzial bietet den Kunden und den Markt besser zu verstehen. Jedoch muss im Unternehmen auch ein Bewusstsein dafür geschaffen werden, dass die neuen Datenquellen auch unsichere subjektive Daten liefern und nicht nur Fakten, wie es im klassischen Data Warehouse üblich ist.[167] Durch große Datenmengen bzw. viele subjektive Meinungen kann eine gewisse Objektivität erreicht und können einzelne Falschmeldungen bzw. Fehler verschleiert werden.[168] Dennoch wird sich eine gewisse Unsicherheit nicht vermeiden lassen.

Aus technischer Sicht können einige Komponenten aus einem klassischen BI-System ebenfalls auch für die Aufbereitung und Auswertung von Daten aus dem Big-Data-Umfeld genutzt werden bzw. unterstützen. So kann in manchen Fällen das Visualisierungswerkzeug beibehalten werden. Bei der Orchestrierung der Datenaufbereitungsprozesse kann möglicherweise ein ETL-Werkzeug unterstützen, indem es die Ausführung bestimmter Betriebssystembefehle oder Skripte steuert. Auch die relationale Datenhaltungskomponente kann genutzt werden. So eignet sich diese gut um Steuerinformationen, die bei Verarbeitungen genutzt werden, zu

---

[167] Vgl. Bing 2012, S. 9.

[168] Vgl. Freiknecht 2014, S. 173.

pflegen oder zur Datensicherung der Ergebnisse von Analysen, die unstrukturierte in strukturierte Daten überführen. Allerdings müssen in der Gesamtarchitektur auch neue Komponenten mit aufgenommen werden. Dies beginnt mit der Extraktion der Daten, da klassische ETL-Werkzeuge zwar viele Möglichkeiten bieten, strukturierte Datenquellen (relationale Datenbanken, csv-Dateien) zu nutzen, aber beispielsweise noch keine Lösungen mitbringen, um Web 2.0-Inhalte (z.B. aus sozialen Netzwerken, Foren, Blogs, etc.) direkt abzugreifen. Für die Datenhaltung und -verarbeitung wurden in Abschnitt 4.2 verschiedene Möglichkeiten aufgezeigt, die in Abhängigkeit vom Anwendungsfall die Architektur sinnvoll ergänzen können.

4.5.2 Mögliche Architekturen

Grundsätzlich können drei unterschiedliche Architekturvarianten, die relationale Ansätze mit Big-Data-Ansätzen vereinen, unterschieden werden. Am Beispiel Hadoop werden diese in Dijcks 2014 als Reife-Phasen von Hadoop- und RDBMS-Implementierungen bezeichnet.

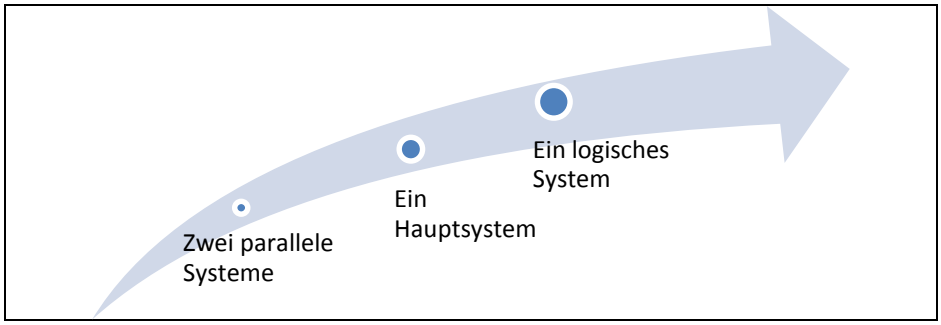

Abbildung 15: Reife-Phasen Big Data mit relationaler Welt[169]

In der ersten Stufe existieren das Big-Data-System und die relationale Welt als zwei voneinander getrennte und unabhängige Systeme (vgl. Abbildung 16a). Die Big-Data-Lösung ist damit isoliert von den übrigen Systemen und kann nicht von den zuvor vorgestellten Synergieeffekten profitieren. Auf der zweiten Stufe existiert ein Hauptsystem, welches die Daten aus beiden konsolidiert. Dabei können sowohl die Daten aus dem Big-Data-System nach einer ausreichenden Strukturierung in die relationale Datenbank übertragen werden (vgl. Abbildung 16c) als auch umgekehrt die Daten aus dem DWH in eine Big-Data-Umgebung (vgl. Ab-

[169] Vgl. Dijcks 2014, S. 12.

bildung 16d). Die Überführung der relationalen Daten in eine Big-Data-Umgebung erscheint dabei zwar zunächst einfacher. Dafür entsteht aber möglicherweise mehr Aufwand bei der Datenbereitstellung für das BI-Werkzeug zur Visualisierung. Eine Mischung aus der ersten und zweiten Stufe ist eine komplett getrennte Datenhaltung und -verarbeitung, aber eine gemeinsame Konsolidierung im BI-Werkzeug (vgl. Abbildung 16b).

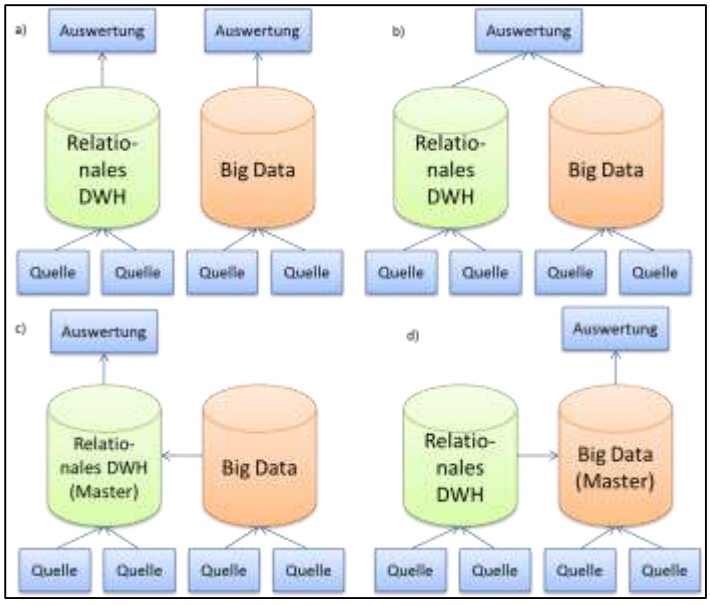

Abbildung 16: Architekturvarianten

Die letzte Variante ist ein logisches System (vgl. Abbildung 17).

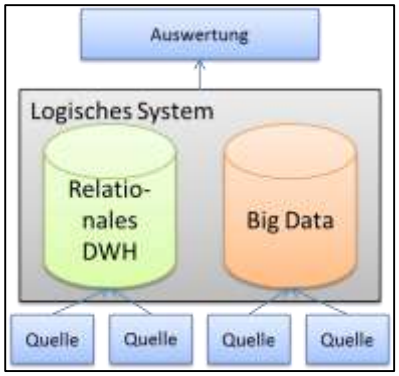

Abbildung 17: Ein logisches System

Dabei existieren zwar physisch unterschiedliche Komponenten zur Datenhaltung und -verarbeitung, aber es existiert eine zusätzliche logische Schicht, die beide Welten vereint und Möglichkeiten bietet, beide auf die gleiche Weise abzufragen. Oracle bietet solch eine Möglichkeit beispielsweise unter dem Namen Oracle Big

Data SQL, bei der Daten aus der Oracle-Datenbank, Hadoop und der Oracle NoSQL-Datenbank auf die gleiche Weise mit SQL abgefragt werden können.[170]

[170] Vgl. Harrist 2015.

5 Fallbeispiel Tatort

In diesem Kapitel wird am Beispiel Twitter gezeigt, wie durch Big Data populär gewordene Datenquellen auch in ein Data Warehouse integriert werden können. Als Untersuchungsgegenstand werden dabei die Tweets zur Kriminalserie Tatort verwendet. Die Kriminalserie existiert inzwischen seit 1970 und ist damit die am längsten laufende Krimireihe im deutschen Sprachraum.[171] Dabei werden die Kriminalfälle in den einzelnen Episoden an wechselnden Spielorten durch unterschiedliche Ermittlerteams aus verschiedenen, meist deutschen Städten aufgeklärt. Eine vereinfachte Darstellung der wesentlichen Entitäten ist in Abbildung 18 zu erkennen. Jedes Ermittlerteam besteht meist aus mehreren Ermittlern und kommt in einer oder mehreren Episoden vor. Die meisten Teams sind einem festen Ort, an dem Sie ermitteln, zugeordnet. Allerdings kann in einzelnen Episoden ihr Einsatzort auch von diesem abweichen. Einige Teams ermitteln auch an wechselnden Orten einer Region.

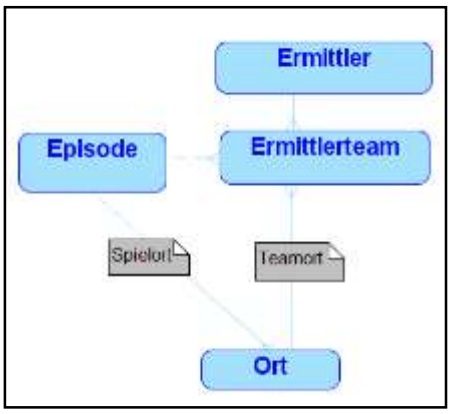

Abbildung 18: Domänenklassendiagramm Tatort

5.1 Zielsetzung

Ziel des Fallbeispiels ist die Entwicklung eines Tatort Data Warehouse. Dieses wird neben klassischen, relationalen Datenquellen auch unstrukturierte Twitter-Daten berücksichtigen. Es wird der gesamte Datenfluss von der Extraktion der Daten bis hin zur Aufbereitung und beispielhaften Visualisierung der Daten in Dashboards abgedeckt.

---

[171] Vgl. Wikipedia 2015.

In den Dashboards werden insbesondere drei Kennzahlen sowohl isoliert als auch gemeinsam hinsichtlich verschiedener Dimensionen ausgewertet. Die erste Kennzahl beinhaltet die Einschaltquoten zu den Tatortepisoden. Diese repräsentiert eine typische Kennzahl aus einem Data Warehouse. Hinzu kommen zwei Kennzahlen, die zunächst aus den unstrukturierten Tweets abgeleitet werden müssen. Zum einen ist es die Tweet-Anzahl, zum anderen die Stimmung oder auch Valenz. Die Tweet-Anzahl lässt sich leicht durch Zählen bestimmen. Die Stimmung zu ermitteln und zu quantifizieren, ist dagegen komplexer. Dort kommt eine Sentiment-Analyse (vgl. Abschnitt 4.4) zum Einsatz. Zu den Dimensionen hinsichtlich derer die Kennzahlen analysiert werden, zählen neben den Standarddimensionen eines DWH wie Ort oder Zeit auch die Tatort-Stammdaten (z.B. Episode, Ermittlerteam).

Folgende Beispielfragen sollen sich mit den Daten aus dem DWH beantworten lassen:

- Wie beliebt ist ein bestimmter Ermittler oder ein Ermittlerteam?
- Kommt es dabei zu Veränderungen über die Zeit?
- Welche Episoden kamen gut an und welche weniger?
- Sind Unterschiede zwischen der Stimmung vor Beginn einer Episode (Erwartungshaltung), während der Ausstrahlung und nach der Episode zu beobachten?
- Tritt ein Home-Bias-Effekt auf? Sind Tweets, die aus derselben Stadt stammen, in der auch der Tatort spielt, positiver geneigt als andere? Sind sonstige regionale Unterschiede in der Stimmung zu erkennen?
- Welche Meinungen scheinen besonders „mächtig“ zu sein? (Viele Retweets, Favoriten, etc.)
- Ist ein Zusammenhang zwischen der Stimmung oder der Tweet-Anzahl und den Einschaltquoten zu erkennen?

5.2 Entwurf und Umsetzung

Im Folgenden werden die Details zum Entwurf und dessen Umsetzung beschrieben. Dafür wird zunächst in Abschnitt 5.2.1 ein Gesamtüberblick über die Architektur und die eingesetzten Werkzeuge gegeben. Diese besteht aus fünf Teilbereichen, die in den darauf folgenden Abschnitten vertieft werden. Zum Abschluss des Kapitels werden in Abschnitt 5.3 die Ergebnisse diskutiert.

5.2.1 Gesamtarchitektur

Die fünf Bereiche der Gesamtarchitektur sind in Abbildung 19 dargestellt. Dabei spiegelt der Datensicherungsbereich das klassische Data Warehouse wieder. Die eingesetzten Methoden und Werkzeuge in den Bereichen Vorverarbeitung und Analyse repräsentieren die Big-Data-Komponente. Aus Sicht von Kapitel 4.5.2 bleibt das Data-Warehouse-System damit das führende System und erhält zusätzliche Daten aus der Big-Data-Komponente (vgl. Abbildung 16c). Die Architektur ist auf eine Batch-Verarbeitung ausgelegt, die die Tweets zu einer Episodenausstrahlung als einen Batch betrachtet.

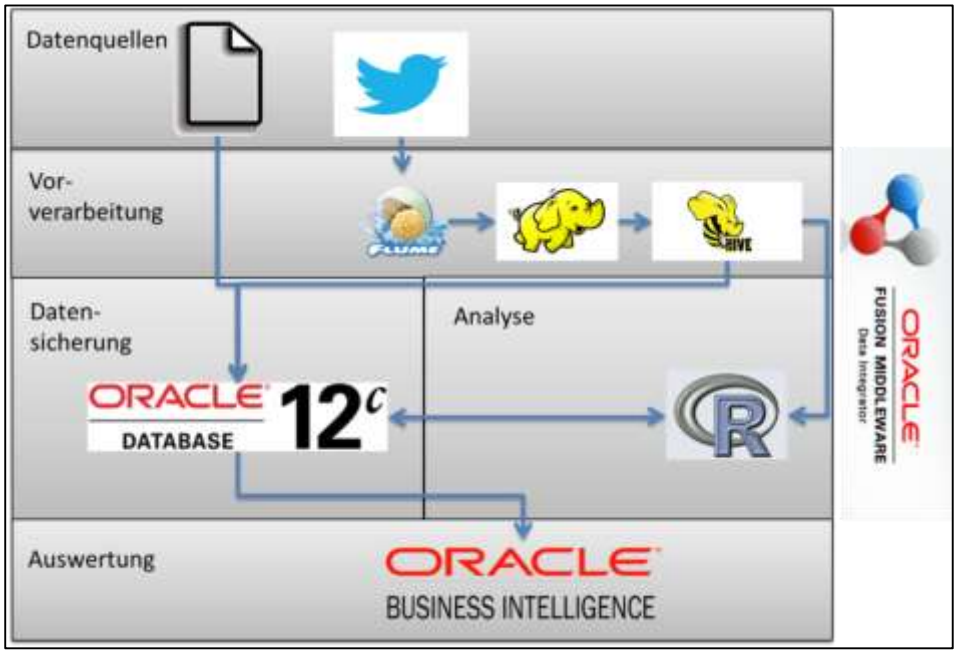

Abbildung 19: Gesamtarchitektur

Es sind zwei Arten von Datenquellen vorhanden. Zum einen eine Reihe weitestgehend strukturierter Daten (z.B. zu Tatort-Stammdaten), die in csv-Dateien vorliegen und direkt in den Datensicherungsbereich geladen werden können. Die andere Datenquelle ist Twitter, die Tweets zur Tatort-Serie liefert. Diese werden mit Hilfe des Werkzeuges Flume aus Twitter extrahiert und als Dateien im JSON-Format auf dem HDFS gespeichert. Dort werden sie mittels Hive in eine Tabellenstruktur überführt und für eine Sentiment-Analyse mit R vorbereitet.

Anschließend werden die Tweet-Texte mit Hilfe eines speziellen Konnektors zum einen direkt in R für die Analyse importiert und zum anderen zusammen mit weiteren Attributen (z.B. Ort, Uhrzeit, Nutzer) der Twitter-Schnittstelle in die Daten-

sicherungsschicht geladen. Datensicherung und Analyse sind in Abbildung 19 durch eine bidirektionale Verbindung miteinander verbunden. Dies bedeutet, dass aus Sicht von R (Analyse) das Data Warehouse gleichzeitig eine Quelle als auch das Ziel darstellt. Es werden von dort insbesondere ein Sentiment-Lexikon und Kontextinformationen, die in der Sentiment-Analyse verwendet werden, bezogen. Gleichzeitig werden die R-Analyse-Ergebnisse in der Datensicherungsschicht in strukturierter Form persistiert.

Innerhalb des Data Warehouse finden weitere Schritte zur Aufbereitung und Anreicherung der Daten statt. Der Großteil dieser Verarbeitung ist mit dem ETL-Werkzeug Oracle Data Integrator (ODI) umgesetzt. Ebenso geschehen die Gesamtkoordination der Verarbeitungen sowie die Planung der Ladezeitpunkte dort.

Schließlich werden die Daten mit Hilfe der Oracle Business Intelligence Enterprise Edition (OBIEE) in Dashboards dargestellt.

Die Entwicklungsumgebung und die Auswahl der eingesetzten Werkzeuge sind durch die Entscheidung geprägt, eine Suite-Lösung zu verwenden und keine Umgebung nach dem Ansatz Best of Breed aufzubauen. Zentrale Vorteile sind dadurch ein geringerer Einrichtungsaufwand und eine bessere Kompatibilität zwischen den unterschiedlichen Werkzeugen. So basiert die Umgebung auf einer virtuellen Maschine, die unter dem Namen Oracle Big Data Lite[172] durch Oracle bereitgestellt wird. Sie simuliert die Oracle Big Data Appliance, welche ein Gesamtsystem aus Hard- und Software zur Lösung von Big-Data-Problemstellungen ist.

Die virtuelle Maschine enthält unter anderem eine Cloudera Hadoop Distribution in der Version CDH4.5, eine Oracle 12c Datenbank sowie den Oracle Data Integrator 12cR1. Hadoop wurde bereits in Abschnitt 4.2.2 ausreichend vorgestellt. Die Oracle 12c Datenbank ist die aktuelle Version der relationalen Datenbank von Oracle. Beim Oracle Data Integrator handelt es sich um Oracle´s führendes ETL-Werkzeug. In diesem werden die Datenflüsse zum Beladen einer Tabelle in sogenannten Mappings abgebildet. Die Ausführung dieser kann über Packages organisiert werden. Ein weiteres bereits vorhandenes Werkzeug ist die Oracle R-Distribution in der Version 3.0.1. Dies ist eine kommerzielle Version des Statis-

[172] Vgl. Oracle 2015.

tik-Werkzeuges R, die sich insbesondere durch zusätzliche Funktionen zur Interaktion mit der Oracle-Datenbank von der freien R-Version abhebt. Diese befinden sich in einem Paket Oracle R Enterprise (ORE). So wird zum Beispiel ermöglicht aus R heraus Funktionen in der Datenbank auszuführen oder auch aus der Datenbank R-Funktionalitäten zu nutzen. Zudem bietet Oracle eine Reihe an Big-Data-Konnektoren, die zum Datenaustausch zwischen Hadoop, R und der Oracle-Datenbank verwendet werden können.

Die OBIEE ist nicht Bestandteil der virtuellen Maschine, aber passend zu den anderen Komponenten das Werkzeug bzw. der Werkzeugkasten zur Darstellung von BI-Auswertungen von Oracle. Es ist in der Version 11g auf einem separaten Host installiert. Die OBIEE enthält verschiedene Einzelwerkzeuge, um sowohl „pixelperfekte“ (Pixel Perfect) Berichte mit dem BI Publisher oder auch Dashboards mit Oracle Answers zu erstellen. Außerdem wird noch das Zusatzprogramm MapBuilder für die OBIEE verwendet. Dieses dient zur Gestaltung von Landkarten für Berichte mit einer Kartendarstellung. Da die OBIEE auch eine direkte Anbindung auf Hive ermöglicht, wäre auch die Architekturvariante aus Abbildung 16b denkbar mit einer Konsolidierung aller Daten in der OBIEE. Allerdings wären dabei die Abfragen zu langsam, da für nahezu alle Abfragen ein MapReduce-Batch-Prozess ausgeführt werden müsste.

5.2.2 Extraktion

In diesem Abschnitt wird die Extraktion der Twitter-Daten fokussiert. Dabei werden die relevanten Tweets mit Hilfe der Twitter-Streaming-API während der Episodenausstrahlung „mitgeschnitten“. Zur Identifikation der relevanten Tweets wird das Hashtag *#tatort* als Suchschlüssel verwendet. Zudem werden nur deutschsprachige Tweets gefiltert. Die Verwendung der Streaming-API bietet sich für den gegeben Anwendungsfall an, da diese automatisiert und in Echtzeit alle Tweets zum angegeben Suchbegriff sammelt. Dadurch wird auf einfache Art und Weise sichergestellt, dass möglichst viele relevante Tweets erfasst werden. Mit der REST-API könnte zwar im Nachhinein auf dieselben Tweets zugegriffen werden. Jedoch wären die Filterbedingungen, um auf den relevanten Zeitraum einzuschränken, vergleichsweise umständlich. Außerdem würden für Stoßzeiten die Einschränkungen bezüglich Anzahl der Anfragen und der Tweets an ihre Grenzen gelangen.

Zur Verarbeitung des Twitter-Streams wird Flume eingesetzt. Wie bereits in Abschnitt 4.2.2.3 erläutert, muss in diesem ein sogenannter Agent konfiguriert werden, der aus einer Quelle (Source), einem Zwischenspeicher und dem Ziel (Sink) besteht. Als Quelle wird ein Java-Programm genutzt, welches die Streaming-API von Twitter als benutzerdefinierte Quelle ermöglicht. Solch ein Programm wird von Cloudera bereitgestellt und hier verwendet.[173] Dieses nutzt die öffentliche Java-Bibliothek Twitter4J[174]. Als Zwischenspeicher reicht der interne Speicher von Flume aus, da keine aufwändigen Operationen oder Transformationen während der Extraktion notwendig sind. Das JSON-Format, in dem Twitter neben den Tweet-Texten noch zahlreiche weitere Attribute liefert (vgl. Abschnitt 4.3.3), wird zunächst für die Speicherung im Ziel HDFS beibehalten. Die Ermittlung des Speicherpfades geschieht dynamisch über das Datum. Tweets zur Tatort-Episode vom 01.02.2015 werden zum Beispiel auf dem HDFS unter dem Pfad /flume/tatort/2015/02/01 abgelegt. So hat jede Episodenausstrahlung einen eigenen Ordner. Dies bringt Vorteile für die spätere Verarbeitung in Hive und auch ein mögliches Datenlebenszyklusmanagement. Die komplette Flume-Konfiguration findet sich im Anhang A.

Die Tweets werden nicht nur während der Spielzeit der Episoden gesammelt, sondern mit einer gewissen Vor- und Nachlaufzeit. Abbildung 20 zeigt das typische Zusammenspiel zwischen Episode (Sendetag), Streaming-Zeiten und dem Speicherort. Durch die Tweets, die vor Sendebeginn gesammelt werden, sind möglicherweise Aussagen zu den Zuschauererwartungen möglich. Durch die Nachlaufzeit sollen auch Diskussionen und Meinungen erfasst werden, die erst im Nachgang publiziert werden. Weiterhin ist hervorzuheben, dass nur Erstausstrahlungen, die i.d.R. sonntagabends gezeigt werden, Betrachtungsgenstand sind.

---

[173]Vgl. Natkins 2012, S. 2.

[174] Http://twitter4j.org/en/index.html.

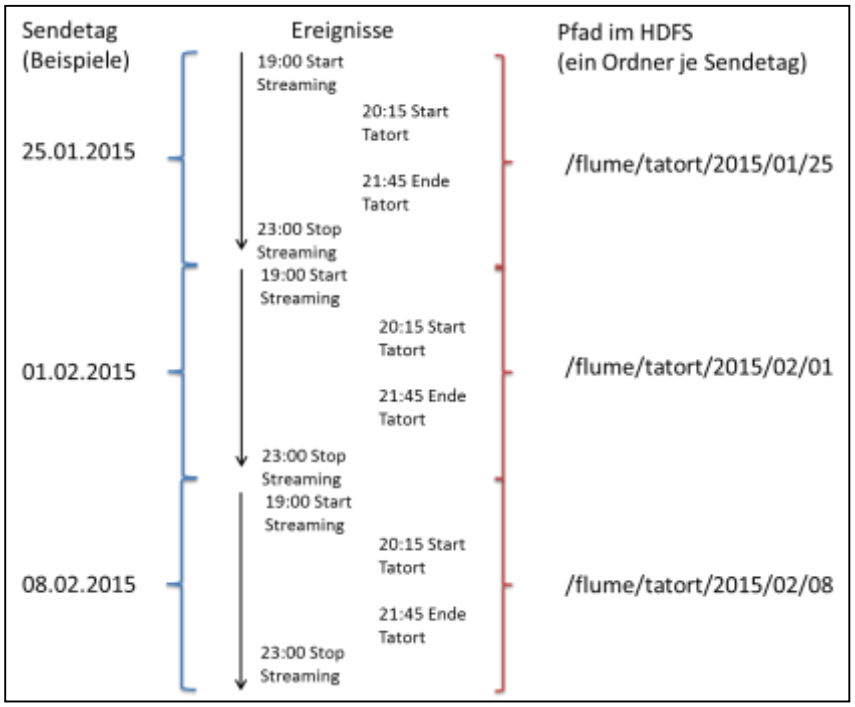

Abbildung 20: Extraktion

5.2.3 Vorverarbeitung

In dieser Schicht werden die Daten zunächst vom JSON-Format in eine Tabellenstruktur überführt und gefiltert. Eine Stärke von Hive ist, dass es insbesondere bei der Verarbeitung von semi-strukturierten Daten sehr flexibel ist. So kann mit Hilfe eines SerDe (Serializer/Deserializer) in Hive bekannt gemacht werden, wie Daten im JSON-Format einzulesen sind. Dies ermöglicht es, dass mit Hilfe einer „external Table“ direkt auf die Tweets, die im JSON-Format auf dem HDFS liegen, zugegriffen werden kann. Ein fertiger JSON-SerDe für Hive kann ebenfalls von Cloudera[175] heruntergeladen und in Hive importiert werden. Zu beachten ist, dass dieser mit einem einfachen Import der zugehörigen Jar-Datei nur für die aktuelle Session verfügbar ist. Um diesen global (z.B. auch für Abfragen aus R) und dauerhaft verfügbar zu machen, muss dieser in einer entsprechenden Hive Konfigurationsdatei (hive-site.xml mit dem Parameter hive.aux.jars.path) bekannt gemacht werden.

Das Hive-Datenmodell ist vereinfacht in Abbildung 21 dargestellt.

[175] Vgl. Natkins S. 3.

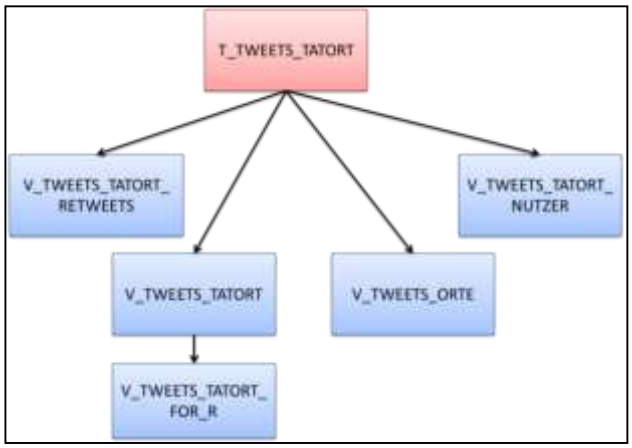

Abbildung 21: Hive-Datenmodell

Es besteht aus einer Tabelle T_TWEETS_TATORT. Bei dieser handelt es sich um die externe Tabelle, die direkt auf das HDFS zugreift und die JSON-Daten in eine Tabellenstruktur überführt. Somit werden die Daten nicht erneut innerhalb von Hive gespeichert. Für einen optimierten Zugriff ist diese Tabelle nach dem Sendetag partitioniert. Für diese Partitionierung ist die im vorangegangen Kapitel beschriebene Erstellung von eigenen Verzeichnissen je Sendetag notwendig. Inhaltlich ist dies eine flache Tabelle, die den Großteil der Attribute enthält, die durch die Twitter-API zur Verfügung stehen. Um flexibel für Erweiterungen zu sein, enthält die Tabelle mehr Attribute als dies für aktuelle Analysen notwendig ist. Lediglich einige technische oder redundante Attribute werden nicht aus der JSON-Datei übernommen. Im folgenden Vorverarbeitungsschritt wird diese sehr breite Tabelle durch vier Views nach Informationen über die Tweets selbst (V_TWEETS_TATORT), über die Twitter-Nutzer (V_TWEETS_TATORT_ NUTZER), über die Orte, von denen aus getweetet wird bzw. die mit einem Tweet verknüpft sind (V_TWEETS_ORTE), und über die Retweets (V_TWEETS_TATORT_RETWEETS) gegliedert.

Diese vier Views stellen auch die Schnittstelle zur Oracle-Datenbank dar. Darüber hinaus existiert noch eine auf die View V_TWEETS_TATORT kaskadierende View. Diese stellt die Tweet-Texte konkateniert mit den dazugehörigen IDs für die Analyse in R bereit. Diese erleichtert den Import nach R.

Die DDL-Befehle (Data Definition Language) zu allen Hive-Objekten befinden sich im Anhang B.

5.2.4 Datensicherung

Der Bereich zur Datensicherung entspricht einem relationalen Data Warehouse. Dieses untergliedert sich in die drei Schichten Stage, Core und Mart (vgl. Abbildung 22).

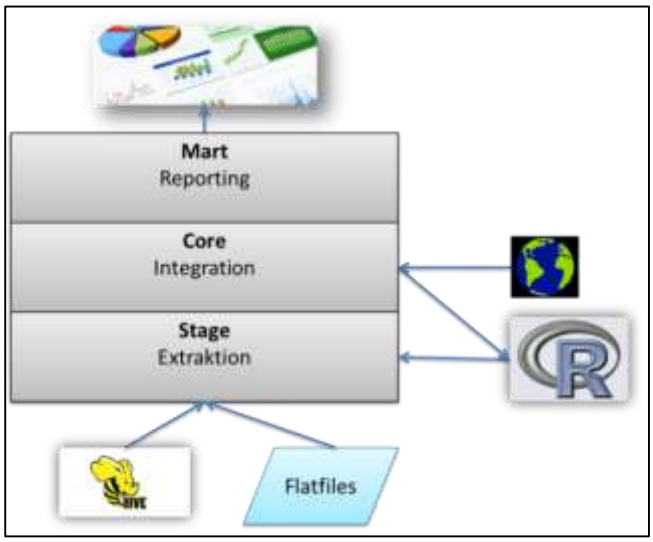

Abbildung 22: Tatort Data Warehouse

Dabei ist Stage die Extraktions- bzw. Dateneingangsschicht. In diese werden die Daten aus R, Hive und den Flatfiles zur aktuellen Verarbeitung zwischengespeichert. Die Datenstruktur entspricht der aus der jeweiligen Quelle. In der Core-Schicht werden insbesondere die Daten von der aktuellen Verarbeitung, die sich in der Stage-Schicht befinden, in die Daten der vorangegangen Verarbeitungen integriert. Außerdem werden in dieser Schicht auch Referenzdaten, Steuerinformationen und Parametertabellen, die auch für die Analyse in R benötigt werden, gespeichert, gepflegt und bereitgestellt. Die Datenstruktur des Core ist nahe an einer dritten Normalform. Die Mart-Schicht bedient sich ausschließlich aus dem Core. Dort werden die Daten speziell für die Berichterstellung vorbereitet und bereitgestellt. Dazu werden die Daten in ein Star-Schema überführt. Physisch befinden sich alle drei Schichten im selben (Tatort-)Datenbank-Schema, sind aber logisch durch ihre Namenskonventionen unterscheidbar.

Im Folgenden wird eine Beschreibung der Quellen (Abschnitt 5.2.4.1), ein Überblick über die Verarbeitungsflüsse (Abschnitt 5.2.4.2) und anschließend der Datenmodelle sowie der Datenladeprozesse auf Tabellenebene gegeben. Letzteres ist

unterteilt nach den logischen Schichten Stage (Abschnitt 5.2.4.3), Core (Abschnitt 5.2.4.4) und Mart (Abschnitt 5.2.4.5).

5.2.4.1 Quellen

Das Data Warehouse bezieht die Daten aus insgesamt drei Schnittstellen (Hive, Flatfile und R). Die Twitter-Daten aus **Hive** können direkt mit dem ODI abgegriffen werden. Die Struktur wurde bereits in Abschnitt 5.2.3 erläutert.

Über die **Flatfile-Schnittstelle** wird zum einen das vorgefertigte Sentiment-Lexikon SentiWS bezogen, welches bereits in Abschnitt 4.4.2 behandelt wurde. Verfügbar sind die Daten über je eine Datei mit positiven und negativen Begriffen. Zum anderen umfasst diese Schnittstelle Daten zu den Tatort-Episoden, Ermittlern und Teams. Diese Informationen können in einem produktiven Betrieb möglicherweise direkt aus einer anderen (operativen) Datenbank bezogen werden. Diese wird hier durch diese Flatfile-Schnittstelle simuliert, die zwei csv-Dateien liefert. Dabei enthält die eine Datei verschiedene Informationen (z.B. Titel, Ausstrahlungstermin, Einschaltquote) zu einer einzelnen Episode[176] (Episoden.csv) und die andere einen Überblick über die Ermittler[177] und ihre Zuordnung zu einem Ermittlerteam (Ermittlerteams.csv). Im Anhang H sind exemplarisch die Inhalte zur Episoden.csv hinterlegt.

Bei der **R-Schnittstelle** werden die Analyse-Ergebnisse durch R direkt in den Stage-Bereich der Oracle-Datenbank geschrieben. Dabei werden zu jedem Tweet (Tweet-Id) neben einem Sentiment-Wert auch die Schlüssel der Entitäten/Kontexte, die diesem Tweet zugeordnet werden konnten, angegeben. Beide Attribute sind optional.

Zudem ist in Abbildung 22 eine Weltkugel zu sehen, die direkt mit dem Core verknüpft ist. Diese stellt geographisches Kartenmaterial dar. Herkunft ist ein durch Oracle bereitgestellter Export (World Sample Data Bunde)[178] mit Ortsdaten zur Erstellung von Karten. Es sind also insbesondere neben Ortsbezeichnungen auch Ortskoordinaten vorhanden. Diese Ortsdaten werden aus zwei Gründen verwendet. Zum einen dienen sie als Referenzdaten zur Datenqualitätssicherung im Core

---

[176] Quelle: Wikipedia 2015.

[177] Quelle: Wikipedia 2015.

[178] Quelle: Oracle 2010.

und zum anderen zur Kartendarstellung in den späteren Berichten. Der Export ist einmalig in ein eigenes Datenbankschema in der Datenbank des Tatort-DWH importiert. Daher kann auf eine vorherige Extraktion in die Stage-Schicht verzichtet werden.

5.2.4.2 Verarbeitungsflüsse

Die Verarbeitung aller Daten für das Tatort-DWH verteilt sich auf vier Ladeketten, die die Ausführung steuern. Diese sind dafür zuständig die zugehörigen Daten von der Quelle bis zu ihrem jeweiligen Ziel, was meistens die Mart-Schicht ist, zu laden. Diese Ladeketten sind im ODI in Packages umgesetzt und nach Themenbereichen bzw. Datenherkunft getrennt. Einen Überblick gibt Tabelle 3.

Name	Inhalt	Ausführungszeitpunkt/Intervall
PCK_REFRESH_ STAMMDATEN	Aktualisierung geographischer und zeitlicher Referenz-Daten	Initial und nach Bedarf.
PCK_REFRESH_ WOERTERBUECH	Aktualisierung der Wörterbücher	Initial und nach Bedarf.
PCK_REFRESH_TATORT	Aktualisierung der Tatort-Daten zu Episoden, Teams und Ermittlern	Wöchentlich nach Erfassung der Einschaltquoten.
PCK_REFRESH_TWEETS	Verarbeitung neuer Daten von Twitter	Wöchentlich nach dem Tweet-Sammeln.

Tabelle 3: ODI-Packages

Die beiden oberen Ladeketten werden nach dem initialen Laden vergleichsweise selten bzw. unregelmäßig ausgeführt, da dort wenige Änderungen zu erwarten sind. Die zeitlichen Daten müssen nur geladen werden, wenn ein initiales Laden notwendig ist (z.B. zur Wiederherstellung von Daten, nach einer Änderung in der Struktur, nach dem Ausrollen in einer neuen Umgebung, etc.). Eine Aktualisierung der geographischen Referenz-Daten wird darüber hinaus noch dann benötigt, wenn Daten zu einem weiteren Land in den Twitter-Daten auftauchen. Um Hardware-Ressourcen zu sparen, werden diese nur nach Bedarf im DWH verfügbar gemacht. Das Hinzukommen weiterer Länder ist zwar grundsätzlich möglich, da aber nur deutschsprachige Tweets extrahiert werden, eher die Ausnahme. Neben wenig zu erwartenden Änderungen haben die zeitlichen und geographischen Daten ebenfalls die Gemeinsamkeit, dass sie nicht spezifisch für das Tatort-DWH sind, sondern einen eher generischen Charakter haben und auch in einem anderen

DWH vorkommen könnten. Aus diesem Grund werden sie zusammen in einer Ladekette PCK_REFRESH_STAMMDATEN verarbeitet. Wörterbuchänderungen sind meist mit manuellen Anpassungen verbunden, wie z.B. dem Pflegen neuer Einträge in einer Parametertabelle oder Nutzung einer neuen Version des vorgefertigten Wörterbuches. Da solche Änderungen unregelmäßig aber bewusst stattfinden und kurzfristig nutzbar sein müssen, ist es sinnvoll diese mit einer eigenen Ladekette (PCK_REFRESH_WÖRTERBUCH) unmittelbar nach solchen Anpassungen im DWH schnell aktualisieren zu können.

Regelmäßig müssen dagegen die Ladeketten zu den Tatort- und Twitter-Daten ausgeführt werden. Beide sind von der Episodenausstrahlung abhängig, die in der Regel wöchentlich stattfindet. Die Tatort-Daten aus der Flatfile-Schnittstelle werden durch das Package PCK_REFRESH_TATORT nach jeder Episodenstrahlung aktualisiert, da diese die Episodendaten inklusive der Einschaltquoten enthalten.

Die Twitter-Daten können sobald das „Mitschneiden“ der Tweets abgeschlossen ist mit dem Package PCK_REFRESH_TWEETS verarbeitet werden. Diese Ladekette umfasst den größten Bereich der Gesamtverarbeitung. Vereinfacht ist die Verarbeitung dieser Ladekette in Abbildung 23 dargestellt.

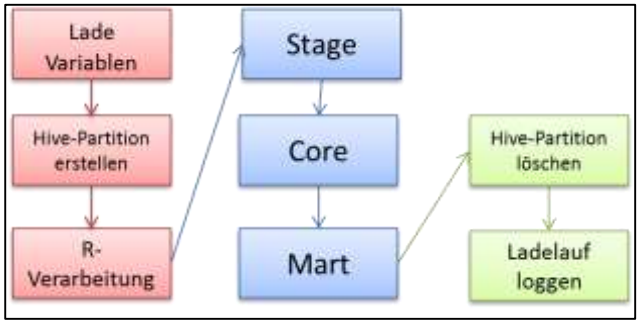

Abbildung 23: Ladekette Twitter-Daten

Zunächst wird mit Hilfe einer Logging-Tabelle und einer View aus dem Core des DWH der zu verarbeitende Sendetermin mit dem zugehörigen Pfad im HDFS ermittelt und in Variablen im ODI gespeichert. Diese werden genutzt, um die entsprechende Hive-Partition für die aktuelle Verarbeitung zu erstellen. In einer weiteren Variable wird zudem die Information gespeichert, ob die Tweets in der Sommer- oder Winterzeit erstellt wurden. Da das Erstellungsdatum der Tweets in der UTC-Zeit (koordinierte Weltzeit) empfangen wird, muss dieses in UTC+2 für die Sommerzeit bzw. UTC+1 für die Winterzeit umgewandelt werden. Im An-

schluss wird die R-Verarbeitung über ein Betriebssystem-Kommando gestartet. Da in der gegebenen Entwicklungsumgebung R, Hive und ODI physisch auf einem Host liegen und sich einen Arbeitsspeicher teilen, beginnt erst im Anschluss die eigentliche DWH-Verarbeitung mit der Stage-Beladung. In einem regulären Betrieb, in dem die genannten Applikationen auf unterschiedlichen Hosts laufen oder mehr Arbeitsspeicher zur Verfügung steht, kann die R-Verarbeitung auch mit der Stage-Beladung parallelisiert werden, um die Gesamtlaufzeit zu reduzieren. Für die Core und Mart-Beladung ist es jedoch wichtig, dass die Beladung der vorangegangen Schicht komplett abgeschlossen ist. Nach der eigentlichen Verarbeitung wird wieder die zuvor erstellte Hive-Partition entfernt. Somit ist die Hive-Schnittstelle leer, wenn keine Bearbeitung stattfindet. Zum Abschluss wird der erfolgreiche Ladelauf in der eingangs erwähnten Logging-Tabelle festgehalten.

5.2.4.3 Stage

Das Datenmodell der Stage-Schicht ist in Abbildung 24 zu sehen. Es umfasst neun Tabellen und zwei Views. Es enthält keine Primärschlüssel, Fremdschüssel oder anderen Datenbank-Constraints.

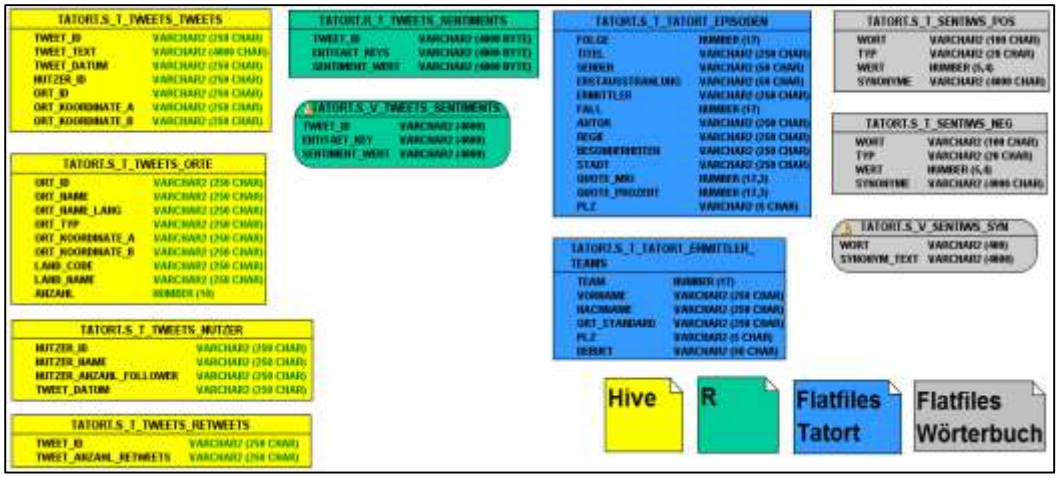

Abbildung 24: Datenmodell Stage

In Abbildung 24 sind die Objekte nach der Herkunft ihrer Daten gruppiert. Die vier Tabellen, die aus Hive ihre Daten beziehen, sind der Pendant zu den zuvor vorgestellten Hive Schnittstellen-Views (vgl. Abschnitt 5.2.3). Geladen werden diese jeweils über ein eigenes ODI-Mapping, welches die Zieltabellen leert und alle Daten aus den Quell-Views einfügt. In die Tabelle R_TWEETS_SENTIMENTS werden die Ergebnisse der Sentiment-Analyse durch R gesichert. Da mehrere Entitäten-Schlüssel zu einem Tweet zugeordnet werden können, ist

das Feld ENTITAET_KEYS nicht atomar. In der View S_V_TWEETS_ SENTIMENTS wird daher für jeden Entitäten-Schlüssel ein separater Datensatz erzeugt. Außerdem werden Tweets, denen kein Sentiment-Wert zugeordnet werden konnte, herausgefiltert.

Die beiden csv-Dateien mit den Tatort-Daten werden in die beiden Stage-Tabellen S_TATORT_EPISODEN und S_TATORT_ERMITTLER_TEAMS geladen, die ebenfalls zuvor geleert werden. Gleiches geschieht zunächst auch mit den beiden Dateien zum Wörterbuch SentiWS. Da sich jedoch die Synonyme bzw. unterschiedlichen Wortformen zu einem Hauptwort ähnlich wie zuvor schon die Entitäten-Schlüssel in einem nicht atomaren Feld befinden, wird ebenfalls in einer separaten View (S_V_SENTIWS_SYN) für jedes Synonym ein eigener Datensatz erzeugt.

5.2.4.4 Core

Auf die Core-Schicht entfallen neben der Integration der Stage-Daten noch zwei weitere Aufgaben. Zum einen beinhaltet sie Steuer- und Parametertabellen und zum anderen bietet sie eine Schnittstelle an, in der sie Wörterbücher und Kontextinformationen für R bereitstellt. Technisch sind im gesamten Core-Datenmodell Primär- und Fremdschlüssel angelegt. Zudem werden künstliche Schlüssel verwendet. Zu erkennen sind diese an dem Suffix „KEY“ im Spaltennamen. Im Anhang E ist ein detaillierter Überblick über alle Mappings mit ihren Quell- und Zieltabellen. Daher werden an dieser Stelle nur die wichtigsten Aspekte des Datenmodells und der Beladungen beschrieben.

Im Folgenden wird zunächst der Teil des Core-Datenmodells betrachtet, in dem die Stage-Daten integriert werden. Zum Laden dieser Tabellen wird in den ODI-Mappings die Lademethode inkrementelles Update eingesetzt. Bei dieser werden nicht vorhandene Datensätze eingefügt und vorhandene Datensätze aktualisiert. Generell sind hier drei Bereiche zu unterscheiden (vgl. Abbildung 25). Im oberen Bereich befinden sich in den blau markierten Tabellen die Informationen zu den Tweets, unten links sind grün dargestellt die Tatort-Informationen und unten rechts befinden sich in grau neutrale geographischen Referenzdaten.

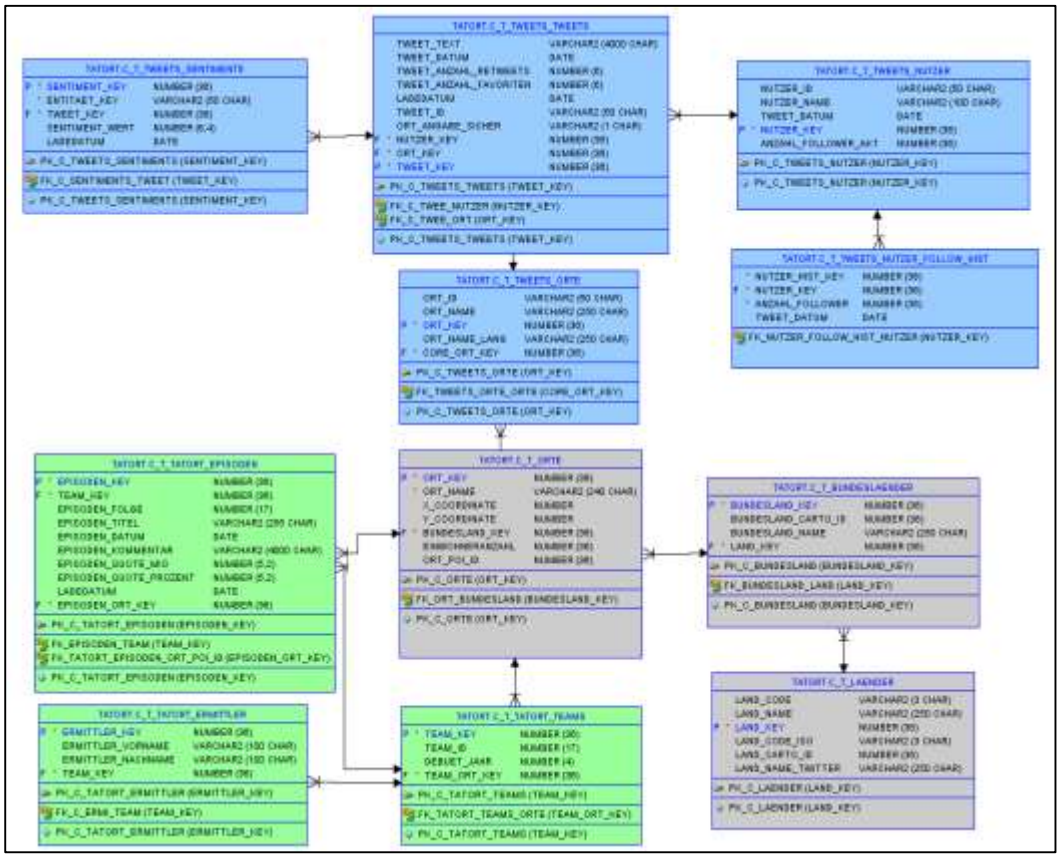

Abbildung 25: Datenmodell Core

Die Daten für die drei Tabellen mit den geographischen Daten kommen nicht über die Stage-Schicht, sondern werden direkt über eine View aus dem in Abschnitt 5.2.4.1 beschriebenen Import in den Core geladen. Diese werden als Referenzdaten benötigt, da sowohl die Tweet-Daten als auch die Tatort-Daten Ortsinformationen enthalten. Dabei haben die Ortsinformationen der Tweet-Daten aus Datenqualitätssicht einige Nachteile. So sind diese zum einen nicht immer auf derselben Granularität. In den meisten Fällen ist die Information auf Stadt-Ebene. Jedoch kommt es ebenfalls vor, dass diese auf Stadtteil-, Kreis- oder Bundeslandebene vorhanden ist. Hinzu kommt die Schreibweise und Sprache, in der die Ortsangabe geliefert wird. Zumeist wird es in der Landesssprache des Ortes (im aktuellen Fallbeispiel deutsch), aber teilweise auch in Englisch wiedergegeben. So ist beispielsweise für die Stadt München neben der deutschen Schreibweise auch „Munich“ in den Daten zu finden. Um diese Unterschiede innerhalb der Twitter-Daten und auch zu den Tatort-Daten zu harmonisieren, werden Referenz-Daten verwendet. Für die Tatort-Daten geschieht die Zuordnung alleine über den Namen. Bei den Twitter-Daten ist eine Zuordnung über den Namen aufgrund der genannten

Nachteile sehr schwierig. Dafür haben die Twitter-Daten den Vorteil, dass diese neben den Ortsbezeichnungen ebenfalls auch die Ortskoordinaten enthalten. So werden diese für eine entsprechende Verknüpfung genutzt. Die Twitter-Daten werden jedoch verwendet, um die Referenzdaten über den Ländercode auf die relevanten Länder zu filtern. Dadurch kann das Datenvolumen beträchtlich reduziert werden, da auch nur zu diesen Ländern Ortsinformationen auf niedrigerem Level gespeichert werden. Dabei wird die Länderbezeichnung in der jeweiligen Landessprache, die in den Twitter-Daten enthalten ist, neben der englischen Bezeichnung, die in den Referenzdaten enthalten ist, gespeichert. Daher ist in der Relation der Referenzdaten auch das Attribut LAND_NAME_TWITTER zu finden.

Für die Twitter-Daten sind fünf Tabellen im Core vorgesehen. Diese haben sich auf den ersten Blick nur gering zum Stage-Bereich verändert. Dennoch finden einige Transformationen statt. Neben einigen Datentyp-Anpassungen, der Sicherstellung der referentiellen Integrität und der Verknüpfung mit den Ortsreferenzdaten, wird außerdem die Follower-Anzahl der Twitter-Nutzer historisiert.

Die Tatort-Daten werden im Core auf drei Tabellen (Episoden, Ermittler und Teams) verteilt. Dadurch werden Redundanzen entfernt. Zu beachten ist, dass sowohl eine Episode als auch ein Team eine Ortsinformation enthält. Die Ortsinformation des Teams entspricht dem Ort, in dem das Team üblicherweise ermittelt. Da es davon jedoch auch Abweichungen geben kann, wie bereits in Abbildung 18 gezeigt wurde, kann einer Episode eine abweichende Ortsinformation zugeordnet werden.

Zum Bewältigen der Aufgaben hinsichtlich der Verwaltung der Steuer- und Parameterinformationen und der Bereitstellung dieser für R dienen im Core die in Abbildung 26 dargestellten Datenbankobjekte.

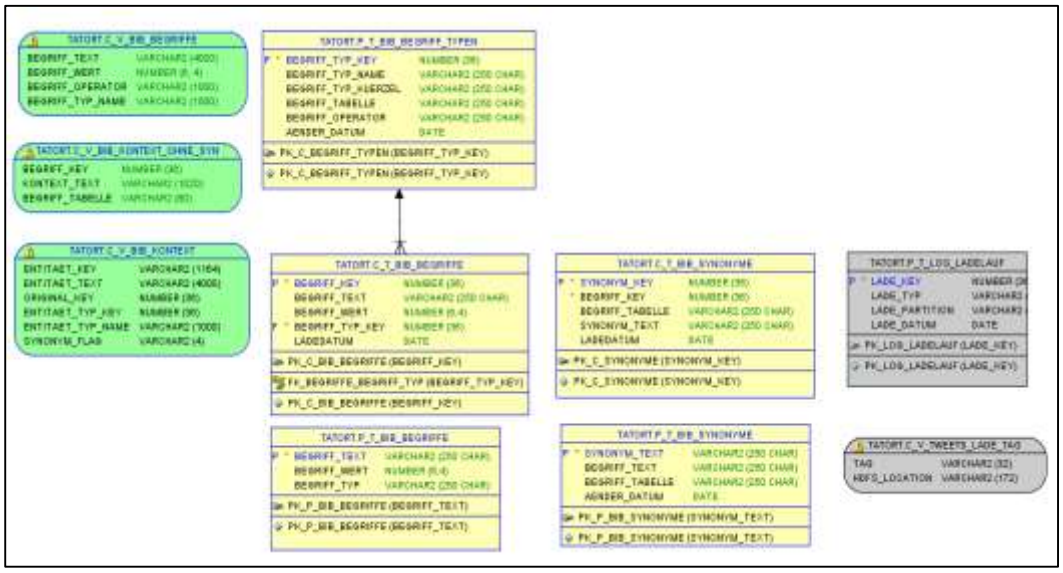

Abbildung 26: Parameter-/Steuertabellen

Links ist die Schnittstelle zu R in grün und in der Mitte der Parameter- und Steuerbereich in gelb zu erkennen. Rechts sind zwei weitere Objekte zur Unterstützung der Überwachung und des Ausführens der Ladeprozesse zu sehen. In der Tabelle P_T_LOG_LADELAUF loggt das in Abschnitt 5.2.4.2 vorgestellt Package PCK_REFRESH_TWEETS jede erfolgreiche Ausführung und aus der View C_V_TWEETS_LADE_TAG erhält es den als nächstes zu verarbeitenden Ausstrahlungstag mit dem dazugehörigem Pfad im HDFS. Dieser Eintrag ist standardmäßig das älteste Sendedatum aus den Tatort-Stammdaten, welches in der Log-Tabelle noch nicht verarbeitet wurde. Durch eine entsprechende Manipulation dieser Tabelle kann daher leicht auch eine Verarbeitung zu einer anderen Ausstrahlung bzw. eine Neuverarbeitung aller Ausstrahlungen erzwungen werden.

Spaltenname	Beschreibung	Beispiele
BEGRIFF_TEXT	Suchbegriff, Phrase	super; schlecht; kein; den; @
BEGRIFF_WERT	Wenn der Begriff eine positive/negative Neigung hat, ist die Stärke in diesem Wert quantifiziert. Wert zwischen -1 und 1.	0,5012; -0,7706; -1;0
BEGRIFF_OPERATOR	Operation, die bei Auftreten des Suchbegriffes durchgeführt werden soll	ADD; SUBTRACT; MULTIPLY; REMOVE
BEGRIFF_TYP_NAME	Bezeichnung für den Begriffstyp oder auch das Wörterbuch. Hierrüber wird ein Begriff einem Wörterbuch zugeordnet.	PHRASE_POS; PHRASE_NEG; INCREASER; DECREASER; NEGATION; STOPWORD; STOPPATTERN

Tabelle 4: Wörterbücher

Die View C_V_BIB_BEGRIFFE stellt die Wörterbücher für R bereit. Die Attribute dieser View sind in Tabelle 4 beschrieben. Aus den Beispielen geht bereits hervor, dass verschiedene Wörterbücher vorhanden sind, die jeweils einem Begriffstyp zugeordnet sind. Die verschiedenen Begriffstypen werden im Parameter-/Steuerbereich in der Tabelle P_T_BIB_ BEGRIFF_TYPEN manuell gepflegt. Die eigentlichen Begriffe stammen zum einen aus dem SentiWS-Wörterbuch, welches sich im Stage-Bereich befindet und zum anderen können zusätzliche Begriffe manuell in der Parametertabelle P_T_BIB_BEGIRFFE gepflegt werden. Diese werden zusammen in die Tabelle C_T_BIB_BEGRIFFE geladen. In der View C_V_BIB_BEGRIFFE sind ebenfalls Synonyme berücksichtigt. Die unterschiedlichen Wortformen der Wörterbücher aus dem Stage werden ebenfalls als solche behandelt. Außerdem können ebenfalls über eine weitere Parametertabelle (P_T_BIB_SYNONYME) zusätzliche definiert werden. Beides zusammen wird in der Tabelle C_T_BIB_SYNONYME zusammengeführt.

Die View C_V_BIB_KONTEXT_OHNE_SYN bereitet die Tatort-Daten (Episoden, Ermittler, Teams) in einer Art Entität-Schlüssel-Wert-Struktur für R auf. Der Entität-Typ ist dabei die Entität, der Entität-Key der Schlüssel und die übrigen Attribute der Wert. Auch diesen Daten können über die zuvor vorgestellte Parametertabelle Synonyme zugeordnet werden. Berücksichtigt werden diese in einer weiteren View C_V_BIB_KONTEXT (vgl. Tabelle 5).

Spaltenname	Beschreibung	Beispiele
ENTITAET_KEY	Eindeutiger Identifizierungs-schlüssel der Kontextinforma-tion	EP_159; ER_447; TE_160
ENTITAET_TEXT	Suchtext zur Kontextinforma-tion	hydra; schenk; koeln
ORIGINAL_KEY	Ursprünglicher Schlüssel in den Tatortstammdaten	159; 447; 160
ENTITAET_TYP_KEY	Schlüssel zum Entität-Typ	18; 17;16
ENTITAET_TYP_NAME	Bezeichnung zum Entität-Typ	TATORT_EPISODE; TATORT_ERMITTLER; TATORT_TEAM
SYNONYM_FLAG	Kennzeichen, ob es sich bei dem Begriff um ein Synonym zu einem anderen Begriff handelt	x; (null)

Tabelle 5: Kontextinformationen

5.2.4.5 Mart

Das Mart-Datenmodell, welches in Abbildung 27 zu sehen ist, ist mit einer Ausnahme in einem Star-Schema umgesetzt. Bei der Ausnahme handelt es sich um die Ortsinformation in den Tatort-Stammdaten (Tatort-Episoden). Diese sind nicht direkt in der dazugehörigen Dimension gespeichert, sondern als Referenz zur Ortsdimension, wie es eher in einem Snowflake-Schema üblich ist. Der Grund ist, dass die Ortsdimension eine Hierarchie ist. Daher müssten alternative alle Levels der Hierarchie mit in der Tatort-Dimension gespeichert werden oder die anderen Levels wären nicht zu den Tatort-Orten in der Auswertung verfügbar.

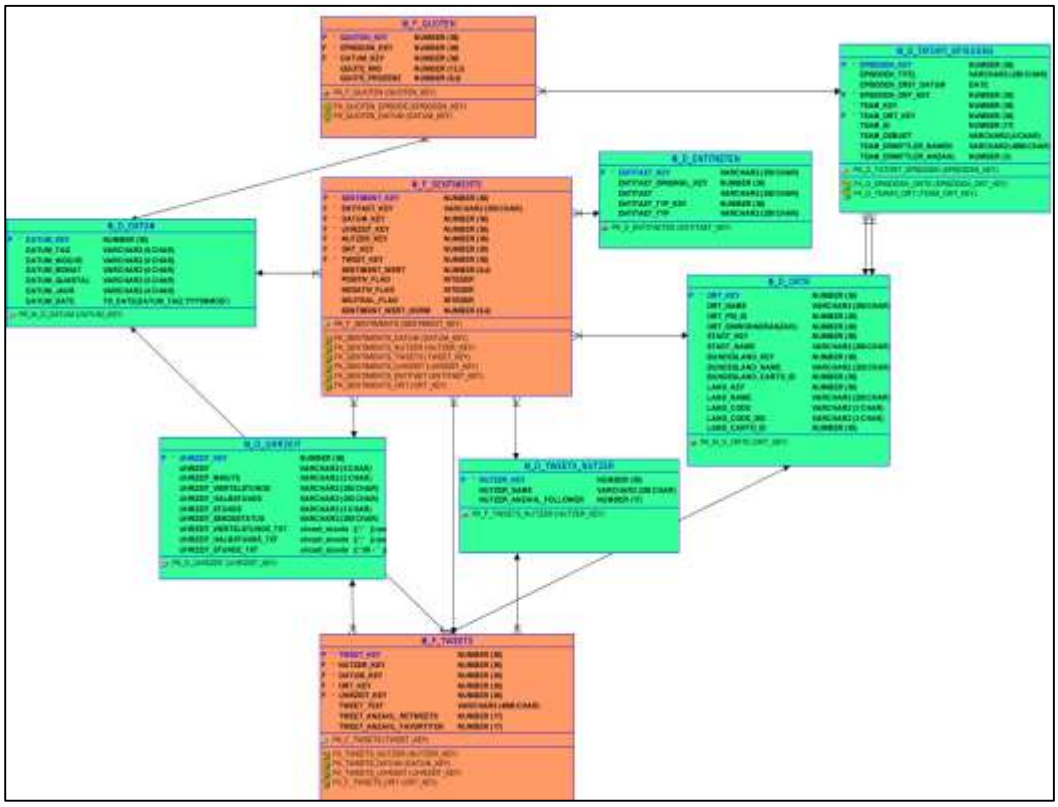

Abbildung 27: Datenmodell Mart

Insgesamt umfasst das Mart-Datenmodell drei Fakten- und sechs Dimensionstabellen. Die ODI-Mappings dazu sind ebenfalls im Anhang E aufgelistet. Zu den Einschaltquoten, Sentiments und Tweets existiert jeweils eine eigene Faktentabelle. Die Einschaltquoten werden dabei in einer absoluten und in einer prozentualen Kennzahl angegeben. Die Sentiments haben einen absoluten Sentiment-Wert, der in der Analyse-Schicht berechnet wird und den Sentiment quantifiziert. Für eine bessere Auswertung wird dieser zusätzlich auf einer Skala von -1 bis 1 normalisiert. Zudem wird jedes Sentiment mit einem Flag ausgestattet, ob dieses positiv, negativ oder neutral ist. Es wird von dem skalierten Sentiment-Wert abgeleitet. Die unteren 40% werden als negativ, die oberen 40% als positiv und die übrigen Werte als neutral kategorisiert. Diese Flags eignen sich gut zur Ermittlung der Gesamtanzahl an positiven oder negativen Sentiments. Die Faktentabelle zu den Tweets wird in erster Linie als eine faktenlose Faktentabelle zur Ermittlung der Tweet-Anzahl verwendet. Die Anzahl Retweets und Favoriten werden zunächst nicht als Kennzahl ausgewertet, sondern lediglich als Stammdaten verwendet. Außerdem stellt die Tweet-Faktentabelle auch zusätzliche Attribute für die Sentiments zur Verfügung, wie aus der Verbindung im Datenmodell hervorgeht. Dies

ermöglicht es zu nicht aggregierten Sentiments im Dashboard auch den Ursprungs-Tweet anzuzeigen. Die drei Dimensionen Datum, Uhrzeit und Ort sind nicht direkt auf einen Fakt zugeschnitten, sondern generell wiederverwendbar. Dabei werden Datum und Uhrzeit über ein selbsterstelltes SQL, welches im ODI als Prozedur hinterlegt ist, beladen. In der Ortsdimension werden die regionalen Daten aus dem Core denormalisiert dargestellt. Die Tweet-Nutzer sind gleichzeitig die Autoren des Tweets und die Halter des zugehörigen Sentiments. Die Entitäten-Dimension wird aus der Kontext-View (ohne Synonyme) aus dem Core abgeleitet. Diese entspricht auch dem Zielobjekt des Sentiments. Damit kann das in Abschnitt 4.4.1 vorgestellte Quintupel mit Hilfe der Faktentabelle zu den Sentiments und den Dimensionen komplett bedient werden. Die Dimensionstabelle D_TATORT_EPSIODEN enthält Stammdaten zu den Tatort-Episoden und Ermittlern. Da aktuell nur Erstausstrahlungen betrachtet werden, wäre es auch möglich diese Attribute direkt in die Faktentabelle zu den Quoten zu integrieren. Da es aber auch Fragestellungen geben kann, die z.B. Vergleiche zwischen Erstausstrahlung und Wiederholung fordern, ist es zukunftssicherer direkt die Stammdaten zu einer Tatort-Episode in einer eigenen Dimension zu sammeln.

5.2.5 Analyse

Die Analyse hat zum einen das Ziel die Tweets hinsichtlich ihres Sentiments zu bewerten und zum anderen Erkenntnisse darüber zu gewinnen, worauf sich der Tweet und damit auch das Sentiment beziehen. Implementiert ist die Analyse mit kaskadierenden Skripten im Statistik-Werkzeug R. Der Code ist im Anhang D zu finden. Einen Überblick über die R-Verarbeitung, die sich üblicherweise auf genau einen Sendetag bezieht, zeigt Abbildung 28. Diese und die folgenden Abschnitte gliedern sich nach dem EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe). Abschnitt 5.2.5.1 beschreibt die Eingangsdaten, die für die Analyse benötigt werden und wie diese in R eingelesen werden. In Abschnitt 5.2.5.2 wird die Durchführung der eigentlichen Analyse erläutert. Zuletzt zeigt Abschnitt 5.2.5.3 die Ausgabe bzw. Sicherung dieser in der Oracle-Datenbank.

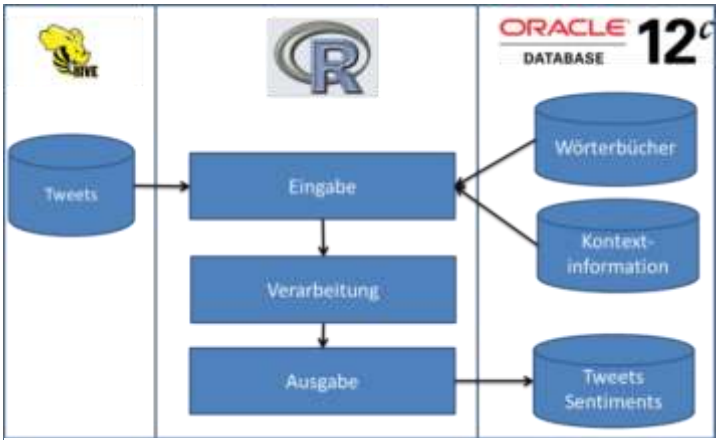

Abbildung 28: Verarbeitung in R

5.2.5.1 Eingabe

Bei der Eingabe sind zwei Quellen zu unterscheiden. Dies sind zum einen die über Hive zur Verfügung gestellten Tweets und zum anderen die Wörterbücher und Kontextinformationen, die im Tatort DWH (Oracle-Datenbank) hinterlegt sind.

Das Einlesen der Hive-Daten geschieht mit einem der Oracle Big-Data-Konnektoren, dem Oracle R-Konnektor für Hadoop. Dieser erweitert die Funktionalität vom in Kapitel 5.2.1 vorgestellten ORE-Paket um Funktionen zum Zugriff (Verbinden, lesen, schreiben) auf Hive. Damit können die Daten der in Abschnitt 5.2.3 vorgestellten View V_TWEETS_TATORT_FOR_R in R verfügbar gemacht werden. Zunächst geschieht dies als ORE-Objekt, welches nur eine Referenz auf das Hive-Objekt darstellt. Die Daten werden damit noch nicht physisch in R importiert. Um sicherzustellen, dass auch nur Tweets in R eingelesen werden, die zu einer Ausstrahlung gehören, werden diese nochmals anhand des Datums gefiltert. Erst danach wird das ORE-Objekt in ein Data-Frame (R-Datentyp zur Speicherung von Daten in einer Tabellenstruktur) umgewandelt und somit auch in den internen R-Speicher geladen.

Zum Zugriff auf die Oracle-Datenbank könnte ebenfalls das ORE-Paket verwendet werden. Jedoch wird stattdessen ein anderes R-Paket genutzt, welches auch in der freien R-Version zur Verfügung steht. Dies hat den Vorteil, dass parallel auf Hive und Oracle zugegriffen werden kann, ohne eine der beiden Verbindungen zu unterbrechen. Zudem enthält das ORE-Paket keine Möglichkeiten eine zur Laufzeit generierte Zeichenkette als SQL-Abfrage an die Datenbank zu senden. Für eine selbstgeschriebene Funktion (*f_get_begriffe*), die den Oracle-Zugriff kapselt, wird dies allerdings benötigt. Diese ermöglicht einen einfachen Zugriff auf die

Wörterbücher der Tabelle C_V_BIB_BEGRIFFE. Der Funktion wird der Name eines oder mehrerer Begriffstypen aus Tabelle 6 als Parameter übergeben und zurückgegeben wird ein Data-Frame mit allen relevanten Einträgen.

Name/Parameter	Beschreibung	Beispiel	Wertebereich
PHRASE_POS	Positive Phrase	Gut	]0,1]
PHRASE_NEG	Negative Phrase	Schlecht	[-1,0[
INCREASER	Verstärkt einen Sentiment-Wert	Sehr	]1,2]
DECREASER	Schwächt einen Sentiment-Wert	Bisschen	[0,1[
NEGATION	Kehrt den Sentiment-Wert um	Kein	-1
STOPWORD	Wort ohne Bedeutung bzw. Relevanz für die Analyse	Eine	-
STOPPATTERN	Muster für ein Wort ohne Bedeutung bzw. Relevanz für die Analyse	# (Hashtag in Tweets)	-

Tabelle 6: Wörterbücher

Der Großteil der beiden ersten Einträge stammt aus dem vorgefertigten Wörterbuch SentiWS. Die drei folgenden Kategorien und die Stopp-Muser sind manuell gepflegt. Die Basismenge der Stopp-Wörter stammt dagegen aus einer Bibliothek in R und wurde manuell erweitert.

Ähnlich geschieht der Zugriff auf die Kontext-Informationen (View C_V_BIB_KONTEXT) mit einer weiteren Funktion (*f_get_context*). Da die Kontextdaten immer nur gesamtheitlich verwendet werden, sind keine Parameter notwendig und die Funktion liefert alle Kontextinformationen in einem Data-Frame. Diesem werden aus der View die Episodentitel, Namen der Ermittler, Städtenamen der Spielorte und einige allgemeine Eigenschaften, die typisch für eine Fernsehserie sind (z.B. Musik, Handlung), zur Verfügung gestellt. Analog zu den anderen Wörterbüchern können diese flexibel erweitert werden. Dabei ist sowohl die Aufnahme komplett neuer Begriffe möglich (z.B. die Kameraführung als Eigenschaft einer Fernsehserie) als auch von Synonymen zu einem bereits vorhanden Begriff / einer vorhandenen Entität (z.B. der Spitzname eines Ermittlers).

5.2.5.2 Verarbeitung

Die Verarbeitung untergliedert sich nochmals in drei Bereiche und zwar Vorbereitung der Tweets, Bestimmung des Sentiment-Wertes und Zuordnung zu Kontextinformationen.

Während der **Vorbereitung der Tweets** werden diverse Text-Mining-Funktionen auf diesen zur Bereinigung und Aufbereitung der Tweet-Texte angewendet. Dabei werden insbesondere nicht benötigte und störende Textteile entfernt. Diese können zum einen das Ergebnis verfälschen. Zum anderen kann durch das Entfernen dieser die Performanz verbessern, da in der Analyse weniger Wörter betrachtet werden müssen. Dafür wird das Data-Frame mit den Tweets zunächst in einen Korpus umgewandelt. Dabei handelt es sich um eine Datenstruktur speziell zur Verwaltung von Dokumenten. Jeder Tweet entspricht in diesem einem Dokument. Der Korpus und auch die enthaltenen Dokumente können zudem um Metadaten angereichert werden. Bei den Bereinigungen unterstützt das R-Paket *tm*, welches eine Funktion *tm_map* enthält. Diese ermöglicht es Textfunktionen, die in diesem Paket mitgeliefert, selbstentwickelt oder durch andere Pakete zur Verfügung gestellt werden, effizient auf allen Dokumenten eines Korpus auszuführen.

Eine Übersicht über alle Vorbereitungen und Bereinigungen zeigt das folgende Datenflussdiagramm (vgl. Abbildung 29).

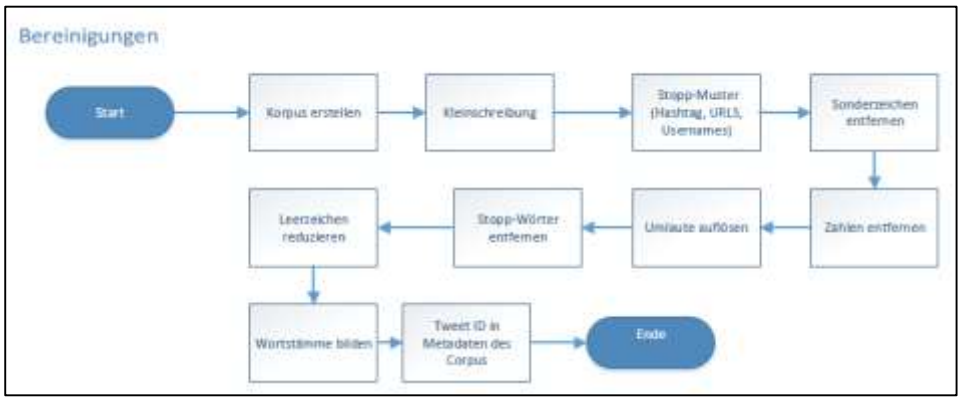

Abbildung 29: Vorbereitung der Tweets

Zur Transformation aller Buchstaben in Kleinschreibung, zum Entfernen von Sonderzeichen, Zahlen und überflüssiger Leerzeichen können direkt fertige Funktionen des tm-Pakets verwendet werden. Auch zum Entfernen bestimmter Wörter ist eine Funktion vorhanden. Dieser werden die Begriffe aus dem Stopp-

Wörterbuch, welches mit der zuvor vorgestellten Funktion *f_get_begriffe* und dem Parameter *STOPWORDS* abgefragt wird, übergeben.

Ähnliches geschieht mit den Stopp-Mustern. In R existieren Standard-Text-Funktionen (z.B. *gsub*), die mit Hilfe regulärer Ausdrücke nach Mustern in Zeichenketten suchen und entsprechende Ersetzungen durchführen können. Zusammen mit dem Stopp-Muster-Wörterbuch (Parameter *STOPPATTERN)* werden so insbesondere URLs, Nutzernamen (z.B. @TweetUser) und Hashtags (z.B. #tatort) entfernt.

Zum Auflösen der Umlaute wird eine ähnliche Funktion verwendet. Da diese fest definiert sind, geschieht die Umwandlung direkt in einem regulären Ausdruck. Neben den Umlauten (ä, ö, ü) wird dabei auch das „ß“ aufgelöst.

Die letzte Transformation ist das Bilden von Wortstämmen. Dies ist der Teil eines Wortes, der die Basis bildet und unabhängig von Kasus, Numerus, Verbform etc. immer identisch ist. Dazu existiert auch eine Funktion im *tm*-Paket. Da diese jedoch in der verwendeten Version immer nur den Wortstamm des letzten Wortes eines Dokumentes bildet, wird diese in einer Funktion *f_stemDocument* entsprechend erweitert.

Nach den Bereinigungen wird zuletzt zu jedem Tweet die Tweet-Id in den Metadaten des Korpus gesichert. Diese wird benötigt, um die Analyseergebnisse später mit den Daten in der Datensicherungssicht zu verknüpfen.

Im nächsten Schritt wird zu jedem Tweet ein **Sentiment-Wert bestimmt**. Da Tweets durch die Beschränkung der 140 Zeichen immer sehr kurz sind, kann angenommen werden, dass selten mehrere unterschiedliche Sentiments in einem Tweet stecken.[179] Verfolgt wird hier ein lexikonbasierter Ansatz (vgl. 4.4.2). Somit werden den Tweets aufgrund von Übereinstimmungen mit Wörtern aus einem Sentiment-Lexikon bestimmte Werte zugerechnet. Den Ablauf zur Bestimmung des Sentiment-Wertes zeigt Abbildung 30.

[179] Vgl. Pak / Paroubek 2010, S. 1321.

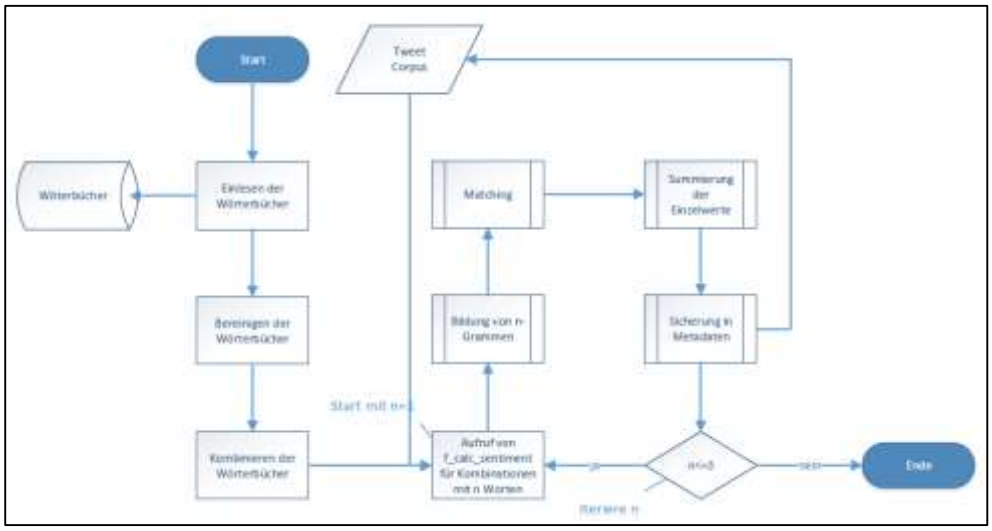

Abbildung 30: Flussdiagramm Sentiment-Analyse

Nachdem die Wörterbücher (PHRASE_POS, PHRASE_NEG, INCREASER, DECREASER) eingelesen sind, müssen die Begriffe dieser ebenfalls bereinigt werden. Ansonsten ist das Auffinden von Übereinstimmungen nicht möglich. Dabei ist vor allem eine Umwandlung in Kleinschreibung und die Bildung von Wortstämmen wichtig. Durch Kombination der Wörterbücher für Phrasen, Negationen und Verstärker/Abschwächer werden Wortketten von ein bis zu drei Worten generiert. Bei Ketten mit nur einem Wort handelt es sich direkt um eine positive oder negative Phrase (z.B. „schön“), bei zwei werden entweder Negationen oder Verstärker/Abschwächer mit den Phrasen kombiniert (z.B. „nicht gut“ oder „sehr gut“) und bei drei werden Phrasen mit Verstärkern/Abschwächern und Negationen kombiniert (z.B. „nicht sehr schön“). Bei der Wortkombination findet ebenfalls die Berechnung eines Gesamtwertes für diese Kombination aus den zugehörigen Einzelwerten statt. Ein Punkt, der dabei besonders hervorzuheben ist, ist das gemeinsame Auftreten von Negationen mit Verstärkern/Abschwächern. Dies soll am Beispiel von Abbildung 31 verdeutlicht werden. Dort sind verschiedene Kombinationsmöglichkeiten von Negationen (kein, nicht), einem Verstärkter (sehr) und einem Abschwächer (bisschen) auf einer Achse von ganz negativ bis absolut positiv dargestellt.

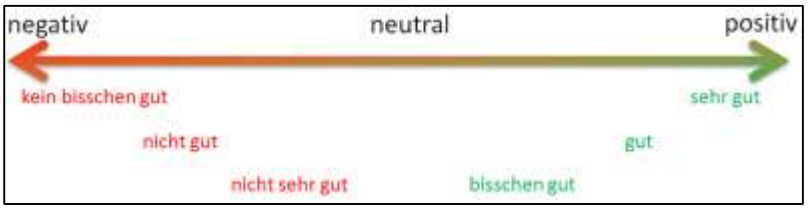

Abbildung 31: Beispiel für Wortkombinationen

Wenn dem Wort „gut“ der Wert 0,6, der Negation „nicht“ -1, dem Verstärker „sehr“ 1,5 und dem Abschwächer „bisschen“ 0,75 zugeordnet ist, wird deutlich, dass bei Kombinationen mit zwei Wörtern direkt das Produkt ihrer Werte sinnvoll verwendet werden kann. Bei Kombinationen mit allen drei Wortarten spiegelt das allerdings nicht in allen Fällen die Realität wieder. So ergibt beispielweise für den Begriff „sehr gut“ die Multiplikation von 0,6 und 1,5 einen Gesamtwert von 0,9. Dies ordnet diesen korrekt auf der Achse ein, da „sehr gut“ positiver als nur „gut“ zu sehen ist. Ähnlich ist es bei dem Abschwächer. Der Begriff „bisschen gut“ erhält den Wert 0,45 und ist somit kleiner als der Wert für „gut“, was auch der Achse entspricht. Bei der Kombination von drei Begriffen (z.B. „nicht sehr gut“) funktioniert dies jedoch nicht. Das Produkt aller drei Einzelwerte ergibt -0,9 $(-1 \times 1{,}5 \times 0{,}6)$. Dies würde ausdrücken, dass dieser Ausdruck negativer wäre als „nicht gut“, was jedoch nicht der Fall ist. Grund ist, dass der verstärkende Begriff „sehr“ in Kombination mit einer Negation, wie z.B. „nicht“ eine abschwächende Wirkung hat. Ähnlich verhält es sich mit abschwächenden Begriffen (z.B. „kein bisschen gut“). Hier wirkt sich der Begriff „bisschen“ verstärkend aus. Um diesem Zusammenhang in der Bewertung gerecht zu werden, wird bei einem gemeinsamen Auftreten von Negation und Verstärker bzw. Abschwächer der Umkehrwert von letzterem verwendet. Für das Beispiel „nicht sehr gut“ bedeutet dies, dass sich ein Gesamtwert von -0,4 $(-1 \times \frac{1}{1{,}5} \times 0{,}6)$ anstatt von -0,9 ergibt, der auch passend zu der Achse ist.

Im nächsten Schritt werden die Wortketten in Abhängigkeit ihrer Wortanzahl in drei verschiedenen Stufen beginnend mit den Einzelworten mit den Tweets abgeglichen. Die Analyse für eine Stufe geschieht innerhalb einer R-Funktion (*f_calc_sentiment*). Dieser werden die Tweet-Texte, die aus dem Sentiment-Lexikon generierten Wortketten zu einer Stufe und die Anzahl der Worte für eine Wortkette dieser Stufe übergeben. Innerhalb der Funktion werden mit zuletzt genanntem Parameter n-Gramme aus den Tweets gebildet, wobei das *n* diesem Pa-

rameter entspricht. Die n-Gramme werden mit den übergebenen Wortketten abgeglichen. Um den Gesamtsentiment-Wert eines Tweets zu berechnen, wird die Summe der Einzelwerte zu den gefundenen Wörtern bzw. Wortketten ermittelt. Wird ein Tweet in mehreren Stufen bewertet, werden die Ergebnisse in jeder Stufe aufaddiert. Dabei muss davon ausgegangen werden, dass bei einer gefundenen Wortkette bereits auf der vorherigen Stufe ein Treffer stattgefunden haben muss. Um eine Verfälschung der Ergebnisse zu vermeiden, muss dieser bereinigt werden. Daher werden bereits bei der Berechnung des Gesamtwertes zu einer Wortkette die zugehörigen Werte zu der nächstniedrigeren Stufe abgezogen. An dem Beispiel von folgendem Original-Tweet wird diese Berechnung verdeutlicht:

Franggen #tatort schlug sich nicht schlecht. Viel Dialekt, gute Story. 2,5 @Tatort

Die im Sentiment-Lexikon vorhandenen Begriffe sind grün markiert. Der Sentiment-Wert nach der Ausführung der jeweiligen Stufe und der Gesamtsentiment-Wert von 1,1422 sind in Tabelle 7 zu sehen.

Stufe	**Beschreibung**	**Einzelwert**	**Gesamtwert**
1 (Unigramme)	Wort „schlecht“ gefunden	-0,7706	-0,399
	Wort „gut“ gefunden	0,3716	
2 (Bigramme)	Wortkette „nicht gut“ gefunden	-1 x -0,7706 = 0,7706	1,1422
	Korrektur Unigramme („schlecht“)	-(-0,7706) = 0,7706	
3 (Trigramme)	-	-	**1,1422**

Tabelle 7: Beispiel Bestimmung Sentiment-Wert

In einem zweiten Analyse-Schritt werden den Tweets **Kontextinformationen** bzw. Entitäten zugeordnet, auf welche sie sich beziehen. Diese Zuordnung soll es ermöglichen das Zielobjekt, welchem der zuvor berechnete Sentiment-Wert gilt, zu identifizieren. Dabei besteht auch die Möglichkeit, dass sich ein Tweet auf mehrere Entitäten beziehen kann. Der Ablauf für die Zuordnung gleicht stark dem der zuvor vorgestellten Sentiment-Analyse. Auch auf den Kontext-Informationen findet nach dem Einlesen über die Funktion *f_get_context* eine entsprechende Bereinigung statt. Im Anschluss wird für jede Kontextinformation die Wortanzahl

ermittelt. In Abhängigkeit dieser Anzahl werden die Kontextinformationen und Tweets einer Funktion (*f_match_context*) übergeben. Dies geschieht ebenfalls dreistufig für Unigramme, Bigramme und Trigramme. In dieser werden wieder n-Gramme aus den Tweets generiert, die mit den entsprechenden Kontextinformationen abgeglichen werden. Bei einem Treffer wird ein Schlüssel für die Entität in den Metadaten des Dokumentes festgehalten. Sollten bei den Ausführungen unterschiedliche Entitäten einem Tweet zugeordnet werden können, werden die Schlüssel durch Konkatenation zusammengefasst.

5.2.5.3 Ausgabe

Zu diesem Zeitpunkt befinden sich die Ergebnisse der Analyse nur im flüchtigen Speicher von R. Zusammengefasst ist dies ein Objekt vom Typ Korpus mit den Tweet-Texten als Dokumente. Zusätzlich wurden während der Vorbereitung und der Analyse drei Informationen in den Metadaten der Tweets abgelegt und zwar die Tweet-Id, ein Sentiment-Wert und zuletzt eine Liste von Entitäten-Schlüsseln. Diese Metadaten werden aus dem Korpus extrahiert und zu einem Data-Frame geformt. Dabei werden bereits sprechende Attributnamen festgelegt (vgl. Tabelle 8).

Attributbezeichnung	Datentyp	Beschreibung	Beispiel
TWEET_ID	Integer	Id des Tweets	567063470932525056
ENTITAET_KEYS	String	Ids zur zugehörigen Entität	TE_158, ER_475, ER_484
SENTIMENT_WERT	Dezimal	Stimmungswert	0.1628

Tabelle 8: Sentiment Data-Frame

Schließlich wird mit dem ORE-Paket eine Verbindung zur Oracle-Datenbank geöffnet und das Data-Frame in den Stage-Bereich des Tatort-DWH geschrieben. Physisch wird dabei zuerst die Zieltabelle gelöscht und im Anschluss daran mit den aktuellen Daten neu angelegt.

5.2.6 Auswertung

In dieser Schicht werden die Daten aus der Mart-Schicht des Tatort-DWH und somit auch die Ergebnisse der R-Analyse visualisiert. Die Darstellung geschieht mit dem Dashboard-Werkzeug Answers aus der OBIEE. Dazu muss zunächst im Oracle BI Administrations-Werkzeug, welches ebenfalls Bestandteil der OBIEE

ist, ein Repository erstellt werden. Dies ist eine Metaschicht zwischen den physischen Datenbankobjekten und den Dashboards. Dort werden verschiedene Konfigurationen, Transformationen und Gruppierungen der Daten in drei Schichten vorgenommen (vgl. Abbildung 32).

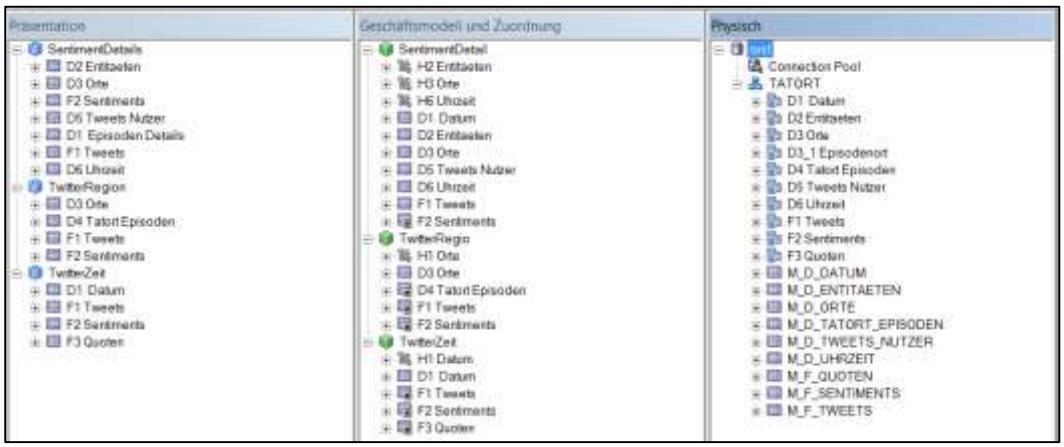

Abbildung 32: Oracle Business Intelligence Administration

In der **physischen Schicht** wird die Verbindung zur Datenquelle, welche im Fallbeispiel die Mart-Schicht des Tatort-DWH ist, hergestellt und das physische Datenmodell importiert (vgl. Anhang F). Zudem wird für jede Tabellen ein Alias vergeben.

In der Schicht **Geschäftsmodell und Zuordnung** geschieht eine Gruppierung in drei Geschäftsmodelle. Innerhalb dieser werden Kennzahlen (markiert mit Doppelkreuz #), Dimensionen und Hierarchien (dargestellt durch drei Pfeile) innerhalb von Dimensionen modelliert. So existieren beispielsweise in der Hierarchie H3 Orte die Level Land, Bundesland und Stadt und in der Hierarchie H1 Datum die Level Jahr, Quartal, Monat und Datum (Tag). Durch die Verwendung von Hierarchien ist es wichtig, dass zu den Kennzahlen Aggregationsregeln konfiguriert werden. Außerdem erhalten die Attribute bereits den Namen, der auch später in Oracle Answers sichtbar ist. Einen Überblick über den Zweck des jeweiligen Geschäftsmodells beinhaltet Tabelle 9. Neben einer Beschreibung der Zielsetzung und der enthaltenen Kennzahlen, wird auch jeweils eine der im Abschnitt 5.1 vorgestellten Beispielfragen zugeordnet. Die Geschäftsmodelle sind allerdings so geschnitten, dass neben dieser noch weitere Fragenstellungen beantwortet werden können.

Geschäftsmodell	Zielsetzung	Kennzahlen	Beispielfragestellung
SentimentDetail	Flexible Detail-Analyse der Sentimente mit vielen Attributen.	Sentiment (Wert, Anzahl positiv, negativ, neutral)	Sind Unterschiede zwischen der Stimmung vor Beginn einer Episode (Erwartungshaltung), während der Ausstrahlung und nach der Episode zu beobachten?
TwitterRegio	Auswertung der Twitter-Daten (Sentiments und Anzahl) hinsichtlich ihrer regionalen Herkunft.	Sentiment-Wert (Wert, Anzahl positiv, negativ, neutral) Anzahl Tweets Tatort-Spielorte	Sind regionale Unterschiede in der Stimmung zu erkennen?
TwitterZeit	Zeitliche Auswertung der Sentiments, Tweet-Anzahl und Einschaltquoten.	Einschaltquote (absolut, prozentual) Anzahl Tweets Sentiment (Wert, Anzahl positiv, negativ, neutral)	Ist ein Zusammenhang zwischen der Stimmung oder Tweet-Anzahl und Einschaltquoten den zu erkennen?

Tabelle 9: Geschäftsmodelle

Die drei Geschäftsmodelle werden schließlich in sogenannte Themenbereiche der **Präsentationsschicht** überführt. Dort werden zum einen die für die Präsentation relevanten Attribute selektiert, sodass nur noch inhaltlich relevante und keine technischen Schlüsselspalten angezeigt werden. Zum anderen muss für die Hierarchien entschieden werden, welche Attribute auf welchem Level dargestellt werden sollen. Die drei erstellten Themenbereiche bilden die Basis für die Analyse und Auswertungen in Oracle Answers. Im Folgenden werden einige Beispiel-Analysen aus Answers gezeigt, die auf diesen Themenbereichen basieren.

Der erste Bericht (vgl. Abbildung 33) zeigt einen **Überblick** über den Verlauf der Einschaltquoten, Sentiments und der Tweet-Anzahl.

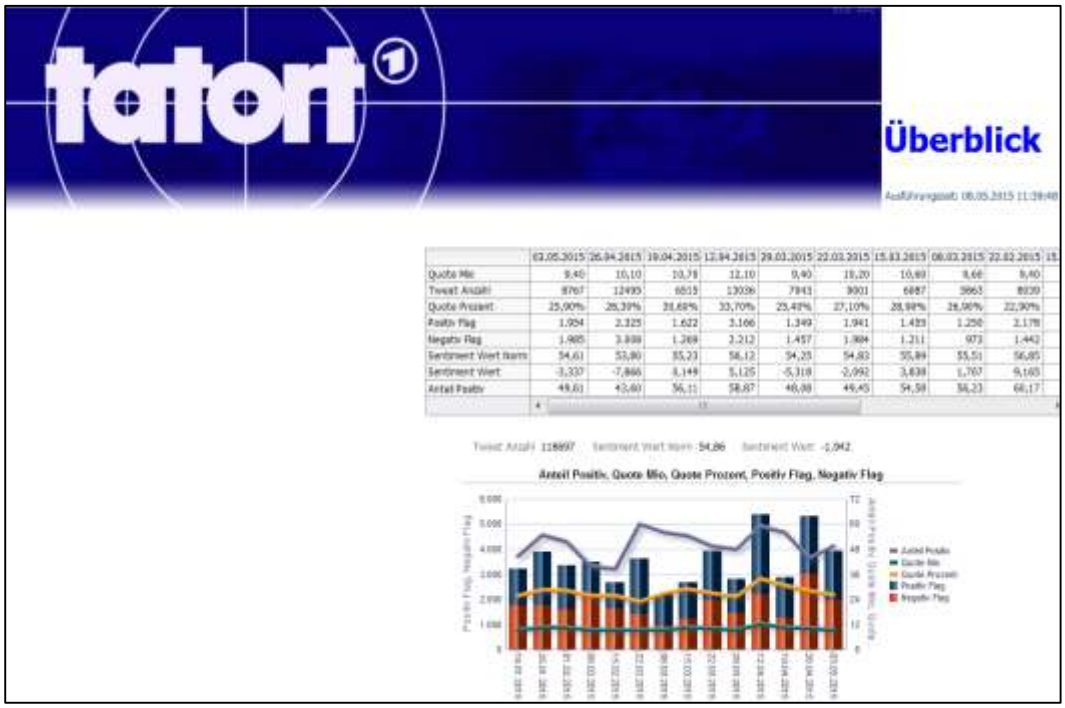

Abbildung 33: Beispielbericht Überblick

Zu erkennen ist, dass für diesen fast 120.000 Tweets analysiert wurden, die im Durchschnitt mit einem normalisierten Sentiment-Wert von 54,86 (Skala 0 bis 100) bewertet sind. Neutrale Sentiments sind exkludiert, da diese einen Großteil aller Sentiments ausmachen und aggregierte Durchschnittswerte neutralisieren. Dennoch liegen solche Durchschnittswerte sehr dicht beieinander, was eine Interpretation erschwert. Deutlichere Unterschiede sind dagegen am Anteil der positiv kategorisierten Sentiments zu erkennen. Anhand der Linien in der Grafik ist zu erkennen, dass für einige Ausstrahlungstermine der Verlauf dieses Anteils und der Einschaltquote sich ähnlich verhalten (z.B. am 12.04.2015). Dies lässt eine Abhängigkeit vermuten. Jedoch ist dies nicht immer der Fall. Am 22.02.2015 ist der Verlauf beispielsweise entgegengesetzt. Für eine endgültige Aussage diesbezügliche würde mehr Datenmaterial benötigt werden, um die Korrelation zwischen diesen ermitteln zu können. Für den Zusammenhang zwischen der Anzahl der Tweets und der Einschaltquote gilt ähnliches. Für die Anzahl der Tweets kann im Diagramm die Balkenhöhe als Indikator genutzt werden.

Im zweiten Beispielbericht ist eine Darstellung nach **Regionen** (Vgl. Abbildung 34) zu sehen.

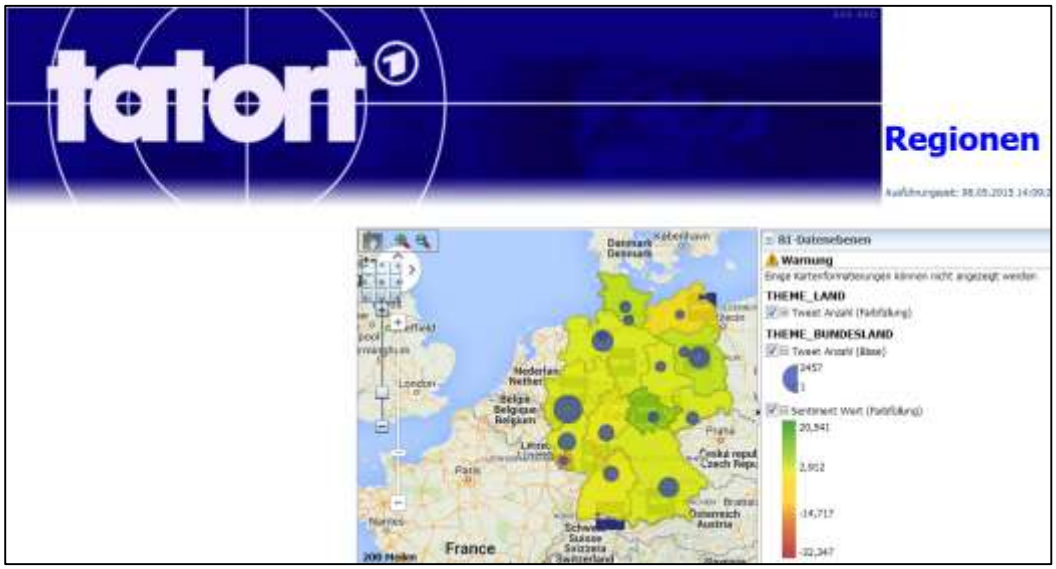

Abbildung 34: Beispielbericht Regionen

Dieser wertet die Tweets nach den Regionen der Twitter-Nutzer aus. Die Größe der Blasen visualisiert dabei die Anzahl der Tweets je Bundesland. So lässt sich beispielsweise erkennen, dass in Berlin, Niedersachsen und Nordrhein-Westfalen am meisten zu Tatort getwittert wird. Dagegen wird beispielsweise in Hessen oder Baden-Württemberg weniger getwittert. Die Stimmung im jeweiligen Bundesland wird über den Farbton dargestellt. Hier sind nur kleine Unterschiede zu erkennen. Alle Bundesländer liegen vergleichsweise neutral in einem hellgrünen Farbton. In den dunkleren herrscht dabei eine leicht bessere Stimmung. Es existieren jedoch keine Ausreißer, die besonders gut (komplett dunkel grün) oder besonders schlecht (rot) sind. Zuletzt befinden sich im Hintergrund Tatort-Symbole, die die verschiedenen Spielorte kennzeichnen. Die Schwierigkeit bei solch einer regionalen Auswertung nach Nutzerherkunft ist, dass der Großteil der Twitter-Nutzer in Deutschland die Speicherung der Ortsangabe deaktiviert hat. Nur für etwa fünf Prozent der gesammelten Tatort-Tweets liegt diese Information vor. Um repräsentative Ergebnisse zu erhalten, müssten entsprechend viele Tweets vorhanden sein. Dies macht es beispielsweise nicht möglich diesen Bericht auf Episoden-Ebene darzustellen, um zu untersuchen, ob Nutzer aus einer bestimmten Region den Episoden, die in dieser Region spielen, positiver geneigt sind als Nutzer aus einer anderen Region.

Der nächste Bericht (vgl. Abbildung 35) wertet die Stimmung hinsichtlich der **Uhrzeit** aus. Dazu können vorher über einige Auswahllisten eine oder mehrere Episoden ausgewählt werden.

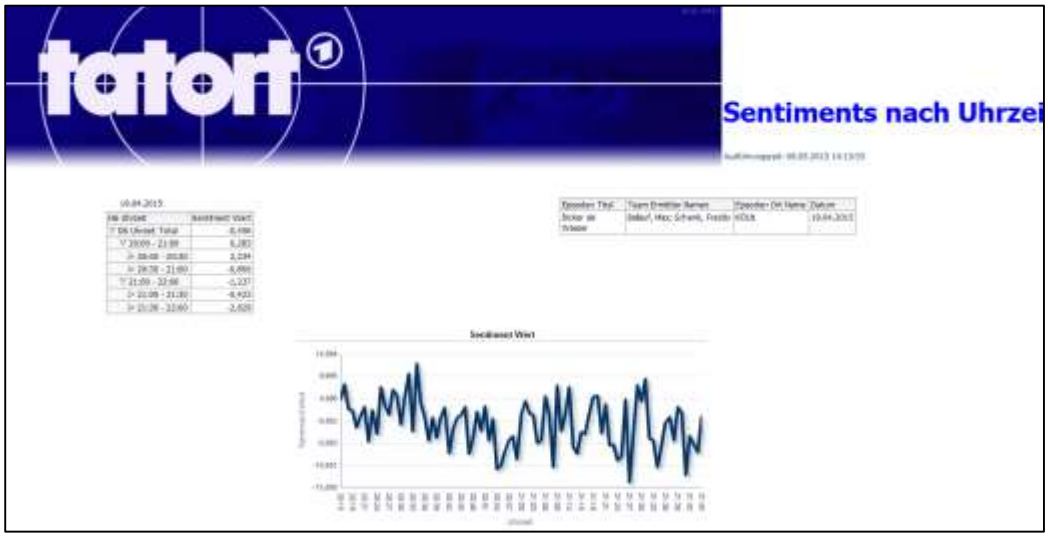

Abbildung 35: Beispielbericht Uhrzeit

Als Beispiel ist eine Episode aus Köln zu sehen. Die Stimmung wird auf Minutenebene aggregiert und als Liniendiagramm dargestellt. Im Großen und Ganzen lassen sich einige durchgehende Schwankungen erkennen und ein leicht negativer Gesamttrend. Dies zeigt auch die Tabelle links. Die erste Viertelstunde hatte den besten Sentiment-Wert und die letzte den schlechtesten.

Der letzte Beispielbericht enthält einen Überblick über die **Top/Flop-Tweet-Texte**. So sind in Abbildung 36 Tweet-Texte zu sehen, die einen besonders guten und in Abbildung 37 diejenigen, die einen besonders schlechten Sentiment-Wert haben. Diese Auswertung kann über alle Tweets oder auch nur für Tweets einer einzelnen Episode ausgeführt werden. Genutzt werden kann diese in erster Linie, um einen ersten Eindruck der Meinungen zu erhalten. Zudem kann diese auch Vertrauen bei den Anwendern schaffen, da dieser in diesem Bericht auch Werte zusammen mit dem Text sehen kann.

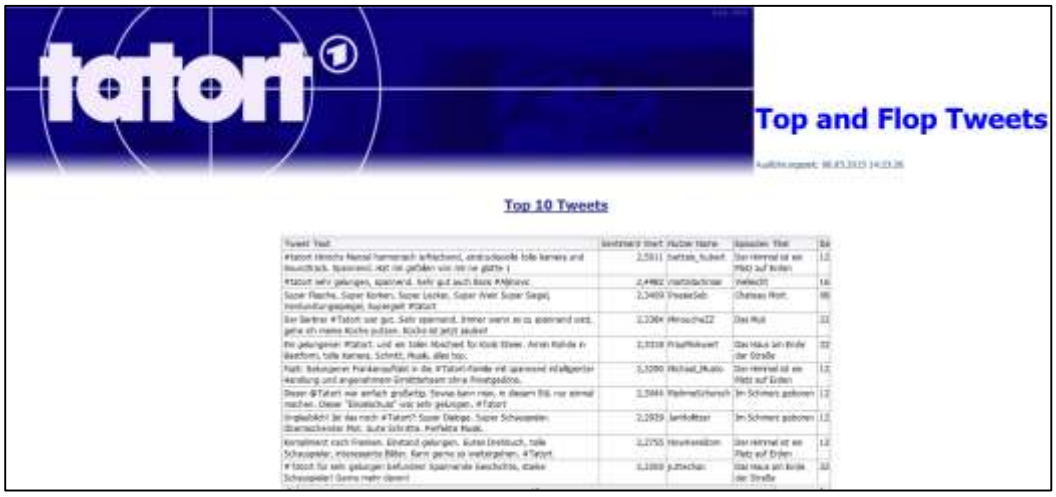

Abbildung 36: Top Tweets

Festzustellen ist dabei, dass das Programm sowohl bei den positiven als auch bei den negativen Tweets immer richtig liegt.

Flop 10 Tweets

TWEET_TEXT	Sentiment Wert	Nutzer Name	Episoden Titel
#Zeit für 90's #Rap schlecht schlecht schlecht. ECHT schelle schlecht #Tatort	-3,082	Ztztztztztztztztz	Der Maulwurf
Wie schlecht. Schlecht. Schlecht. Schlecht. #tatort #schlecht.	-3,082	mortlinho	Chateau Mort
bei schlechten Kritiken lass ich den #tatort schon mal ausfallen, aber bei soooooooo schlechten Kritiken muss ich reinschauen...	-2,996	erfolgsprinzip	Der Maulwurf
Ab den #mdr Euer #Tatort ist schlecht; schlecht gespielt, schlechter Plot und total kranke Extreme!!! "Untergang des Tatortsonntags"	-2,821	abrhamburg	Niedere Instinkte
An den #mdr Euer #Tatort ist schlecht; schlecht gespielt, schlechter Plot und total kranke Extreme!!! "Untergang des Tatortsonntags" #Satire	-2,821	abrhamburg	Niedere Instinkte
Regisseur auf Drogen? Schlimme Drogen. Schlimme, schlimme Drogen. Schlimme, schlimme, schlimme Drogen. #Tatort	-2,460	larnius	Niedere Instinkte
Kann mich nicht entscheiden, weshalb ich den heutigen #Tatort so schlecht finde: A.) Schlechte Schauspieler B.) Schlechte Texte Bin raus.	-2,312	DejanFreiburg	Winternebel
Schlechte Dialoge, schlechte SchauspielerInnen und eine schlechte Geschichte kennzeichnen den heutigen #tatort.	-2,312	Maeldae	Winternebel
Richtig schlecht! Schlechte Dialoge, schlechte Story. #Tatort	-2,308	CresslWood	Der sanfte Tod
Schlecht konstruiertes Ende bei einem grottigen #Tatort - schlimm, schlimmer, am schlimmsten ☹	-2,209	pintawohl	Niedere Instinkte

Abbildung 37: Flop Tweets

Bei den negativen fällt jedoch ein Tweet auf. Der Nutzer „abrhamburg“ scheint einen Tweet korrigiert bzw. verändert zu haben, was zu einem neuen Tweet und somit auch einem weiteren Sentiment geführt hat. Insbesondere hinsichtlich der Aggregationen könnte an dieser Stelle diskutiert werden, ob beide beibehalten und gezählt werden sollten.

5.3 Ergebnis

Zum Erfolg solch einer Anwendung trägt insbesondere die Qualität der Sentiment-Analyse bei, da diese das Vertrauen der Anwender erlangen muss. Dies ist insbesondere wichtig, da die Ergebnisse dieser Analyse im Vergleich zu klassischen Berechnungen in einem Data Warehouse ungenau sind. Daher ist es essentiell die Qualität durch regelmäßige Überprüfungen zu messen und ständig zu verbessern.

Prinzipiell sind zur Validierung der Ergebnisse der Tatort-Sentiments zwei unterschiedliche Ansätze denkbar. Beim ersten werden die Tweets parallel durch einen Menschen manuell bewertet. Da dies mit einem erheblichen Aufwand verbunden sein kann, muss auf eine geeignete Stichprobe geachtet werden, die möglichst breit streut. Idealerweise muss sichergestellt werden, dass positiv-, negativ- aber auch neutralbewertete Tweets geprüft werden. Die Top- und Flop-Liste aus dem vorherigen Abschnitt reicht dazu nicht aus. Beim zweiten Ansatz werden weitere automatisierte Verfahren eingesetzt, mit denen parallel die Tweets bewertet und die Resultate verglichen werden. Dieses automatisierte Verfahren kann ebenfalls nach dem lexikonbasierten Ansatz arbeiten und dabei beispielsweise ein anderes Wörterbuch benutzen. Um jedoch falsch bewertete Tweets zu identifizieren, die auf Schwachstellen des lexikalischen Ansatzes zurückzuführen sind, bringt es Vorteile einen komplett anderen Ansatz (z.B. eine statistische Methode, wie Naive Bayés) zu verwenden. Grundsätzlich hat der automatisierte Ansatz den Vorteil, dass dieser nach der Entwicklung, ebenfalls alle Tweets recht schnell bewerten kann. So kann er auch regelmäßig ausgeführt werden. Allerdings hat er den Nachteil, dass spätestens, wenn größere Abweichungen auftreten, auf eine menschliche Nachprüfung/Entscheidung nicht verzichtet werden kann und auch bei Übereinstimmungen eine gewisse Restunsicherheit bestehen bleibt. Aufgrund dieser Vor- und Nachteile wird sich eine Kombination als ideal erweisen. So können alle Analyse-Ergebnisse mit denen eines statistischen Ansatzes schnell und komplett automatisiert verglichen werden. Werden dabei größere Abweichungen gefunden, können diese und auch zusätzlich kleinere Stichproben durch einen Menschen überprüft werden.

Für die Tatort-Sentiments wird ausschließlich ein manueller Abgleich durchgeführt. Dazu werden per Zufallsfunktion jeweils 100 positiv-, neutral- und negativ-

bewertete Tweets direkt aus der Datenbank selektiert, die im Anschluss manuell bewertet werden. Einfachheitshalber findet eine Zuordnung lediglich in die Kategorien positiv, neutral oder negativ statt und nicht die Berechnung eines Sentiment-Wertes. Mit Hilfe der Ergebnisse kann zum einen eine quantifizierbare Aussage über die Qualität der Analyse getroffen werden. Zum anderen können durch die falsch bewerteten Tweets Anhaltspunkte gefunden werden, wo die Analyse verbessert werden kann. Im Idealfall kann dies durch eine Anpassung der Wörterbücher geschehen. Die kompletten Ergebnisse der manuellen Validierung sind im Anhang G zu finden. Eine Zusammenfassung der Ergebnisse zeigt Tabelle 10.

Bewertet mit Sentiment-Lexikon / manuell	Positiv	Neutral	Negativ
Positiv	53	16	0
Neutral	38	72	33
Negativ	9	12	67

Tabelle 10: Validierung Sentiment-Analyse

Als positiv ist festzustellen, dass in allen drei Kategorien, die richtig bewerteten Tweets überwiegen. Ebenso ist eine komplett falsche Zuordnung sehr selten. So sind positiv bewertete Tweets selten negativ und umgekehrt sind sogar in der Stichprobe als negativ bewertete Tweets nie positiv. Allerdings zeigen die Ergebnisse auch, dass die neutralen Tweets nicht immer korrekt zugeordnet werden können. Dadurch sind insbesondere die 53%, die als positiv bewertet wurden und auch wirklich positiv sind, auffallend niedrig. Die Ursachen für diese Ungenauigkeiten sind auf Schwachstellen in dem lexikonbasierten Ansatz, dem genutzten Sentiment-Lexikon oder auch auf die Vorbereitungen für die Analyse bzw. Nachbereitung für die Auswertung zurückzuführen.

Ein Beispiel, welches die ersten beiden Punkte betrifft, liefert folgender Tweet (vgl. Abbildung 38):

TWEET_TEXT	SENTIMENT_WERT	KATEGORIE
Schön, wenn ein #Tatort gleich mit einem komplizierten "Fall" beginnt	-0,2456	NEUTRAL

Abbildung 38: Beispiel-Tweet Homonym

In diesem taucht das Homonym „Fall“ auf. Dieses kann vom Verb „fallen“ stammen, aber auch einen grammatikalischen Fall oder auch einen (von der Polizei) zu lösenden Fall beschreiben. Im aktuellen Tweet ist die letztgenannte Bedeutung die zutreffende. Allerdings ist im Sentiment-Lexikon nur für die erste Bedeutung ein Sentiment-Wert hinterlegt und zwar ein negativer. Dieser neutralisiert das Wort „schön“ vom Satzbeginn und führt zu einer neutralen anstatt einer positiven Bewertung für diesen Tweet. In dem Fall könnte eine Lösung ein Komplettentfernen des Wortes „Fall“ aus dem Lexikon die pragmatischste Lösung sein, da im Kontext einer Kriminalserie das Wort „Fall“ häufig mit dieser Bedeutung zu erwarten ist und daher keinen Sentiment hat.

An folgendem Tweet (vgl. Abbildung 39) wird eine weitere Schwachstelle des genutzten Sentiment-Lexikons deutlich:

TWEET_TEXT	SENTIMENT_WERT	KATEGORIE
Boah ist das öde in Erfurt. #tatort #motzecke	-0,457	NEUTRAL

Abbildung 39: Beispiel-Tweet Sentiment-Wert

Das Sentiment-Lexikon ist noch zu wenig auf den Kontext der Fernsehserie Tatort angepasst. Dieser Tweet ist ebenfalls als neutral kategorisiert worden. Grund dafür ist, dass das Wort „öde“ einen sehr geringen Sentiment-Wert hat. Ähnliches gilt auch für das Wort „langweilig“. Für eine Fernsehserie und insbesondere eine Kriminalserie ist „öde“ oder „langweilig“ allerdings etwas sehr negatives. Hier macht eine regelmäßige Neuberechnung der Sentiment-Werte Sinn. Dazu kann eine Variante eines korpusbasierten Ansatzes, der in Abschnitt 4.4.2 vorgestellt wurde, mit den Tatort-Tweets selbst als Korpus sinnvoll verwendet werden. Hinsichtlich des Lexikons fällt ebenfalls auf, dass einige Worte fehlen, die aber in den Tweets zur Umschreibung der Tatort-Serie verwendet werden und auch eine Stimmung ausdrücken (z.B. „unterhaltsam“). Solche Wörter können natürlich manuell dem Sentiment-Lexikon angefügt und wie zuvor beschrieben mit einem Sentiment-Wert ausgestattet werden. Alternativ oder zusätzlich könnte dies auch automatisiert geschehen. Es können beispielsweise immer sehr gut und sehr schlecht bewertete Tweet-Texte betrachtet werden und Wörter, die in diesen häufig vorkommen, aber noch nicht im Sentiment-Lexikon vorhanden sind, aufgenommen werden.

Das nächste Beispiel (vgl. Abbildung 40) zeigt ein Problem, welches durch die Vorbereitungen der Tweet-Texte entsteht und zwar durch die eingesetzte Funktion zum Bilden der Wortstämme.

Abbildung 40: Beispiel-Tweet Wortstamm

Dieses Problem entsteht durch die Auflösung der Umlaute. Dabei transformiert die Wortstamm-Funktion „ä“ in „a“, „ö“ in „o“ und „ü“ in „u“. Dies führt dazu, dass der Wortstamm vom Wort „schön“ genau wie das Adverb „schon“ lautet und das Adverb immer einen positiven Sentiment-Wert erhält. Dieser verbessert den Gesamtwert des Textes, sodass dieser nicht mehr als negativ, sondern als neutral kategorisiert wird. Die einfache Lösung für das konkrete Beispiel ist die Aufnahme des Adverbs „schon“ in das Stoppwörterbuch. Allerdings existieren auch weitere Wörter, bei denen solch ein Problem auftritt (z.B. tauschen und täuschen). Zudem ermittelt diese Funktion auch für verschiedene Worte ohne Umlaute, die eine unterschiedliche Bedeutung haben, den gleichen Wortstamm. Ein Beispiel dafür sind das vom positiven Wort „mögen“ stammende Verb „mag“ und das negative Adjektiv „mager“. Für beide wird von der Funktion der Wortstamm „mag“ zurückgegeben. Dies führt dazu, dass „mag“ als „mager“ interpretiert und somit fälschlicherweise negativ bewertet wird. Daher kann eine nachhaltige Lösung nur durch eine Verbesserung der Wortstammfunktion erzielt werden.

Ein letztes Beispiel (vgl. Abbildung 41) zeigt ein Problem, welches sich nicht komplett vermeiden lässt.

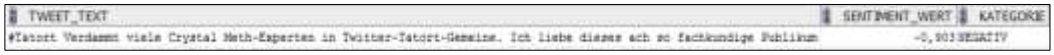

Abbildung 41: Beispiel-Tweet Rechtschreibfehler

Dieses wird durch einen Rechtschreibfehler verursacht. Ein fehlendes „d“ in Gemeinde führt dazu, dass dieses Wort als „gemein“ und damit negativ bewertet wird. Solche Fehler sind sicherlich ein Schwachpunkt des lexikonbasierten Ansatzes. Allerdings kann auch nicht jeder statistische Ansatz damit umgehen. Solche, die das Auftauchen bestimmter Wörter als Eigenschaften betrachten, wie der in Abschnitt 4.4.3 beschriebene Naive Bayés, werden ebenfalls damit Schwierigkeiten haben. Häufig vorkommende bzw. typische Rechtschreibfehler können beim

lexikonbasierten Ansatz auch mit in die Wörterbücher aufgenommen und damit abgefangen werden. Jedoch sollte dabei vermieden werden jeden einzelnen Rechtschreibfehler, der einmal vorkommt mit aufzunehmen. Dadurch würde eine Überanpassung des Sentiment-Lexikons an den aktuellen Daten geschehen, was für die Bewertung neuer Tweets vermutlich eher geringe Vorteile bringt. Dagegen besteht jedoch das Risiko einer schlechteren Performanz, da das Lexikon zu groß wird. Zudem können gerade in Verbindung mit dem zuvor gezeigten Problem zur Wortstammfunktion neue Seiteneffekte entstehen.

Die Zuordnung zu den Zielobjekten bzw. Entitäten muss ebenfalls genau wie die Ermittlung der Sentiment-Werte überwacht und verbessert. Da bei dieser Zuordnung jedoch weniger Spielraum besteht, dass sich durch den Kontext das Ergebnis verändert ist diese weitaus weniger fehleranfällig. Aber auch dort gibt es einige Tücken. So zeigt das Beispiel in Abbildung 42, dass der Ermittler mit dem Namen „Stark“ und das Adjektiv schwer auseinander zu halten sind.

Abbildung 42: Beispiel Entitätenzuordnung

Verbessert werden könnte dort das Ergebnis durch einen sogenannten Part-Of-Speech-Tagger, der automatisiert Wörtern ihre jeweilige Wortart zuordnet. Das würde ermöglichen in solch einem Fall Adjektive nicht als Entitäten zuzulassen. Dennoch dürfte auch ein Part-Of-Speech-Tagger Probleme mit dem Wortspiel im zweiten Tweet aus Abbildung 42 haben.

Auch das Hinzufügen neuer Begriffe im zugehörigen Wörterbuch mit Entitätenbegriffen sollte teilautomatisiert auf Basis des Korpus unterstützt werden. Gerade in Verbindung mit dem genannten Part-Of-Speech-Tagging könnten dazu beispielsweise Nomen, die in besonders vielen Tweets vorkommen als mögliche Kandidaten angeboten werden. Weitere Kandidaten können aber auch Begriffe sein, die mit einem Verknüpfungswort, wie z.B. „und“ mit einem bereits erfassten Zielobjekt verbunden sind.

6 Fazit und Ausblick

In der Arbeit wurden mit Hilfe einer Big-Data-Komponente Nachrichten aus Twitter extrahiert, analysiert und in ein klassisches, relationales Data Warehouse geladen. In diesem wurden aus den Ergebnissen der Analyse Kennzahlen abgeleitet, die die Stimmung der Twitter-Nachrichten wiederspiegeln. Diese wurden unter anderem auch zusammen mit klassischen Kennzahlen nach verschiedenen Dimensionen in Dashboards dargestellt.

Durch die Auswertung der Stimmung nach verschiedenen Dimensionen lassen sich vielseitige Fragestellungen beantworten. Beispielsweise können dadurch beliebte und unbeliebte Charaktere, Schauspieler, Episoden oder Eigenschaften von diesen ermittelt werden. Im aktuellen Beispiel Tatort können solche Erkenntnisse genutzt werden, um zukünftige Episoden auf den Geschmack der Zuschauer bzw. der Twitter-Gemeinde anzupassen. Auch zu strategischen Entscheidungen, ob ein Tatort-Team komplett abgesetzt werden oder ein neues Team an einem neuen Spielort starten sollte, können solche Informationen herangezogen werden.

Übertragbar ist diese Anwendung nicht nur auf andere Fernsehsendungen, sondern auch auf die Bereiche Industrie und Handel. Dort kommen insbesondere Kennzahlen zum Einsatz, die von den Verkaufszahlen abhängig sind, um den Erfolg eines Produktes zu bewerten. Mit der Stimmung steht eine neue Sichtweise zur Verfügung. So kann ein Handyhersteller solch eine Anwendung nutzen, um die Stimmung nach den verschiedenen Eigenschaften bzw. Funktionen seiner Geräte auszuwerten. Das gleiche kann er auch für Geräte von Mitbewerbern machen und ähnlich wie bei den Tatort-Episoden die Erkenntnisse zur Optimierung seiner Produkte und seines Sortimentes nutzen. Auch Aussagen zur Kundenzufriedenheit, die klassischerweise nur über Umfragen möglich sind, können einfacher, häufiger und schneller gemacht werden. Zudem können Auswertungen nach dem Kunden bzw. dem Twitter-Nutzer in der Marketing-Abteilung genutzt werden, um den Kunden besser zu verstehen und individuell anzusprechen.

Dennoch ist es ebenfalls wichtig sich auch über die Grenzen solch einer Anwendung bewusst zu sein. Mit den vorgestellten Methoden und Technologien aus dem Big-Data-Bereich liegen diese weniger bei der grundsätzlichen Extraktion und Verarbeitung der Tweets oder der Integration in das DWH. Vielmehr liegt die

Schwierigkeit in der Analyse und Auswertung. So zeigt sich, dass bezüglich der Sentiment-Analyse noch Verbesserungsbedarf besteht. Für eine zuverlässige Bewertung der Tweet-Texte reicht nicht nur ein funktionierender Algorithmus. Es ist auch ein Prozess mit einer ständigen Überwachung und Verbesserung der Analyse notwendig. Für Auswertungen hinsichtlich einiger Dimensionen fehlt es zudem an Datenmaterial. Dies ist auch darauf zurück zu führen, dass nicht immer alle Stammdaten (z.B. die Region des Twitter-Nutzers) zu allen Tweets gepflegt sind. Auch die Identifikation der Zielobjekte / Entitäten, auf die sich ein Tweet bezieht, bietet analog zur Sentiment-Analyse Optimierungsbedarf. Darüber hinaus muss zu dem Produkt oder dem Thema auch entsprechend viel in den sozialen Netzwerken diskutiert werden. Nur dann besteht die Möglichkeit die Meinungen des „Twitter-Schwarms“ verallgemeinern zu können. Wenn selbst beim Sammeln von Nachrichten über einen längeren Zeitraum oder aus unterschiedlichen sozialen Netzwerken die Datenmenge zu gering ist, können extreme Meinungen einzelner das Ergebnis verfälschen und so die Auswertungen die gewünschte Aussagekraft verfehlen.

Literaturverzeichnis

Aegerter 2014

Aegerter, Janine (22.08.2014): Bier neben Pampers erhöht den Umsatz. Http://www.netzwoche.ch/News/2014/08/22/Bier-neben-Pampers-erhoeht-den-Umsatz.aspx?pa=2. Abruf 01.04.2015.

Apache Chukwa 2015

Apache (24.03.2015): Apache Chukwa. Https://chukwa.apache.org/. Abruf am 22.05.2015.

Apache Flume 2014

Apache (18.11.2014): Apache Flume. Https://flume.apache.org/. Abruf am 11.04.2015.

Apache Hadoop 2015

Apache (05.05.2015): Apache Hadoop. Http://hadoop.apache.org/. Abruf am 22.05.2015.

Apache HBase 2015

Apache (27.04.2015): Apache HBase. Http://hbase.apache.org/. Abruf am 04.04.2015.

Apache Mahout 2015

Apache (11.04.2015): Apache Mahout? Http://mahout.apache.org/. Abruf am 22.05.2015.

Apache Oozie 2014

Apache (08.12.2014): Apache Oozie Workflow Scheduler for Hadoop. Http://oozie.apache.org/. Abruf am 22.05.2015.

Apache TEZ 2015

Apache (19.05.2015): Apache TEZ. Https://tez.apache.org/. Abruf am 17.04.2015.

Apache ZooKeeper 2015

Apache (o. Datum): Apache ZooKeeper. Https://zookeeper.apache.org/. Abruf am 22.05.2015.

Apel et al. 2015

Apel, Detlef; Behme, Wolfgang; Eberlein, Rüdiger; Merighi, Christian: Datenqualität erfolgreich steuern. Praxislösungen für Business-Intelligence-Projekte. 3. Aufl. München 2015.

Bachmann et al. 2014

Bachmann, Ronald; Kemper, Guide; Gerzer, Thomas: Big Data – Fluch oder Segen? Unternehmen im Spiegel des gesellschaftlichen Wandels. Heidelberg 2014.

Bauer / Günzel 2013

Bauer, Andreas; Günzel, Holger (Hrsg.): Data Warehouse Systeme. Architektur, Entwicklung, Anwendung. 4. Aufl. Heidelberg 2013.

Bange 2010

Bange, Carsten: Werkzeuge für analytische Informationssysteme. In: Chamoni / Gluchowski 2010. S. 131 - 156.

Baron 2013

Baron, Pavlo: Big Data für IT-Entscheider. Riesige Datenmengen und moderne Technologien gewinnbringend nutzen. München 2013.

Beyer / Thoo 2014

Beyer, Mark A.; Thoo, Eric (24.07.2014): Magic Quadrant for Data Integration Tools. Http://www.gartner.com/technology/reprints.do?id=1-1YAXV15&ct=140728&st=sb. Abruf am 26.03.2015.

Biehn 2013

Biehn, Neil (07.05.2013): The missing V´s in Big Data. Viability and Value. Http://businessintelligence.com/bi-insights/the-missing-vs-in-big-data-viability-and-value/. Abruf am 08.03.2015.

Calzolari et al. 2010

Calzolari, Nicoletta; Choukri, Khalid; Maegaard, Bente; Mariani, Joseph; Odijk, Jan; Piperidis, Stelios; Rosner, Mike; Tapias, Daniel: Proceedings of the Seventh International Conference on Language Resources and Evaluation. Valletta 2010.

Chamoni / Gluchowski 2010

Chamoni, Peter; Gluchowski, Peter (Hrsg.): Analytische Informationssysteme. Business-Intelligence-Technologien und -Anwendungen. 4. Auflage. Berlin et al. 2010.

Chamoni et al. 2010

Charmoni, Peter; Beekmann, Frank; Bley, Tanja: Ausgewählte Verfahren des Data Mining. In: Chamoni / Gluchowski 2010. S. 329 - 356.

Danneman / Heimann 2014

Danneman, Nathan; Heimann, Richard: Social Media Mining with R. Deploy cutting-edge sentiment analysis techniques to real-world social media data using R. Birmingham 2014.

Dean / Ghemawat 2004

Dean, Jeffrey; Ghemawat, Sanjay: MapReduce. Simplified Data Processing on Large Clusters. In OSDI'04: Sixth Symposium on Operating System Design and Implementation. San Francisco 2004.

Dijcks 2014

Dijcks, Jean-Pierre. SQL und Hadoop - eine gemeinsame Zukunft. In DOAG News Februar 2014. S. 10-14.

Düsing 2010

Düsing, Roland: Knowledge Discovery in Databases. In: Chamoni / Gluchowski 2010. S. 281 - 306.

Faeskorn-Woyke et al. 2007

Faeskorn-Woyke, Heide; Bertelsmeier, Birgit; Riemer, Petra; Bauer, Elena: Datenbanksysteme. Theorie und Praxis mit SQL2003, Oracle und MySQL. München 2007.

Frampton 2015

Frampton, Michael: Big Data Made Easy. A Working Guide to the Complete Hadoop Toolset. New York 2015.

Freiknecht 2014

Freiknecht, Jonas: Big Data in der Praxis. Lösungen mit Hadoop, HBase und Hive. Daten speichern, aufbereiten und visualisieren. München 2014.

Frisch 2014

Frisch, Martin: Möglichkeiten zur Abfrage und Auswertung von Daten in einem Hadoop-Cluster (Projektarbeit). Köln 2014.

Gates 2011

Gates, Alan: Programming Pig. Sebastopol 2011.

Gluchowski 2010

Gluchowski, Peter: Techniken und Werkzeuge zur Unterstützung des betrieblichen Berichtswesens. In: Chamoni / Gluchowski 2010. S. 259 - 280.

Grehan 2014

Grahan, Rick (02.04.2014): Big data showdown: Cassandra vs. HBase. http://www.infoworld.com/article/2610656/database/big-data-showdown--cassandra-vs--hbase.html?page=2. Abruf am 14.04.2015.

Grimes 2013

Grimes, Seth (31.07.2013): 4 V´s for Big Data Analytics. Http://breakthroughanalysis.com/2013/07/31/4-vs-for-big-data-analytics/. Abruf am 08.03.2015.

Hahne 2010

Hahne, Michael: Mehrdimensionale Datenmodellierung für analyseorientierte Informationssysteme. In: Chamoni / Gluchowski 2010. S. 229 - 258.

Hahne 2014

Hahne, Michael: Modellierung von Business-Intelligence-Systemen. Leitfaden für erfolgreiche Projekte auf Basis flexibler Data-Warehouse-Architekturen. Heidelberg 2014.

Harrist 2015

Harrist, Margaret (o. Datum): New Technology Bridges Oracle, Hadoop, and NoSQL Data Stores. Http://www.oracle.com/us/corporate/features/big-data-sql/index.html. Abruf am 29.04.2015.

Hewitt 2011

Hewitt, Eben: Cassandra. The Definitive Guide. Sebastopol 2011.

Hopkins / Evelson 2011

Hopkins, Brian; Evelson Boris: Expand Your Digital Horizon with Big Data. Cambridge 2011.

IDC 2014

IDC (2014): Executive Summary. Data Growth, Business Opportunitites, and the IT Imperatives. Http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm. Abruf am 08.03.2015.

Inmon 2005

Inmon, William H.: Building the Data Warehouse. 4. Aufl. Indianapolis 2005.

Inmon et al. 2008

Inmon, William H.; Strauss, Derek; Neushloss, Genia: DW2.0. The Architec ture fort he Next Generation of Data Warehousing. Burlington 2008.

Kemper / Eickler 2013

Kemper, Alfons; Eickler, André: Datenbanksysteme. Eine Einführung. 9. Aufl. München 2013.

Kemper / Finger 2010

Kemper, Hans-Georg; Finger, Ralf: Datentransformation im Data Warehouse. In: Chamoni / Gluchowski 2010. S. 159 - 174.

Kimball / Ross 2013

Kimball, Ralph; Ross, Margy: The Data Warehouse Toolkit. The Definitive Guide to Dimensional Modeling. 3. Aufl. Indianapolis 2013.

King 2013

King, Stefanie: Big Data. Potential und Barrieren der Nutzung im Unternehmenskontext. München 2013.

Krcmar 2015

Krcmar, Helmut: Einführung in das Informationsmanagement. 2. Aufl. Heidelberg 2015.

Krishnan 2013

Krishnan, Krish: Data Warehousing in the Age of Big Data. Waltham 2013.

Kummerfeld 2015

Kummerfeld, Claudio (10.04.2015): Social Media-Analyse: „Apple Watch“ wird wohl ein Verkaufsschlager. http://finanzmarktwelt.de/social-media-analyse-apple-watch-wird-wohl-ein-verkaufsschlager-11079/. Abruf am 28.04.2015.

Ladner 2014

Ladner, Ralf (09.04.2014): Weltweite Datenmenge wird bis 2020 um den Faktor 10 wachsen. Internet of Things oder die Informationsflut der Dinge. http://www.funkschau.de/datacenter/artikel/107695/. Abruf am 08.05.2015.

Lang 2012

Lang, Michael: CIO-Handbuch. Best Practices für die neuen Herausforderungen des IT-Managements. Düsseldorf 2012.

Lantz 2013

Lantz, Brett: Machine Learning with R. Birmingham 2013.

Liu 2012

Liu, Bing: Sentiment Analysis and Opinion Mining. San Rafael 2012.

Lütters / Egger 2013

Lütters, Holger; Egger, Marc: “Listening is the new asking”: Social Media-Analyse in der Marktforschung. In transfer Werbeforschung & Praxis, 59 (4) 2013, S. 34-41.

Manhart 2008

Manhart, Klaus (23.02.2008): Grundlagen Business Intelligence. Datensammlung und Data Warehouses. Http://www.tecchannel.de/server/sql/1739205/business_intelligence_teil_2_datensammlung_und_data_warehouses/index2.html. Abruf am 14.03.2015.

Monhanty et al. 2013

Mohanty, Soumendra; Jagadeesh, Madhu; Srivatsa, Harsha: Big Data Imperatives. Enterprise Big Data Warehouse, BI Implementations and Analytics. New York 2013.

Müller / Lenz 2013

Müller, Roland M.; Lenz, Hans Joachim: Business Intelligence. Berlin 2013.

Murthy et al. 2014

Murthy, Arun C.; Vavilapalli, Vinod Kumar; Eadline, Doug; Niemiec, Joseph; Markham, Jeff: Apache Hadoop YARN. Moving beyond MapReduce and Batch Processing with Apache Hadoop 2. New Jersey 2014.

Nadkarni 2015

Nadkarni, Prakash (o. Datum): An Introduction to Entity-Attribute-Value Design for Generic Clinical Study Data Management Systems. Http://ycmi.med.yale.edu/nadkarni/Introduction%20to%20EAV%20systems.htm. Abruf am 07.03.2015.

Natkins 2012

Natkins, Jon (19.09.2012): How-to. Analyze Twitter Data with Apache Hadoop. Http://blog.cloudera.com/blog/2012/09/analyzing-twitter-data-with-hadoop/. Abruf am 08.10.2014.

Navrade 2008

Navrade, Frank: Strategische Planung mit Data-Warehouse-Systemen. Wiesbaden 2008.

Oracle 2010

Oracle (02.09.2010): Oracle Spatial and Graph. World Sample Data Bundle.
http://www.oracle.com/technetwork/middleware/mapviewer/downloads/navteq-data-download-168399.html. Abruf am 22.04.2015.

Oracle 2015

Oracle (o. Datum): Big Data Lite Virtual Machine. Getting Started.
Http://www.oracle.com/technetwork/database/bigdata-appliance/oracle-bigdatalite-241-2172621.html. Abruf am 07.02.2015.

Pak / Paroubek 2010

Pak, Alexander; Paroubek, Patrick: Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Calzolari et al. 2010. S. 1320 – S. 1326.

Pavlo 2013

Baron, Pavlo: Big Data für IT-Entscheider. Riesige Datenmengen und moderne Technologien gewinnbringend nutzen. München 2013.

Prajapati 2013

Prajapati, Vignesh: Big Data Analytics with R and Hadoop. Birmingham 2013.

Proctor 2008

Proctor, Leslie (12.03.2008): Corda CenterView Dashboard. Http://www.dashboardinsight.com/dashboards/tactical/corda-centerview-dashboard.aspx. Abruf am 28.03.2015.

Proffitt 2013

Proffitt, Brian (23.05.2013): Hadoop. What It Is And How It Works. Http://readwrite.com/2013/05/23/hadoop-what-it-is-and-how-it-works. Abruf am 18.02.2014.

Rath 2014

Rath, Hans Holger: Unstrukturierte Daten sind Tickets für den Geschäftserfolg. Http://www.bigdata-insider.de/analytics/articles/460995/. Abruf am 04.04.2015.

Remus et al. 2010

Remus, Robert; Quasthoff, Uwe; Heyer, Gerhard: SentiWS - a Publicly Available German-language Resource for Sentiment Analysis.
In: Calzolari et al. 2010. S. 1168-1171.

Rossak 2013

Rossak, Ines: Datenintegration. München 2013.

Russel 2014

Russel, Matthew A.: Mining the Social Web. Data Mining Facebook, Twitter Linkedin, Googl+, Github and more. 2. Aufl. Sebastopol 2014.

Schröder 2015

Schröder, Jens (05.02.2015): Top 20 der sozialen Netzwerke in Deutschland. Twitter, Pinterest und Reddit boomen. http://meedia.de/2015/02/05/top-20-der-sozialen-netzwerke-in-deutschland-twitter-pinterest-und-reddit-boomen/. Abruf am 09.03.2015.

Saake et al. 2013

Saake, Gunter; Sattler, Kai-Uwe; Heuer, Andreas: Datenbanken. Konzepte und Sprachen. 5. Aufl. Heidelberg 2013.

Speare 2015

Speare, Geoffrey (01.03.2015): ETL vs. ELT. What is the Big Difference? https://www.ironsidegroup.com/2015/03/01/etl-vs-elt-whats-the-big-difference/. Abruf am 21.03.2015.

Spies 2012

Spies. Rüdiger (22.03.2012): IDC: Die 5 wichtigsten Vs für Big Data. http://www.cio.de/a/idc-die-5-wichtigen-vs-fuer-big-data,2308215,2. Anruf am 06.04.2015.

Tan et al. 2014

Tan, Pang-Ning; Steinbach, Michael; Kumar, Vipin: Introduction to Data Mining. Edinburgh 2014.

Twitter 2015

Twitter (o. Datum): Twitter API Documentation. https://dev.twitter.com/overview/documentation. Abruf am 18.04.2015.

Vinodhini / Chandrasekaran 2012

Vinodhini, G.; Chandrasekaran, RM (2012): Sentiment Analysis and Opinion Mining: A Survey. In International Journal of Advanced Research in Computer Science and Software Engineering. Volume 2, Issue 6. S. 282-292. http://www.ijarcsse.com/docs/papers/June2012/Volume_2_issue_6/V2I600263.pdf. Abruf am 21.04.2015.

Vorhies 2013

Vorhies, Bill (2013): The Big Deal about Big Data: What´s Inside – Structured, Unstructured, and Semi-Structured Data. http://data-magnum.com/the-big-deal-about-big-data-whats-inside-structured-unstructured-and-semi-structured-data/. Abruf am 04.03.2015.

White 2015

White, Tom: Hadoop. The Definitive Guide. 4. Aufl. Sebastopol 2015.

Wikipedia 2015

Wikipedia (o. Datum): Tatort (Fernsehreihe). Http://de.wikipedia.org/wiki/Tatort_(Fernsehreihe). Abruf am 06.02.2015

Zerbes 2014

Zerbes, Gunther (25.04.2014): Datenbanktechnologien – ACID, BASE, CAP und Google Spanner. Http://www.norcom.de/de/fachartikel/datenbanktechnologien-acid-base-cap-und-google-spanner. Abruf am 13.04.2015.

Anhang A: Flume Konfigurationsdatei

flume.conf

```
# The configuration file needs to define the sources,
# the channels and the sinks.
# Sources, channels and sinks are defined per agent,
# in this case called 'TwitterAgent'

TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sources.Twitter.consumerKey = [consumerKey]
TwitterAgent.sources.Twitter.consumerSecret = [consumerSecret]
TwitterAgent.sources.Twitter.accessToken = [accessToken]
TwitterAgent.sources.Twitter.accessTokenSecret = [accessTokenSecret]
TwitterAgent.sources.Twitter.keywords = #tatort

TwitterAgent.sinks.HDFS.channel = MemChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path = hdfs://bigdatalite.localdomain:8020/user/oracle/flume/tweets_tatort/%Y/%m/%d/
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

TwitterAgent.channels.MemChannel.type = memory
TwitterAgent.channels.MemChannel.capacity = 10000
TwitterAgent.channels.MemChannel.transactionCapacity = 1000
```

Abbildung 43: Flume Konfigurationsdatei

Anhang B: Hive

Tabellen-Definitionen:

```
CREATE EXTERNAL TABLE t_tweets_tatort (
  id BIGINT,
  created_at STRING,
  source STRING,
  lang STRING,
  retweeted_status STRUCT<
    id:BIGINT,
    text:STRING,
    user:STRUCT<screen_name:STRING,name:STRING>,
    retweet_l:INT>,
  entities STRUCT<
    urls:ARRAY<STRUCT<expanded_url:STRING>>,
    user_mentions:ARRAY<STRUCT<screen_name:STRING,name:STRING>>,
    hashtags:ARRAY<STRUCT<text:STRING>>>,
  place STRUCT<
    country:STRING,
    country_code:STRING,
    id:STRING,
    full_name:STRING,
    name:STRING,
    place_type:STRING,
    bounding_box:STRUCT<
      coordinates:ARRAY<ARRAY<ARRAY<STRING>>>,
      type:STRING>>,
  coordinates STRUCT<
    coordinates:ARRAY<STRING>,
    type:STRING>,
  text STRING,
  user STRUCT<
    id:STRING,
    screen_name:STRING,
    name:STRING,
    friends_count:INT,
    followers_count:INT,
    statuses_count:INT,
    verified:BOOLEAN,
    utc_offset:INT,
    time_zone:STRING>,
  in_reply_to_screen_name STRING,
  in_reply_to_status_id BIGINT
)
PARTITIONED BY (datehour INT)
ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe'
LOCATION '/user/oracle/flume/tweets_tatort';
```

View-Definitionen:

```
CREATE OR REPLACE VIEW v_tweets_tatort
    AS
    SELECT DISTINCT
           CAST(id AS STRING)                   AS TWEET_ID,
           REGEXP_REPLACE(text,'[\\r\\n]',' ') AS TWEET_TEXT,
           created_at                           AS TWEET_DATUM,
           user.id                              AS NUTZER_ID,
           datehour                             AS TWEET_TAG,
           coordinates.coordinates[0]           AS ORT_KOORDINATE_A,
           coordinates.coordinates[1]           AS ORT_KOORDINATE_B,
           place.id                             AS ORT_ID
    FROM t_tweets_tatort
    WHERE retweeted_status IS NULL
      AND lang = 'de';

CREATE OR REPLACE VIEW V_TWEETS_ORTE AS
SELECT place.id  as ort_id,
       place.name         as ort_name,
       place.full_name    as ort_name_lang,
       place.country      as land,
       place.country_code as land_Code,
       place.place_type   as ort_type,
       coordinate[0]      as ort_koordinate_a,
       coordinate[1]      as ort_koordinate_b,
       count(*) as anzahl
 FROM t_tweets_tatort LATERAL VIEW explode(place.bounding_box.coordinates[0]) co-
ordinateTab AS coordinate
 GROUP BY place.id,
       place.name,
       place.full_name,
       place.country,
       place.country_code,
       place.place_type,
       coordinate[0],
       coordinate[1];

CREATE OR REPLACE VIEW v_tweets_tatort_nutzer
    AS
    SELECT DISTINCT
           COALESCE(user.id,'n/a') AS NUTZER_ID,
           user.screen_name        AS NUTZER_NAME,
           user.followers_count    AS ANZAHL_FOLLOWER,
           created_at              AS TWEET_DATUM
    FROM t_tweets_tatort;

CREATE OR REPLACE VIEW v_tweets_tatort_retweets
AS
SELECT retweeted_status.id as tweet_id,
       count(*) as retweet_count
FROM t_tweets_tatort
WHERE retweeted_status is not null
GROUP BY retweeted_status.id;

CREATE OR REPLACE VIEW v_tweets_tatort_for_r
AS
SELECT CONCAT(tweet_id,' ', tweet_text) AS tweet, tweet_tag as day
FROM v_tweets_tatort;
```

Anhang C: Oracle

Namenskonventionen:

Kürzel	Bedeutung
S_%	Stage-Tabelle/View
C_%	Core-Tabelle/View
M_%	Mart-Tabelle/View
%_T_%	Tabelle
%_V_%	View
M_D_%	Dimensionstabelle
M_F_%	Faktentabelle
PK_%	Package in Oracle
PR_%	Prozedur
F_%	Funktion

Tabelle 11: Oracle Namenskonventionen

Tabellen-Definitionen

```
--------------------------------------------------------
--  DDL for Table C_T_BIB_BEGRIFFE
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_BIB_BEGRIFFE"
   (    "BEGRIFF_KEY" NUMBER(36,0),
        "BEGRIFF_TEXT" VARCHAR2(250 CHAR),
        "BEGRIFF_WERT" NUMBER(6,4),
        "BEGRIFF_TYP_KEY" NUMBER(36,0),
        "LADEDATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_BIB_SYNONYME
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_BIB_SYNONYME"
   (    "SYNONYM_KEY" NUMBER(36,0),
        "BEGRIFF_KEY" NUMBER(36,0),
        "BEGRIFF_TABELLE" VARCHAR2(250 CHAR),
        "SYNONYM_TEXT" VARCHAR2(250 CHAR),
        "LADEDATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_BUNDESLAENDER
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_BUNDESLAENDER"
   (    "BUNDESLAND_KEY" NUMBER(36,0),
        "BUNDESLAND_NAME" VARCHAR2(250 CHAR),
        "LAND_KEY" NUMBER(36,0),
        "BUNDESLAND_CARTO_ID" NUMBER(36,0)
   ) ;
--------------------------------------------------------
```

```
--  DDL for Table C_T_KREISE
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_KREISE"
   (    "KREIS_SCHLUESSEL" VARCHAR2(5 CHAR),
        "KREIS_NAME" VARCHAR2(250 CHAR),
        "KREIS_KEY" NUMBER(36,0),
        "BUNDESLAND_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_LAENDER
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_LAENDER"
   (    "LAND_CODE" VARCHAR2(3 CHAR),
        "LAND_NAME" VARCHAR2(250 CHAR),
        "LAND_KEY" NUMBER(36,0),
        "LAND_CODE_ISO" VARCHAR2(3 CHAR),
        "LAND_CARTO_ID" NUMBER(36,0),
        "LAND_NAME_TWITTER" VARCHAR2(250 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_ORTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_ORTE"
   (    "ORT_KEY" NUMBER(36,0),
        "ORT_NAME" VARCHAR2(240 CHAR),
        "X_COORDINATE" NUMBER,
        "Y_COORDINATE" NUMBER,
        "BUNDESLAND_KEY" NUMBER(36,0),
        "EINWOHNERANZAHL" NUMBER(36,0),
        "ORT_POI_ID" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_ORTE_TMP
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_ORTE_TMP"
   (    "POI_ID" NUMBER(38,0),
        "ORT_NAME" VARCHAR2(240 CHAR),
        "X_COORDINATE" NUMBER,
        "Y_COORDINATE" NUMBER,
        "BUNDESLAND_CART_ID" NUMBER(36,0),
        "EINWOHNERANZAHL" NUMBER(36,0),
        "GROSSRAUM_CART_ID" NUMBER(36,0),
        "LAND_CODE" VARCHAR2(3 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_SENTIMENT_BIB
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_SENTIMENT_BIB"
   (    "BEGRIFF_KEY" NUMBER(36,0),
        "BEGRIFF_TEXT" VARCHAR2(255 CHAR),
        "BEGRIFF_TYP" VARCHAR2(50 CHAR),
        "BEGRIFF_WERT" NUMBER(6,4),
        "BEGRIFF_OPERATOR" VARCHAR2(20 CHAR),
        "LADEDATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_STAEDTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_STAEDTE"
   (    "STADT_PLZ" VARCHAR2(5 CHAR),
        "STADT_NAME" VARCHAR2(250 CHAR),
        "KREIS_KEY" NUMBER(36,0),
        "STADT_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TATORT_EPISODEN
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TATORT_EPISODEN"
   (    "EPISODEN_KEY" NUMBER(36,0),
        "TEAM_KEY" NUMBER(36,0),
        "EPISODEN_FOLGE" NUMBER(17,0),
        "EPISODEN_TITEL" VARCHAR2(255 CHAR),
        "EPISODEN_DATUM" DATE,
        "EPISODEN_KOMMENTAR" VARCHAR2(4000 CHAR),
        "EPISODEN_QUOTE_MIO" NUMBER(5,2),
        "EPISODEN_QUOTE_PROZENT" NUMBER(5,2),
        "LADEDATUM" DATE DEFAULT SYSDATE,
```

```
        "EPISODEN_ORT_KEY" NUMBER(36,0),
        "AKTIV" NUMBER(1,0) DEFAULT 1
   ) ;
   COMMENT ON COLUMN "TATORT"."C_T_TATORT_EPISODEN"."AKTIV" IS 'Aktiv fuer
Ladetag-View';
--------------------------------------------------------
--  DDL for Table C_T_TATORT_ERMITTLER
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TATORT_ERMITTLER"
   (    "ERMITTLER_KEY" NUMBER(36,0),
        "ERMITTLER_VORNAME" VARCHAR2(100 CHAR),
        "ERMITTLER_NACHNAME" VARCHAR2(100 CHAR),
        "TEAM_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TATORT_TEAMS
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TATORT_TEAMS"
   (    "TEAM_KEY" NUMBER(36,0),
        "TEAM_ID" NUMBER(17,0),
        "DEBUET_JAHR" NUMBER(4,0),
        "TEAM_ORT_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TWEETS_LAENDER
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TWEETS_LAENDER"
   (    "LAND_CODE" VARCHAR2(20 CHAR),
        "LAND_NAME" VARCHAR2(250 BYTE)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TWEETS_NUTZER
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TWEETS_NUTZER"
   (    "NUTZER_ID" VARCHAR2(50 CHAR),
        "NUTZER_NAME" VARCHAR2(100 CHAR),
        "TWEET_DATUM" DATE,
        "NUTZER_KEY" NUMBER(36,0),
        "ANZAHL_FOLLOWER_AKT" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TWEETS_NUTZER_FOLLOW_HIST
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TWEETS_NUTZER_FOLLOW_HIST"
   (    "NUTZER_HIST_KEY" NUMBER(36,0),
        "NUTZER_KEY" NUMBER(36,0),
        "ANZAHL_FOLLOWER" NUMBER(36,0),
        "TWEET_DATUM" DATE
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TWEETS_ORTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TWEETS_ORTE"
   (    "ORT_ID" VARCHAR2(50 CHAR),
        "ORT_NAME" VARCHAR2(250 CHAR),
        "ORT_KEY" NUMBER(36,0) DEFAULT "TATORT"."SEQ_TWEETS_ORTE"."NEXTVAL",
        "ORT_NAME_LANG" VARCHAR2(250 CHAR),
        "CORE_ORT_KEY" NUMBER(36,0)

 ) ;
--------------------------------------------------------
--  DDL for Table C_T_TWEETS_SENTIMENTS
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TWEETS_SENTIMENTS"
   (    "SENTIMENT_KEY" NUMBER(36,0),
        "ENTITAET_KEY" VARCHAR2(50 CHAR),
        "TWEET_KEY" NUMBER(36,0),
        "SENTIMENT_WERT" NUMBER(6,4),
        "LADEDATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table C_T_TWEETS_TWEETS
--------------------------------------------------------
  CREATE TABLE "TATORT"."C_T_TWEETS_TWEETS"
   (    "TWEET_TEXT" VARCHAR2(4000 CHAR),

       "TWEET_DATUM" DATE,
```

```
        "TWEET_ANZAHL_RETWEETS" NUMBER(6,0),
        "TWEET_ANZAHL_FAVORITEN" NUMBER(6,0),
        "LADEDATUM" DATE DEFAULT SYSDATE,
        "TWEET_ID" VARCHAR2(50 CHAR),
        "ORT_ANGABE_SICHER" VARCHAR2(1 CHAR),
        "NUTZER_KEY" NUMBER(36,0),
        "ORT_KEY" NUMBER(36,0),
        "TWEET_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table M_D_DATUM
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_D_DATUM"
   (    "DATUM_KEY" NUMBER(36,0),
        "DATUM_TAG" VARCHAR2(8 CHAR),
        "DATUM_WOCHE" VARCHAR2(6 CHAR),
        "DATUM_MONAT" VARCHAR2(6 CHAR),
        "DATUM_QUARTAL" VARCHAR2(5 CHAR),
        "DATUM_JAHR" VARCHAR2(4 CHAR),
        "DATUM_DATE"  DATE  GENERATED  ALWAYS  AS  (TO_DATE("DATUM_TAG",'YYYYMMDD'))
VIRTUAL
   ) ;
--------------------------------------------------------
--  DDL for Table M_D_ENTITAETEN
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_D_ENTITAETEN"
   (    "ENTITAET_KEY" VARCHAR2(250 CHAR),
        "ENTITAET_ORIGINAL_KEY" NUMBER(36,0),
        "ENTITAET" VARCHAR2(250 CHAR),
        "ENTITAET_TYP_KEY" NUMBER(36,0),
        "ENTITAET_TYP" VARCHAR2(250 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table M_D_ORTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_D_ORTE"
   (    "ORT_KEY" NUMBER(38,0),
        "ORT_NAME" VARCHAR2(250 CHAR),
        "ORT_EINWOHNERANZAHL" NUMBER(36,0),
        "BUNDESLAND_KEY" NUMBER(36,0),
        "BUNDESLAND_NAME" VARCHAR2(250 CHAR),
        "LAND_KEY" NUMBER(36,0),
        "LAND_NAME" VARCHAR2(250 CHAR),
        "ORT_POI_ID" NUMBER(36,0),
        "BUNDESLAND_CARTO_ID" NUMBER(36,0),
        "LAND_CODE" VARCHAR2(3 CHAR),
        "LAND_CODE_ISO" VARCHAR2(3 CHAR),
        "LAND_CARTO_ID" NUMBER(36,0),
        "STADT_KEY" NUMBER(36,0),
        "STADT_NAME" VARCHAR2(250 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table M_D_TATORT_EPISODEN
--------------------------------------------------------

   CREATE TABLE "TATORT"."M_D_TATORT_EPISODEN"
   (    "EPISODEN_KEY" NUMBER(36,0),
        "EPISODEN_TITEL" VARCHAR2(250 CHAR),
        "EPISODEN_ORT_KEY" NUMBER(36,0),
        "TEAM_KEY" NUMBER(36,0),
        "TEAM_ID" NUMBER(17,0),
        "TEAM_DEBUET" VARCHAR2(4 CHAR),
        "TEAM_ERMITTLER_NAMEN" VARCHAR2(4000 CHAR),
        "TEAM_ERMITTLER_ANZAHL" NUMBER(3,0),
        "TEAM_ORT_KEY" NUMBER(36,0),
        "EPISODEN_ERST_DATUM_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table M_D_TWEETS_NUTZER
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_D_TWEETS_NUTZER"
   (    "NUTZER_KEY" NUMBER(36,0),
        "NUTZER_NAME" VARCHAR2(250 CHAR),
        "NUTZER_ANZAHL_FOLLOWER" NUMBER(17,0)
   ) ;
--------------------------------------------------------
--  DDL for Table M_D_UHRZEIT
```

```
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_D_UHRZEIT"
   (    "UHRZEIT_KEY" NUMBER(36,0),
        "UHRZEIT" VARCHAR2(5 CHAR),
        "UHRZEIT_MINUTE" VARCHAR2(2 CHAR),
        "UHRZEIT_STUNDE" VARCHAR2(2 CHAR),
        "UHRZEIT_SENDESTATUS" VARCHAR2(250 CHAR),
        "UHRZEIT_VIERTELSTUNDE" VARCHAR2(250 CHAR),
        "UHRZEIT_HALBSTUNDE" VARCHAR2(250 CHAR),
        "UHRZEIT_VIERTELSTUNDE_TXT"   VARCHAR2(250   CHAR)   GENERATED   ALWAYS   AS
("UHRZEIT_STUNDE"||':'||RPAD(TO_CHAR((TO_NUMBER(SUBSTR("UHRZEIT_VIERTELSTUNDE",4))
-1)*15),2,'0')||'   -   '||CASE     WHEN   (TO_NUMBER("UHRZEIT_MINUTE")>=45)   THEN
(LPAD(TO_CHAR(TO_NUMBER("UHRZEIT_STUNDE")+1),2,'0')||':00')                    ELSE
("UHRZEIT_STUNDE"||':'||TO_CHAR(TO_NUMBER(SUBSTR("UHRZEIT_VIERTELSTUNDE",4))*15))
END) VIRTUAL ,
        "UHRZEIT_HALBSTUNDE_TXT"    VARCHAR2(250    CHAR)    GENERATED    ALWAYS    AS
("UHRZEIT_STUNDE"||':'||RPAD(TO_CHAR((TO_NUMBER(SUBSTR("UHRZEIT_HALBSTUNDE",4))-
1)*30),2,'0')||'   -   '||CASE     WHEN   (TO_NUMBER("UHRZEIT_MINUTE")>=30)   THEN
(LPAD(TO_CHAR(TO_NUMBER("UHRZEIT_STUNDE")+1),2,'0')||':00')                    ELSE
("UHRZEIT_STUNDE"||':'||TO_CHAR(TO_NUMBER(SUBSTR("UHRZEIT_HALBSTUNDE",4))*30))
END) VIRTUAL ,
        "UHRZEIT_STUNDE_TXT"    VARCHAR2(250    CHAR)    GENERATED    ALWAYS    AS
("UHRZEIT_STUNDE"||':00                                                        -
'||LPAD(TO_CHAR(TO_NUMBER("UHRZEIT_STUNDE")+1),2,'0')||':00') VIRTUAL
   ) ;
--------------------------------------------------------
--  DDL for Table M_F_QUOTEN
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_F_QUOTEN"
   (    "QUOTEN_KEY" NUMBER(36,0),
        "EPISODEN_KEY" NUMBER(36,0),
        "DATUM_KEY" NUMBER(36,0),
        "QUOTE_MIO" NUMBER(13,3),
        "QUOTE_PROZENZ" NUMBER(6,3)
   ) ;
--------------------------------------------------------
--  DDL for Table M_F_SENTIMENTS
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_F_SENTIMENTS"
   (    "SENTIMENT_KEY" NUMBER(36,0),
        "ENTITAET_KEY" VARCHAR2(250 CHAR),
        "DATUM_KEY" NUMBER(36,0),
        "UHRZEIT_KEY" NUMBER(36,0),
        "NUTZER_KEY" NUMBER(36,0),
        "TWEET_KEY" NUMBER(36,0),
        "SENTIMENT_WERT" NUMBER(6,4),
        "POSITIV_FLAG" NUMBER(*,0),
        "NEGATIV_FLAG" NUMBER(*,0),
        "NEUTRAL_FLAG" NUMBER(*,0),

         "ORT_KEY" NUMBER(36,0),
        "SENTIMENT_WERT_NORM" NUMBER(6,4)
   ) ;
--------------------------------------------------------
--  DDL for Table M_F_TWEETS
--------------------------------------------------------
  CREATE TABLE "TATORT"."M_F_TWEETS"
   (    "TWEET_KEY" NUMBER(36,0),
        "NUTZER_KEY" NUMBER(36,0),
        "DATUM_KEY" NUMBER(36,0),
        "UHRZEIT_KEY" NUMBER(36,0),
        "TWEET_TEXT" VARCHAR2(4000 CHAR),
        "TWEET_ANZAHL_RETWEETS" NUMBER(17,0),
        "TWEET_ANZAHL_FAVORTITEN" NUMBER(17,0),
        "ORT_KEY" NUMBER(36,0)
   ) ;
--------------------------------------------------------
--  DDL for Table P_T_BIB_BEGRIFF_TYPEN
--------------------------------------------------------
  CREATE TABLE "TATORT"."P_T_BIB_BEGRIFF_TYPEN"
   (    "BEGRIFF_TYP_KEY"                    NUMBER(36,0)                    DEFAULT
"TATORT"."SEQ_BEGRIFF_TYPEN"."NEXTVAL",
        "BEGRIFF_TYP_NAME" VARCHAR2(250 CHAR),
        "BEGRIFF_TYP_KUERZEL" VARCHAR2(250 CHAR),
        "BEGRIFF_TABELLE" VARCHAR2(250 CHAR),
        "BEGRIFF_OPERATOR" VARCHAR2(250 CHAR),
```

```
      "AENDER_DATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table P_T_BIB_BEGRIFFE
--------------------------------------------------------
  CREATE TABLE "TATORT"."P_T_BIB_BEGRIFFE"
   (    "BEGRIFF_TEXT" VARCHAR2(250 CHAR),
        "BEGRIFF_WERT" NUMBER(6,4),
        "BEGRIFF_TYP" VARCHAR2(250 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table P_T_BIB_SYNONYME
--------------------------------------------------------
  CREATE TABLE "TATORT"."P_T_BIB_SYNONYME"
   (    "SYNONYM_TEXT" VARCHAR2(250 CHAR),
        "BEGRIFF_TEXT" VARCHAR2(250 CHAR),
        "BEGRIFF_TABELLE" VARCHAR2(250 CHAR),
        "AENDER_DATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table P_T_LOG_LADELAUF
--------------------------------------------------------
  CREATE TABLE "TATORT"."P_T_LOG_LADELAUF"
   (    "LADE_KEY" NUMBER(36,0) DEFAULT "TATORT"."SEQ_LOG_LADELAUF"."NEXTVAL",
        "LADE_TYP" VARCHAR2(255 CHAR) DEFAULT 'TWEETS',
        "LADE_PARTITION" VARCHAR2(8 CHAR),
        "LADE_DATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table P_T_MAP_ORTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."P_T_MAP_ORTE"
   (    "TWITTER_ORT_ID" VARCHAR2(250 CHAR),
        "TWITTER_ORT" VARCHAR2(250 CHAR),
        "STADT_NAME" VARCHAR2(250 CHAR),
        "DATUM" DATE DEFAULT SYSDATE
   ) ;
--------------------------------------------------------
--  DDL for Table R_T_BIB_BEGRIFFE
--------------------------------------------------------
  CREATE TABLE "TATORT"."R_T_BIB_BEGRIFFE"
   (    "BEGRIFF_TEXT" VARCHAR2(4000 BYTE),
        "BEGRIFF_WERT" VARCHAR2(4000 BYTE),
        "BEGRIFF_TYP" VARCHAR2(4000 BYTE)
   ) ;

--------------------------------------------------------
--  DDL for Table R_T_TWEETS_SENTIMENTS
--------------------------------------------------------
  CREATE TABLE "TATORT"."R_T_TWEETS_SENTIMENTS"
   (    "TWEET_ID" VARCHAR2(4000 BYTE),
        "ENTITAET_KEYS" VARCHAR2(4000 BYTE),
        "SENTIMENT_WERT" VARCHAR2(4000 BYTE)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_ORTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_ORTE"
   (    "STADT_PLZ" VARCHAR2(5 CHAR),
        "STADT_NAME" VARCHAR2(250 CHAR),
        "KREIS_SCHLUESSEL" VARCHAR2(5 CHAR),
        "KREIS_NAME" VARCHAR2(250 CHAR),
        "BUNDESLAND_ID" VARCHAR2(5 CHAR),
        "BUNDESLAND_NAME" VARCHAR2(250 CHAR),
        "LAND_CODE" VARCHAR2(5 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_SENTIWS_NEG
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_SENTIWS_NEG"
   (    "WORT" VARCHAR2(100 CHAR),
        "TYP" VARCHAR2(20 CHAR),
        "WERT" NUMBER(5,4),
        "SYNONYME" VARCHAR2(4000 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_SENTIWS_POS
```

```
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_SENTIWS_POS"
   (    "WORT" VARCHAR2(100 CHAR),
        "TYP" VARCHAR2(20 CHAR),
        "WERT" NUMBER(5,4),
        "SYNONYME" VARCHAR2(4000 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_TATORT_EPISODEN
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_TATORT_EPISODEN"
   (    "FOLGE" NUMBER(17,0),
        "TITEL" VARCHAR2(250 CHAR),
        "SENDER" VARCHAR2(50 CHAR),
        "ERSTAUSSTRAHLUNG" VARCHAR2(50 CHAR),
        "ERMITTLER" VARCHAR2(250 CHAR),
        "FALL" NUMBER(17,0),
        "AUTOR" VARCHAR2(250 CHAR),
        "REGIE" VARCHAR2(250 CHAR),
        "BESONDERHEITEN" VARCHAR2(250 CHAR),
        "STADT" VARCHAR2(250 CHAR),
        "QUOTE_MIO" NUMBER(17,3),
        "QUOTE_PROZENT" NUMBER(17,3),
        "PLZ" VARCHAR2(5 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_TATORT_ERMITTLER_TEAMS
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_TATORT_ERMITTLER_TEAMS"
   (    "TEAM" NUMBER(17,0),
        "VORNAME" VARCHAR2(250 CHAR),
        "NACHNAME" VARCHAR2(250 CHAR),
        "ORT_STANDARD" VARCHAR2(250 CHAR),
        "PLZ" VARCHAR2(5 CHAR),
        "DEBUET" VARCHAR2(50 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_TWEETS_NUTZER
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_TWEETS_NUTZER"
   (    "NUTZER_ID" VARCHAR2(250 CHAR),
       "NUTZER_NAME" VARCHAR2(250 CHAR),
        "NUTZER_ANZAHL_FOLLOWER" VARCHAR2(250 CHAR),
        "TWEET_DATUM" VARCHAR2(250 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_TWEETS_ORTE
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_TWEETS_ORTE"
   (    "ORT_ID" VARCHAR2(250 CHAR),
        "ORT_NAME" VARCHAR2(250 CHAR),
        "ORT_NAME_LANG" VARCHAR2(250 CHAR),
        "ORT_TYP" VARCHAR2(250 CHAR),
        "ORT_KOORDINATE_A" VARCHAR2(250 CHAR),
        "ORT_KOORDINATE_B" VARCHAR2(250 CHAR),
        "LAND_CODE" VARCHAR2(250 CHAR),
        "LAND_NAME" VARCHAR2(250 CHAR),
        "ANZAHL" NUMBER(10,0)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_TWEETS_RETWEETS
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_TWEETS_RETWEETS"
   (    "TWEET_ID" VARCHAR2(250 CHAR),
        "TWEET_ANZAHL_RETWEETS" VARCHAR2(250 CHAR)
   ) ;
--------------------------------------------------------
--  DDL for Table S_T_TWEETS_TWEETS
--------------------------------------------------------
  CREATE TABLE "TATORT"."S_T_TWEETS_TWEETS"
   (    "TWEET_ID" VARCHAR2(250 CHAR),
        "TWEET_TEXT" VARCHAR2(4000 CHAR),
        "TWEET_DATUM" VARCHAR2(250 CHAR),
        "NUTZER_ID" VARCHAR2(250 CHAR),
        "ORT_ID" VARCHAR2(250 CHAR),
        "ORT_KOORDINATE_A" VARCHAR2(250 CHAR),
        "ORT_KOORDINATE_B" VARCHAR2(250 CHAR)
```

```
 ) ;
--------------------------------------------------------
--  DDL for Index PK_C_BIB_BEGRIFFE
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_BIB_BEGRIFFE" ON "TATORT"."C_T_BIB_BEGRIFFE"
("BEGRIFF_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_SYNONYME
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_SYNONYME" ON "TATORT"."C_T_BIB_SYNONYME"
("SYNONYM_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_BUNDESLAND
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_BUNDESLAND" ON "TATORT"."C_T_BUNDESLAENDER"
("BUNDESLAND_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_T_KREISE
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_T_KREISE" ON "TATORT"."C_T_KREISE"
("KREIS_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_LAENDER
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_LAENDER" ON "TATORT"."C_T_LAENDER"
("LAND_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_ORTE
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_ORTE" ON "TATORT"."C_T_ORTE" ("ORT_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_ORTE_TMP
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_ORTE_TMP" ON "TATORT"."C_T_ORTE_TMP"
("POI_ID");
--------------------------------------------------------
--  DDL for Index PK_C_SENTIMENT_BIB
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_SENTIMENT_BIB" ON
"TATORT"."C_T_SENTIMENT_BIB" ("BEGRIFF_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_STAEDTE
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_STAEDTE" ON "TATORT"."C_T_STAEDTE"
("STADT_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_TATORT_EPISODEN
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_TATORT_EPISODEN" ON
"TATORT"."C_T_TATORT_EPISODEN" ("EPISODEN_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_TATORT_ERMITTLER
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_TATORT_ERMITTLER" ON
"TATORT"."C_T_TATORT_ERMITTLER" ("ERMITTLER_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_TATORT_TEAMS
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_TATORT_TEAMS" ON "TATORT"."C_T_TATORT_TEAMS"
("TEAM_KEY");
--------------------------------------------------------
--  DDL for Index PK_TWEETS_LAENDER
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_TWEETS_LAENDER" ON
"TATORT"."C_T_TWEETS_LAENDER" ("LAND_CODE");
--------------------------------------------------------
--  DDL for Index PK_C_TWEETS_NUTZER
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_TWEETS_NUTZER" ON
"TATORT"."C_T_TWEETS_NUTZER" ("NUTZER_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_TWEETS_ORTE
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_C_TWEETS_ORTE" ON "TATORT"."C_T_TWEETS_ORTE"
("ORT_KEY");
--------------------------------------------------------
```

```
--  DDL for Index PK_C_TWEETS_SENTIMENTS
--------------------------------------------------------
  CREATE       UNIQUE       INDEX       "TATORT"."PK_C_TWEETS_SENTIMENTS"       ON
"TATORT"."C_T_TWEETS_SENTIMENTS" ("SENTIMENT_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_TWEETS_TWEETS
--------------------------------------------------------

  CREATE        UNIQUE        INDEX        "TATORT"."PK_C_TWEETS_TWEETS"        ON
"TATORT"."C_T_TWEETS_TWEETS" ("TWEET_KEY");
--------------------------------------------------------
--  DDL for Index PK_M_D_DATUM
--------------------------------------------------------

  CREATE    UNIQUE    INDEX    "TATORT"."PK_M_D_DATUM"    ON    "TATORT"."M_D_DATUM"
("DATUM_KEY");
--------------------------------------------------------
--  DDL for Index PK_D_ENTITAETEN
--------------------------------------------------------
  CREATE  UNIQUE  INDEX  "TATORT"."PK_D_ENTITAETEN"  ON  "TATORT"."M_D_ENTITAETEN"
("ENTITAET_KEY");
--------------------------------------------------------
--  DDL for Index PK_D_ORT
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_D_ORT" ON "TATORT"."M_D_ORTE" ("ORT_KEY");
--------------------------------------------------------
--  DDL for Index PK_D_TATORT_ERMITTLERTEAM
--------------------------------------------------------
  CREATE      UNIQUE      INDEX      "TATORT"."PK_D_TATORT_ERMITTLERTEAM"      ON
"TATORT"."M_D_TATORT_EPISODEN" ("EPISODEN_KEY");
--------------------------------------------------------
--  DDL for Index PK_F_TWEETS_NUTZER
--------------------------------------------------------

 CREATE UNIQUE INDEX "TATORT"."PK_F_TWEETS_NUTZER" ON "TATORT"."M_D_TWEETS_NUTZER"
("NUTZER_KEY");
--------------------------------------------------------
--  DDL for Index PK_D_UHRZEIT
--------------------------------------------------------
  CREATE   UNIQUE   INDEX   "TATORT"."PK_D_UHRZEIT"   ON   "TATORT"."M_D_UHRZEIT"
("UHRZEIT_KEY");
--------------------------------------------------------
--  DDL for Index PK_F_QUOTEN
--------------------------------------------------------
 CREATE    UNIQUE    INDEX    "TATORT"."PK_F_QUOTEN"    ON    "TATORT"."M_F_QUOTEN"
("QUOTEN_KEY");
--------------------------------------------------------
--  DDL for Index PK_F_SENTIMENTS
--------------------------------------------------------
  CREATE  UNIQUE  INDEX  "TATORT"."PK_F_SENTIMENTS"  ON  "TATORT"."M_F_SENTIMENTS"
("SENTIMENT_KEY");
--------------------------------------------------------
--  DDL for Index PK_F_TWEETS
--------------------------------------------------------
  CREATE    UNIQUE    INDEX    "TATORT"."PK_F_TWEETS"    ON    "TATORT"."M_F_TWEETS"
("TWEET_KEY");
--------------------------------------------------------
--  DDL for Index PK_C_BEGRIFF_TYPEN
--------------------------------------------------------
  CREATE        UNIQUE        INDEX        "TATORT"."PK_C_BEGRIFF_TYPEN"        ON
"TATORT"."P_T_BIB_BEGRIFF_TYPEN" ("BEGRIFF_TYP_KEY");
--------------------------------------------------------
--  DDL for Index PK_P_BIB_BEGRIFFE
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_P_BIB_BEGRIFFE" ON "TATORT"."P_T_BIB_BEGRIFFE"
("BEGRIFF_TEXT");
--------------------------------------------------------
--  DDL for Index PK_P_BIB_SYNONYME
--------------------------------------------------------
  CREATE UNIQUE INDEX "TATORT"."PK_P_BIB_SYNONYME" ON "TATORT"."P_T_BIB_SYNONYME"
("SYNONYM_TEXT");
--------------------------------------------------------
--  DDL for Index PK_LOG_LADELAUF
--------------------------------------------------------
  CREATE  UNIQUE  INDEX  "TATORT"."PK_LOG_LADELAUF"  ON  "TATORT"."P_T_LOG_LADELAUF"
("LADE_KEY");
--------------------------------------------------------
```

```
--  Constraints for Table C_T_BIB_BEGRIFFE
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_BIB_BEGRIFFE" MODIFY ("BEGRIFF_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_BIB_BEGRIFFE"  MODIFY  ("BEGRIFF_TYP_KEY"  NOT  NULL
ENABLE);
  ALTER  TABLE  "TATORT"."C_T_BIB_BEGRIFFE"  ADD  CONSTRAINT  "PK_C_BIB_BEGRIFFE"
PRIMARY KEY ("BEGRIFF_KEY")
  USING INDEX  ENABLE;
------------------------------------------------------
--  Constraints for Table C_T_BIB_SYNONYME
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_BIB_SYNONYME" MODIFY ("SYNONYM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_BIB_SYNONYME" MODIFY ("BEGRIFF_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_BIB_SYNONYME" ADD CONSTRAINT "PK_C_SYNONYME" PRIMARY
KEY ("SYNONYM_KEY")
  USING INDEX  ENABLE;
------------------------------------------------------
--  Constraints for Table C_T_BUNDESLAENDER
------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_BUNDESLAENDER"  MODIFY  ("BUNDESLAND_KEY"  NOT  NULL
ENABLE);
  ALTER  TABLE  "TATORT"."C_T_BUNDESLAENDER"  ADD  CONSTRAINT  "PK_C_BUNDESLAND"
PRIMARY KEY ("BUNDESLAND_KEY")
  USING INDEX  ENABLE;
  ALTER TABLE "TATORT"."C_T_BUNDESLAENDER" MODIFY ("LAND_KEY" NOT NULL ENABLE);
------------------------------------------------------
--  Constraints for Table C_T_KREISE
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_KREISE" MODIFY ("KREIS_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_KREISE"  ADD  CONSTRAINT  "PK_C_T_KREISE"  PRIMARY  KEY
("KREIS_KEY")
  USING INDEX  ENABLE;

------------------------------------------------------
--  Constraints for Table C_T_LAENDER
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_LAENDER" MODIFY ("LAND_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_LAENDER"  ADD  CONSTRAINT  "PK_C_LAENDER"  PRIMARY  KEY
("LAND_KEY")
  USING INDEX  ENABLE;
------------------------------------------------------
--  Constraints for Table C_T_ORTE
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_ORTE" MODIFY ("ORT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_ORTE" MODIFY ("ORT_NAME" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_ORTE"  ADD  CONSTRAINT  "PK_C_ORTE"  PRIMARY  KEY
("ORT_KEY")
  USING INDEX  ENABLE;
  ALTER TABLE "TATORT"."C_T_ORTE" MODIFY ("BUNDESLAND_KEY" NOT NULL ENABLE);
------------------------------------------------------
--  Constraints for Table C_T_ORTE_TMP
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_ORTE_TMP" MODIFY ("POI_ID" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_ORTE_TMP" ADD CONSTRAINT "PK_C_ORTE_TMP" PRIMARY KEY
("POI_ID")
  USING INDEX  ENABLE;
  ALTER TABLE "TATORT"."C_T_ORTE_TMP" MODIFY ("ORT_NAME" NOT NULL ENABLE);
------------------------------------------------------
--  Constraints for Table C_T_SENTIMENT_BIB
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_SENTIMENT_BIB" MODIFY ("BEGRIFF_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_SENTIMENT_BIB"  ADD  CONSTRAINT  "PK_C_SENTIMENT_BIB"
PRIMARY KEY ("BEGRIFF_KEY")
  USING INDEX  ENABLE;
------------------------------------------------------
--  Constraints for Table C_T_STAEDTE
------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_STAEDTE" MODIFY ("STADT_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_STAEDTE"  ADD  CONSTRAINT  "PK_C_STAEDTE"  PRIMARY  KEY
("STADT_KEY")
  USING INDEX  ENABLE;
------------------------------------------------------
--  Constraints for Table C_T_TATORT_EPISODEN
------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TATORT_EPISODEN"  MODIFY  ("EPISODEN_ORT_KEY"  NOT  NULL
ENABLE);
  ALTER TABLE "TATORT"."C_T_TATORT_EPISODEN" MODIFY ("AKTIV" NOT NULL ENABLE);
```

```
  ALTER TABLE "TATORT"."C_T_TATORT_EPISODEN" MODIFY ("EPISODEN_KEY" NOT NULL
ENABLE);
  ALTER TABLE "TATORT"."C_T_TATORT_EPISODEN" MODIFY ("TEAM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_TATORT_EPISODEN" ADD CONSTRAINT "PK_C_TATORT_EPISODEN"
PRIMARY KEY ("EPISODEN_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TATORT_ERMITTLER
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TATORT_ERMITTLER"  MODIFY  ("ERMITTLER_KEY"  NOT  NULL
ENABLE);
  ALTER TABLE "TATORT"."C_T_TATORT_ERMITTLER" MODIFY ("TEAM_KEY" NOT NULL ENABLE);
  ALTER       TABLE       "TATORT"."C_T_TATORT_ERMITTLER"       ADD       CONSTRAINT
"PK_C_TATORT_ERMITTLER" PRIMARY KEY ("ERMITTLER_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TATORT_TEAMS
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_TATORT_TEAMS" MODIFY ("TEAM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_TATORT_TEAMS" MODIFY ("TEAM_ORT_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TATORT_TEAMS"  ADD  CONSTRAINT  "PK_C_TATORT_TEAMS"
PRIMARY KEY ("TEAM_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TWEETS_LAENDER
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_TWEETS_LAENDER" MODIFY ("LAND_CODE" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_LAENDER"  ADD  CONSTRAINT  "PK_TWEETS_LAENDER"
PRIMARY KEY ("LAND_CODE")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TWEETS_NUTZER
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_TWEETS_NUTZER" MODIFY ("NUTZER_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_NUTZER"  ADD  CONSTRAINT  "PK_C_TWEETS_NUTZER"
PRIMARY KEY ("NUTZER_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TWEETS_NUTZER_FOLLOW_HIST
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TWEETS_NUTZER_FOLLOW_HIST"  MODIFY  ("NUTZER_HIST_KEY"
NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_NUTZER_FOLLOW_HIST"  MODIFY  ("NUTZER_KEY"  NOT
NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_NUTZER_FOLLOW_HIST"  MODIFY  ("ANZAHL_FOLLOWER"
NOT NULL ENABLE);
--------------------------------------------------------
--  Constraints for Table C_T_TWEETS_ORTE
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_TWEETS_ORTE" MODIFY ("CORE_ORT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_TWEETS_ORTE" MODIFY ("ORT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_TWEETS_ORTE" ADD CONSTRAINT "PK_C_TWEETS_ORTE" PRIMARY
KEY ("ORT_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TWEETS_SENTIMENTS
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TWEETS_SENTIMENTS"  MODIFY  ("SENTIMENT_KEY"  NOT  NULL
ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_SENTIMENTS"  MODIFY  ("ENTITAET_KEY"  NOT  NULL
ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_SENTIMENTS"  MODIFY  ("TWEET_KEY"  NOT  NULL
ENABLE);
  ALTER       TABLE       "TATORT"."C_T_TWEETS_SENTIMENTS"       ADD       CONSTRAINT
"PK_C_TWEETS_SENTIMENTS" PRIMARY KEY ("SENTIMENT_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table C_T_TWEETS_TWEETS
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_TWEETS_TWEETS" MODIFY ("NUTZER_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_TWEETS_TWEETS" MODIFY ("ORT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."C_T_TWEETS_TWEETS" MODIFY ("TWEET_KEY" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."C_T_TWEETS_TWEETS"  ADD  CONSTRAINT  "PK_C_TWEETS_TWEETS"
PRIMARY KEY ("TWEET_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_D_DATUM
```

```
--------------------------------------------------------

ALTER TABLE "TATORT"."M_D_DATUM" MODIFY ("DATUM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_D_DATUM" ADD CONSTRAINT "PK_M_D_DATUM" PRIMARY KEY
("DATUM_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_D_ENTITAETEN
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_D_ENTITAETEN" MODIFY ("ENTITAET_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_D_ENTITAETEN" ADD CONSTRAINT "PK_D_ENTITAETEN" PRIMARY
KEY ("ENTITAET_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_D_ORTE
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_D_ORTE" MODIFY ("ORT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_D_ORTE" ADD CONSTRAINT "PK_M_D_ORTE" PRIMARY KEY
("ORT_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_D_TATORT_EPISODEN
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" MODIFY ("EPISODEN_KEY" NOT NULL
ENABLE);
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" MODIFY ("EPISODEN_ORT_KEY" NOT NULL
ENABLE);
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" MODIFY ("TEAM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" ADD CONSTRAINT "PK_D_TATORT_EPISODEN"
PRIMARY KEY ("EPISODEN_KEY")
  USING INDEX  ENABLE;
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" MODIFY ("TEAM_ORT_KEY" NOT NULL
ENABLE);
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" MODIFY ("EPISODEN_ERST_DATUM_KEY" NOT
NULL ENABLE);
--------------------------------------------------------
--  Constraints for Table M_D_TWEETS_NUTZER
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_D_TWEETS_NUTZER" MODIFY ("NUTZER_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_D_TWEETS_NUTZER" ADD CONSTRAINT "PK_F_TWEETS_NUTZER"
PRIMARY KEY ("NUTZER_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_D_UHRZEIT
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_D_UHRZEIT" MODIFY ("UHRZEIT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_D_UHRZEIT" ADD CONSTRAINT "PK_D_UHRZEIT" PRIMARY KEY
("UHRZEIT_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_F_QUOTEN
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_F_QUOTEN" MODIFY ("QUOTEN_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_QUOTEN" MODIFY ("EPISODEN_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_QUOTEN" MODIFY ("DATUM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_QUOTEN" ADD CONSTRAINT "PK_F_QUOTEN" PRIMARY KEY
("QUOTEN_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table M_F_SENTIMENTS
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("SENTIMENT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("ENTITAET_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("DATUM_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("UHRZEIT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("NUTZER_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("TWEET_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" ADD CONSTRAINT "PK_F_SENTIMENTS" PRIMARY
KEY ("SENTIMENT_KEY")
  USING INDEX  ENABLE;
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" MODIFY ("ORT_KEY" NOT NULL ENABLE);
--------------------------------------------------------
--  Constraints for Table M_F_TWEETS
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_F_TWEETS" MODIFY ("TWEET_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_TWEETS" MODIFY ("NUTZER_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_TWEETS" MODIFY ("DATUM_KEY" NOT NULL ENABLE);
```

```
  ALTER TABLE "TATORT"."M_F_TWEETS" MODIFY ("UHRZEIT_KEY" NOT NULL ENABLE);
  ALTER TABLE "TATORT"."M_F_TWEETS" ADD CONSTRAINT "PK_F_TWEETS" PRIMARY KEY
("TWEET_KEY")
  USING INDEX  ENABLE;
  ALTER TABLE "TATORT"."M_F_TWEETS" MODIFY ("ORT_KEY" NOT NULL ENABLE);
--------------------------------------------------------
--  Constraints for Table P_T_BIB_BEGRIFF_TYPEN
--------------------------------------------------------
  ALTER TABLE "TATORT"."P_T_BIB_BEGRIFF_TYPEN" ADD CONSTRAINT "PK_C_BEGRIFF_TYPEN"
PRIMARY KEY ("BEGRIFF_TYP_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table P_T_BIB_BEGRIFFE
--------------------------------------------------------
  ALTER TABLE "TATORT"."P_T_BIB_BEGRIFFE" MODIFY ("BEGRIFF_TEXT" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."P_T_BIB_BEGRIFFE"  ADD  CONSTRAINT  "PK_P_BIB_BEGRIFFE"
PRIMARY KEY ("BEGRIFF_TEXT")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table P_T_BIB_SYNONYME
--------------------------------------------------------
  ALTER TABLE "TATORT"."P_T_BIB_SYNONYME" MODIFY ("SYNONYM_TEXT" NOT NULL ENABLE);
  ALTER  TABLE  "TATORT"."P_T_BIB_SYNONYME"  ADD  CONSTRAINT  "PK_P_BIB_SYNONYME"
PRIMARY KEY ("SYNONYM_TEXT")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Constraints for Table P_T_LOG_LADELAUF
--------------------------------------------------------
  ALTER TABLE "TATORT"."P_T_LOG_LADELAUF" ADD CONSTRAINT "PK_LOG_LADELAUF" PRIMARY
KEY ("LADE_KEY")
  USING INDEX  ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_BIB_BEGRIFFE
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_BIB_BEGRIFFE" ADD CONSTRAINT "FK_BEGRIFFE_BEGRIFF_TYP"
FOREIGN KEY ("BEGRIFF_TYP_KEY")
         REFERENCES "TATORT"."P_T_BIB_BEGRIFF_TYPEN" ("BEGRIFF_TYP_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_BUNDESLAENDER
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_BUNDESLAENDER"  ADD  CONSTRAINT  "FK_BUNDESLAND_LAND"
FOREIGN KEY ("LAND_KEY")
         REFERENCES "TATORT"."C_T_LAENDER" ("LAND_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_KREISE
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_KREISE" ADD CONSTRAINT "FK_C_KREI_BUNDESLAND" FOREIGN
KEY ("BUNDESLAND_KEY")
         REFERENCES "TATORT"."C_T_BUNDESLAENDER" ("BUNDESLAND_KEY") ON DELETE SET
NULL ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_ORTE
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_ORTE" ADD CONSTRAINT "FK_ORT_BUNDESLAND" FOREIGN KEY
("BUNDESLAND_KEY")
         REFERENCES "TATORT"."C_T_BUNDESLAENDER" ("BUNDESLAND_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_STAEDTE
--------------------------------------------------------
  ALTER TABLE "TATORT"."C_T_STAEDTE" ADD CONSTRAINT "FK_C_STAE_KREIS" FOREIGN KEY
("KREIS_KEY")
         REFERENCES "TATORT"."C_T_KREISE" ("KREIS_KEY") ON DELETE SET NULL ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_TATORT_EPISODEN
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TATORT_EPISODEN"  ADD  CONSTRAINT  "FK_EPISODEN_TEAM"
FOREIGN KEY ("TEAM_KEY")
         REFERENCES "TATORT"."C_T_TATORT_TEAMS" ("TEAM_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_TATORT_ERMITTLER
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TATORT_ERMITTLER"  ADD  CONSTRAINT  "FK_C_ERMI_TEAM"
FOREIGN KEY ("TEAM_KEY")
         REFERENCES "TATORT"."C_T_TATORT_TEAMS" ("TEAM_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_TWEETS_NUTZER_FOLLOW_HIST
--------------------------------------------------------
```

```
  ALTER    TABLE    "TATORT"."C_T_TWEETS_NUTZER_FOLLOW_HIST"    ADD    CONSTRAINT
"FK_NUTZER_FOLLOW_HIST_NUTZER" FOREIGN KEY ("NUTZER_KEY")

  REFERENCES "TATORT"."C_T_TWEETS_NUTZER" ("NUTZER_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_TWEETS_ORTE
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TWEETS_ORTE"  ADD  CONSTRAINT  "FK_TWEETS_ORTE_ORTE"
FOREIGN KEY ("CORE_ORT_KEY")
         REFERENCES "TATORT"."C_T_ORTE" ("ORT_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_TWEETS_SENTIMENTS
--------------------------------------------------------
  ALTER      TABLE      "TATORT"."C_T_TWEETS_SENTIMENTS"      ADD      CONSTRAINT
"FK_C_SENTIMENTS_TWEET" FOREIGN KEY ("TWEET_KEY")
         REFERENCES "TATORT"."C_T_TWEETS_TWEETS" ("TWEET_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table C_T_TWEETS_TWEETS
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."C_T_TWEETS_TWEETS"  ADD  CONSTRAINT  "FK_C_TWEE_NUTZER"
FOREIGN KEY ("NUTZER_KEY")
         REFERENCES "TATORT"."C_T_TWEETS_NUTZER" ("NUTZER_KEY") ENABLE;
  ALTER TABLE "TATORT"."C_T_TWEETS_TWEETS" ADD CONSTRAINT "FK_C_TWEE_ORT" FOREIGN
KEY ("ORT_KEY")
         REFERENCES "TATORT"."C_T_TWEETS_ORTE" ("ORT_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table M_D_TATORT_EPISODEN
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_D_TATORT_EPISODEN" ADD CONSTRAINT "FK_D_EPISODEN_ORTE"
FOREIGN KEY ("EPISODEN_ORT_KEY")
         REFERENCES "TATORT"."M_D_ORTE" ("ORT_KEY") ENABLE;
  ALTER  TABLE  "TATORT"."M_D_TATORT_EPISODEN"  ADD  CONSTRAINT  "FK_D_TEAMS_ORT"
FOREIGN KEY ("TEAM_ORT_KEY")
         REFERENCES "TATORT"."M_D_ORTE" ("ORT_KEY") ENABLE;
  ALTER       TABLE       "TATORT"."M_D_TATORT_EPISODEN"       ADD       CONSTRAINT
"FK_TATORT_EPISODEN_DATUM" FOREIGN KEY ("EPISODEN_ERST_DATUM_KEY")
         REFERENCES "TATORT"."M_D_DATUM" ("DATUM_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table M_F_QUOTEN
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_F_QUOTEN" ADD CONSTRAINT "FK_QUOTEN_DATUM" FOREIGN KEY
("DATUM_KEY")
         REFERENCES "TATORT"."M_D_DATUM" ("DATUM_KEY") ENABLE;
  ALTER TABLE "TATORT"."M_F_QUOTEN" ADD CONSTRAINT "FK_QUOTEN_EPISODE" FOREIGN KEY
("EPISODEN_KEY")
         REFERENCES "TATORT"."M_D_TATORT_EPISODEN" ("EPISODEN_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table M_F_SENTIMENTS
--------------------------------------------------------
  ALTER  TABLE  "TATORT"."M_F_SENTIMENTS"  ADD  CONSTRAINT  "FK_SENTIMENTS_DATUM"
FOREIGN KEY ("DATUM_KEY")
         REFERENCES "TATORT"."M_D_DATUM" ("DATUM_KEY") ENABLE;
  ALTER  TABLE  "TATORT"."M_F_SENTIMENTS"  ADD  CONSTRAINT  "FK_SENTIMENTS_ENTITAET"
FOREIGN KEY ("ENTITAET_KEY")
         REFERENCES "TATORT"."M_D_ENTITAETEN" ("ENTITAET_KEY") ENABLE;
  ALTER  TABLE  "TATORT"."M_F_SENTIMENTS"  ADD  CONSTRAINT  "FK_SENTIMENTS_NUTZER"
FOREIGN KEY ("NUTZER_KEY")
         REFERENCES "TATORT"."M_D_TWEETS_NUTZER" ("NUTZER_KEY") ENABLE;
  ALTER TABLE "TATORT"."M_F_SENTIMENTS" ADD CONSTRAINT "FK_SENTIMENTS_ORT" FOREIGN
KEY ("ORT_KEY")
         REFERENCES "TATORT"."M_D_ORTE" ("ORT_KEY") ENABLE;
  ALTER  TABLE  "TATORT"."M_F_SENTIMENTS"  ADD  CONSTRAINT  "FK_SENTIMENTS_TWEETS"
FOREIGN KEY ("TWEET_KEY")
         REFERENCES "TATORT"."M_F_TWEETS" ("TWEET_KEY") ENABLE;
  ALTER  TABLE  "TATORT"."M_F_SENTIMENTS"  ADD  CONSTRAINT  "FK_SENTIMENTS_UHRZEIT"
FOREIGN KEY ("UHRZEIT_KEY")
         REFERENCES "TATORT"."M_D_UHRZEIT" ("UHRZEIT_KEY") ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table M_F_TWEETS
--------------------------------------------------------
  ALTER TABLE "TATORT"."M_F_TWEETS" ADD CONSTRAINT "FK_F_TWEETS_ORT" FOREIGN KEY
("ORT_KEY")
         REFERENCES "TATORT"."M_D_ORTE" ("ORT_KEY") ENABLE;
  ALTER TABLE "TATORT"."M_F_TWEETS" ADD CONSTRAINT "FK_TWEETS_DATUM" FOREIGN KEY
("DATUM_KEY")
         REFERENCES "TATORT"."M_D_DATUM" ("DATUM_KEY") ENABLE;
```

```
  ALTER TABLE "TATORT"."M_F_TWEETS" ADD CONSTRAINT "FK_TWEETS_NUTZER" FOREIGN KEY
("NUTZER_KEY")
          REFERENCES "TATORT"."M_D_TWEETS_NUTZER" ("NUTZER_KEY") ENABLE;
  ALTER TABLE "TATORT"."M_F_TWEETS" ADD CONSTRAINT "FK_TWEETS_UHRZEIT" FOREIGN KEY
("UHRZEIT_KEY")
          REFERENCES "TATORT"."M_D_UHRZEIT" ("UHRZEIT_KEY") ENABLE;
```

View-Definitionen

```
--------------------------------------------------------
--  DDL for View C_V_LAENDER
--------------------------------------------------------
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "TATORT"."C_V_LAENDER" ("LAND_NAME",
"LAND_CODE_ISO", "LAND_CODE", "LAND_CARTO_ID", "LAND_NAME_TWITTER") AS
  SELECT     MIN(LAND_NAME),     LAND_CODE_ISO,     LAND_CODE,     LAND_CARTO_ID,
LAND_NAME_TWITTER FROM (
  SELECT NVL(c.COUNTRY_NAME,ac.name) AS LAND_NAME,
    ac.iso_country_code AS LAND_CODE_ISO,
    NVL(c.COUNTRY_CODE_2, SUBSTR(ac.iso_country_code,1,2))  AS LAND_CODE,
    ac.CARTO_ID         AS LAND_CARTO_ID,
    tl.LAND_NAME        AS LAND_NAME_TWITTER
  FROM mvdemo2.gc_country_profile c
  RIGHT OUTER JOIN mvdemo2.wom_area_country_gen ac
  ON c.country_code_3 = ac.iso_country_code
  INNER JOIN c_t_tweets_laender tl
  ON  NVL(c.COUNTRY_CODE_2, SUBSTR(ac.iso_country_code,1,2))  = tl.land_code
  WHERE  ac.name_label  IS  NOT  NULL    AND  (c.COUNTRY_NAME  IS  NOT  NULL  OR
ac.name_label  =  UPPER(  tl.LAND_NAME)))  GROUP  BY  LAND_CODE_ISO,  LAND_CODE,
LAND_CARTO_ID, LAND_NAME_TWITTER

  UNION ALL

  SELECT 'NOT AVAILABLE'AS LANDE_NAME,
  'n/a' AS LAND_CODE_ISO,
  'n/a' AS LAND_CODE,
  NULL AS LAND_CARTO_ID,
  'not available'AS LANDE_NAME_TWITTER FROM DUAL;
--------------------------------------------------------
--  DDL for View C_V_BIB_KONTEXT
--------------------------------------------------------
  CREATE   OR   REPLACE   FORCE   EDITIONABLE   VIEW   "TATORT"."C_V_BIB_KONTEXT"
("ENTITAET_KEY",     "ENTITAET_TEXT",     "ORIGINAL_KEY",     "ENTITAET_TYP_KEY",
"ENTITAET_TYP_NAME", "SYNONYM_FLAG") AS
  SELECT B.BEGRIFF_TYP_KUERZEL
    || '_'
    || T.BEGRIFF_KEY            AS ENTITAET_KEY ,
    f_norm_text(T.KONTEXT_TEXT) AS ENTITAET_TEXT ,
    T.BEGRIFF_KEY               AS ORIGINAL_KEY ,
    B.BEGRIFF_TYP_KEY           AS ENTITAET_TYP_KEY ,
    B.BEGRIFF_TYP_NAME          AS ENTITAET_TYP_NAME ,
    NULL                        AS SYNONYM_FLAG
  FROM C_V_BIB_KONTEXT_OHNE_SYN T
  INNER JOIN P_T_BIB_BEGRIFF_TYPEN B
  ON T.BEGRIFF_TABELLE     = B.BEGRIFF_TABELLE
  WHERE B.BEGRIFF_TYP_NAME = T.BEGRIFF_TYP_NAME
  OR T.BEGRIFF_TYP_NAME   IS NULL
  UNION
  SELECT B.BEGRIFF_TYP_KUERZEL
    || '_'
    || S.BEGRIFF_KEY            AS KONTEXT_KEY ,
    f_norm_text(S.SYNONYM_TEXT) AS ENTITAET_TEXT ,
    T.BEGRIFF_KEY               AS ORIGINAL_KEY ,
    B.BEGRIFF_TYP_KEY           AS ENTITAET_TYP_KEY ,
    B.BEGRIFF_TYP_NAME          AS ENTITAET_TYP_NAME ,
    'X'                         AS SYNONYM_FLAG
  FROM C_V_BIB_KONTEXT_OHNE_SYN T
  INNER JOIN P_T_BIB_BEGRIFF_TYPEN B
  ON T.BEGRIFF_TABELLE = B.BEGRIFF_TABELLE
  INNER JOIN C_T_BIB_SYNONYME S
  ON T.BEGRIFF_KEY         = S.BEGRIFF_KEY
  AND B.BEGRIFF_TABELLE    = S.BEGRIFF_TABELLE
  WHERE B.BEGRIFF_TYP_NAME = T.BEGRIFF_TYP_NAME
  OR T.BEGRIFF_TYP_NAME   IS NULL
  UNION
  SELECT BT.BEGRIFF_TYP_KUERZEL
    || '_'
    || k.BEGRIFF_KEY            AS KONTEXT_KEY ,
    f_norm_text(B.BEGRIFF_TEXT) AS ENTITAET_TEXT ,
    k.BEGRIFF_KEY               AS ORIGINAL_KEY ,
    BT.BEGRIFF_TYP_KEY          AS ENTITAET_TYP_KEY ,
    BT.BEGRIFF_TYP_NAME         AS ENTITAET_TYP_NAME ,
    'X'                         AS SYNONYM_FLAG
  FROM c_t_bib_begriffe b
```

```
  INNER JOIN p_t_bib_begriff_typen bt
  ON b.begriff_typ_key    = bt.begriff_typ_key
  AND bt.begriff_typ_name = 'EPISODEN_REFERENZ'
  INNER JOIN c_t_tatort_episoden e
  ON 1=1
  INNER JOIN c_v_tweets_lade_tag l
  ON TO_CHAR(e.episoden_datum,'YYYYMMDD') = l.tag
  INNER JOIN c_v_bib_kontext_ohne_syn k
  ON e.episoden_key     = k.begriff_key
  AND k.begriff_tabelle = 'C_T_TATORT_EPISODEN';
--------------------------------------------------------
--  DDL for View S_V_SENTIWS_SYN
--------------------------------------------------------
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "TATORT"."S_V_SENTIWS_SYN" ("WORT",
"SYNONYM_TEXT") AS
  SELECT DISTINCT WORT, REGEXP_SUBSTR(SYNONYME,'[^, ]+',1,RN) AS SYNONYM_TEXT
FROM s_t_sentiws_neg
CROSS JOIN (SELECT ROWNUM AS RN FROM (SELECT MAX(REGEXP_COUNT(SYNONYME,',')+1) MX
FROM s_t_sentiws_neg)
CONNECT BY LEVEL <= MX)
WHERE REGEXP_SUBSTR(SYNONYME,'[^, ]+',1,RN) IS NOT NULL
UNION
  SELECT DISTINCT WORT, REGEXP_SUBSTR(SYNONYME,'[^, ]+',1,RN) AS SYNONYM_TEXT
FROM s_t_sentiws_pos
CROSS JOIN (SELECT ROWNUM AS RN FROM (SELECT MAX(REGEXP_COUNT(SYNONYME,',')+1) MX
FROM s_t_sentiws_pos)
CONNECT BY LEVEL <= MX)
WHERE REGEXP_SUBSTR(SYNONYME,'[^, ]+',1,RN) IS NOT NULL;
--------------------------------------------------------
--  DDL for View C_V_BUNDESLAENDER
--------------------------------------------------------
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "TATORT"."C_V_BUNDESLAENDER"
("BUNDESLAND_NAME", "LAND_CODE", "LAND_CODE_ISO", "BUNDESLAND_CARTO_ID",
"GEOMETRY") AS
  SELECT
  BUNDESLAND_NAME,
  LAND_CODE,
  LAND_CODE_ISO,
  BUNDESLAND_CARTO_ID,
  GEOMETRY
FROM
  (SELECT
    NAME      AS BUNDESLAND_NAME,
    LAND_CODE,
    LAND_CODE_ISO,
    CARTO_ID AS BUNDESLAND_CARTO_ID,
    GEOMETRY,
    RANK()OVER(PARTITION BY NAME ORDER BY SQKM DESC) AS RANG
  FROM mvdemo2.wom_area a
  INNER JOIN C_V_LAENDER l
  ON a.iso_country_code = l.LAND_CODE_ISO
  WHERE feature_type    = 909996
  )
WHERE RANG = 1
UNION ALL
SELECT
    LAND_CODE || '_OTHER' AS BUNDESLAND_NAME,
    LAND_CODE,
    LAND_CODE_ISO,
    NULL AS BUNDESLAND_CARTO_ID,
    NULL AS GEOMETRY
 FROM C_V_LAENDER;


--------------------------------------------------------
--  DDL for View S_V_TWEETS_SENTIMENTS
--------------------------------------------------------
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "TATORT"."S_V_TWEETS_SENTIMENTS"
("TWEET_ID", "ENTITAET_KEY", "SENTIMENT_WERT") AS
  SELECT DISTINCT TWEET_ID,
    REGEXP_SUBSTR(ENTITAET_KEYS,'[^, ]+',1,RN) AS ENTITAET_KEY,
    SENTIMENT_WERT
  FROM r_t_tweets_Sentiments
  CROSS JOIN
    (SELECT ROWNUM AS RN
    FROM
      (SELECT MAX(REGEXP_COUNT(ENTITAET_KEYS,',')+1) MX FROM r_t_tweets_sentiments
```

```
      )
      CONNECT BY LEVEL <= MX
    )
  WHERE REGEXP_SUBSTR(ENTITAET_KEYS,'[^, ]+',1,RN) IS NOT NULL
  UNION
  --Tweets die keine Entitaet-Keys aber einen Sentimentwert haben
  SELECT TWEET_ID,
    ENTITAET_KEYS AS ENTITAET_KEY,
    SENTIMENT_WERT
  FROM r_t_tweets_Sentiments
  WHERE TRIM(ENTITAET_KEYS) IS NULL
  AND SENTIMENT_WERT        != '0';
--------------------------------------------------------
--  DDL for View C_V_TWEETS_ORTE_NA
--------------------------------------------------------
  CREATE   OR   REPLACE   FORCE   EDITIONABLE   VIEW   "TATORT"."C_V_TWEETS_ORTE_NA"
("ORT_ID", "ORT_NAME", "ORT_NAME_LANG", "ORT_KEY") AS
  SELECT 'n/a' ort_id,
    'not available' ort_name,
    'not available' ort_name_lang,
    ort_key
  FROM c_t_orte where ort_name = 'not available'
  UNION
  SELECT t1.ort_id,
     t1.ort_name,
     t1.ort_name_lang,
     t2.ort_key
  FROM  s_t_tweets_orte  t1,c_t_orte  t2  where  t2.ort_name  =  'not  available'  and
t1.ort_id not in (select ort_id from c_v_tweets_orte);
--------------------------------------------------------
--  DDL for View C_V_TWEETS_ORTE
--------------------------------------------------------
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "TATORT"."C_V_TWEETS_ORTE" ("ORT_ID",
"ORT_NAME", "ORT_NAME_LANG", "ORT_KEY") AS
  SELECT ort_id,
    ort_name,
    ort_name_lang,
    ort_key
  FROM
    (SELECT tweetorte.ort_id,
      tweetorte.ort_name,
      tweetorte.ort_name_lang,
      orte.ort_key,
      rank()over(partition  BY  ort_id  order  by  power(max_a-orte.x_coordinate,2)  +
power(x_coordinate-min_a,2)     +     power(max_b-orte.y_coordinate,2)     +     pow-
er(y_coordinate-min_b,2),
      CASE
        WHEN orte.ort_name = tweetorte.ort_name
        THEN 1
        WHEN tweetorte.ort_name_lang = orte.ort_name
        THEN 2
        WHEN tweetorte.ort_name_lang LIKE '%'
          || orte.ort_name
          || '%'
        THEN 3
        ELSE DBMS_RANDOM.VALUE(4,5)
      END) rang

    FROM
      (SELECT ort_id,
        land_code,
        ort_name,
        ort_name_lang,
        MAX(to_number(SUBSTR(ort_koordinate_a,1,10),'999.99999999')) AS max_a,
        MIN(to_number(SUBSTR(ort_koordinate_a,1,10),'999.99999999')) AS min_a,
        MAX(to_number(SUBSTR(ort_koordinate_b,1,10),'999.99999999')) AS max_b,
        MIN(to_number(SUBSTR(ort_koordinate_b,1,10),'999.99999999')) AS min_b
      FROM s_t_tweets_orte
      GROUP BY ort_id,
        land_code,
        ort_name,
        ort_name_lang
      ) tweetorte,
      C_T_ORTE orte
      INNER JOIN c_t_bundeslaender bl ON orte.bundesland_key = bl.bundesland_key

INNER JOIN c_t_laender l ON bl.land_key = l.land_key
```

```
    WHERE tweetorte.land_Code = l.land_code
    AND     orte.x_coordinate     BETWEEN     TRUNC(tweetorte.min_a,2)-0.1     AND
CEIL(tweetorte.max_a * 100+10)/100
    AND     orte.y_coordinate     BETWEEN     TRUNC(tweetorte.min_b,2)-0.1     AND
CEIL(tweetorte.max_b * 100+10)/100
    )
  WHERE rang = 1;
--------------------------------------------------------
--  DDL for View C_V_BIB_BEGRIFFE
--------------------------------------------------------
  CREATE   OR   REPLACE   FORCE   EDITIONABLE   VIEW   "TATORT"."C_V_BIB_BEGRIFFE"
("BEGRIFF_TEXT", "BEGRIFF_WERT", "BEGRIFF_OPERATOR", "BEGRIFF_TYP_NAME") AS
  SELECT f_norm_text(U.BEGRIFF_TEXT),
    CASE  WHEN  BT.BEGRIFF_OPERATOR  =  'SUBTRACT'  THEN  U.BEGRIFF_WERT  *  (-1)  ELSE
U.BEGRIFF_WERT END BEGRIFF_WERT,
    BT.BEGRIFF_OPERATOR,
    BT.BEGRIFF_TYP_NAME
  FROM
    (SELECT B.BEGRIFF_TEXT ,
      B.BEGRIFF_WERT ,
      B.BEGRIFF_TYP_KEY
    FROM C_T_BIB_BEGRIFFE B
    UNION
    SELECT S.SYNONYM_TEXT AS BEGRIFF_TEXT ,
      B.BEGRIFF_WERT ,
      B.BEGRIFF_TYP_KEY
    FROM C_T_BIB_SYNONYME S
    INNER JOIN C_T_BIB_BEGRIFFE B
    ON S.BEGRIFF_KEY        = B.BEGRIFF_KEY
    WHERE S.BEGRIFF_TABELLE = 'C_T_BIB_BEGRIFFE'
    ) U
  INNER JOIN P_T_BIB_BEGRIFF_TYPEN BT
  ON U.BEGRIFF_TYP_KEY = BT.BEGRIFF_TYP_KEY;
--------------------------------------------------------
--  DDL for View C_V_TWEETS_LADE_TAG
--------------------------------------------------------
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "TATORT"."C_V_TWEETS_LADE_TAG" ("TAG",
"HDFS_LOCATION") AS
  SELECT NVL(TAG,'N/A') AS TAG,
    '/user/oracle/flume/tweets_tatort/'
    || SUBSTR(TAG,1,4)
    || '/'
    || SUBSTR(TAG,5,2)
    || '/'
    || SUBSTR(TAG,7,2) AS HDFS_LOCATION
  FROM
    (SELECT TO_CHAR(MIN(episoden_datum),'YYYYMMDD') AS tag
    FROM c_t_tatort_episoden
    WHERE aktiv = 1 and episoden_datum NOT IN
      (SELECT
NVL(to_date(lade_partition,'YYYYMMDD'),TO_DATE('19000101','YYYYMMDD'))
      FROM p_t_log_ladelauf
      ) and episoden_datum < sysdate
    );
--------------------------------------------------------
--  DDL for View C_V_BIB_KONTEXT_OHNE_SYN
--------------------------------------------------------
  CREATE  OR  REPLACE  FORCE  EDITIONABLE  VIEW  "TATORT"."C_V_BIB_KONTEXT_OHNE_SYN"
("BEGRIFF_KEY", "KONTEXT_TEXT", "BEGRIFF_TABELLE", "BEGRIFF_TYP_NAME") AS
  SELECT TEAM_KEY     AS BEGRIFF_KEY ,
       O.ORT_NAME         AS KONTEXT_TEXT ,
      'C_T_TATORT_TEAMS' AS BEGRIFF_TABELLE,
      NULL BEGRIFF_TYP_NAME
    FROM C_T_TATORT_TEAMS T
    LEFT OUTER JOIN c_t_orte o
    ON T.TEAM_ORT_KEY = O.ORT_KEY
    UNION ALL
    SELECT EPISODEN_KEY     AS BEGRIFF_KEY ,
      EPISODEN_TITEL        AS KONTEXT_TEXT ,
      'C_T_TATORT_EPISODEN' AS BEGRIFF_TABELLE,
      NULL BEGRIFF_TYP_NAME
    FROM C_T_TATORT_EPISODEN
    UNION ALL
    SELECT ERMITTLER_KEY     AS BEGRIFF_KEY ,
      ERMITTLER_NACHNAME     AS KONTEXT_TEXT ,
      'C_T_TATORT_ERMITTLER' AS BEGRIFF_TABELLE,
```

```
NULL BEGRIFF_TYP_NAME
    FROM C_T_TATORT_ERMITTLER
    UNION ALL
    SELECT BEGRIFF_KEY     AS BEGRIFF_KEY ,
      BEGRIFF_TEXT         AS KONTEXT_TEXT ,
      b.BEGRIFF_TABELLE    AS BEGRIFF_TABELLE,
      BEGRIFF_TYP_NAME
    FROM    C_T_BIB_BEGRIFFE    a    INNER    JOIN    P_T_BIB_BEGRIFF_TYPEN    b    ON
a.BEGRIFF_TYP_KEY = b.BEGRIFF_TYP_KEY AND BEGRIFF_TYP_NAME = 'CONTEXT_GEN';
```

Funktionen

```
--------------------------------------------------------
--  DDL for Function F_GET_TIMEZONE_OFFSET
--------------------------------------------------------
  CREATE OR REPLACE EDITIONABLE FUNCTION "TATORT"."F_GET_TIMEZONE_OFFSET" (P_DATUM
DATE DEFAULT SYSDATE)
RETURN VARCHAR2
IS
  v_diff_months INTEGER;
  v_diff_days   INTEGER;
BEGIN
  v_diff_months:= TRUNC(MONTHS_BETWEEN(P_DATUM,SYSDATE));
  v_diff_days   := ROUND(p_datum-ADD_MONTHS(SYSDATE,v_diff_months));
  RETURN TO_CHAR(CURRENT_TIMESTAMP + INTERVAL '1'    DAY * v_diff_days + INTERVAL
'1' MONTH * v_diff_months,'TZH');
END F_GET_TIMEZONE_OFFSET;
/
--------------------------------------------------------
--  DDL for Function F_NORM_TEXT
--------------------------------------------------------
  CREATE OR REPLACE EDITIONABLE FUNCTION "TATORT"."F_NORM_TEXT" (P_TEXT VARCHAR2)
RETURN VARCHAR2
IS
v_result VARCHAR2(4000 CHAR);
BEGIN
  v_result            :=            REPLACE(REPLACE(REPLACE(REPLACE(LOWER(P_TEXT),
'ä','a'),'ö','o'),'ü','u'),'ß','ss');
  return v_result;
END f_norm_text;
```

Anhang D: Code R

R-Funktionen

```
#Abfrage des zu verarbeitenden Ladetags
f_get_loading_day<-function(){
  result <- sql("select tag from c_v_tweets_lade_tag"
                , username = "tatort"
                , password = "tatort")
  return (result)
}
#Bilden von Wortstämmen
f_stemDocument <- function(x, language  = "de")
{
  PlainTextDocument(paste(stemDocument(unlist(strsplit(as.character(x),           "
")),language = language), collapse = " "))
}
#Laden von Wörterbüchern aus der Datenbank
f_get_begriffe<-function(p_begriff_typ){
  p_begriff_typ <- paste("'",p_begriff_typ,"'", sep = "",collapse = ",")
  result  <-  sql(paste("select  begriff_text,  begriff_wert,  begriff_typ_name  from
c_v_bib_begriffe where begriff_typ_name IN (", p_begriff_typ, ")")
                , username = "tatort"
                , password = "tatort")
  return (result)
}
#Laden der Kontextinformationen aus der Datenbank
f_get_context<-function(){
  result <- sql("select entitaet_key, entitaet_text from c_v_bib_kontext"
                , username = "tatort"
                , password = "tatort")
  return (result)
}
#Ermittlung und Zuordnung von Sentiment-Werten zu einem Tweet
f_calc_sentiment<-function(x, bib.df, ngram){
  tokens <- NGramTokenizer(x, Weka_control(min = ngram, max = ngram))
  matches <- match(tokens, bib.df$BEGRIFF_TEXT)
  score <- sum(bib.df$BEGRIFF_WERT[matches[!is.na(matches)]])
  score <- score + as.numeric(meta(x, tag = "SENTIMENT_SCORE"))
  meta(x,'SENTIMENT_SCORE') <- score
  return (x)
}
#Zuordnung von Kontext-Ids zu einem Tweet
f_match_context <- function(x, context.df, ngram = 1){
  tokens <- NGramTokenizer(x, Weka_control(min = ngram, max = ngram))
  matches <- match(tokens, context.df$text)
  matches <- context.df$entity_key[matches[!is.na(matches)]]
  matches <-  paste(matches, collapse=", ")
  if (length(meta(x, tag = "ENTITY_KEYS"))>0) {
    matches <-  paste(as.character(meta(x, tag = "ENTITY_KEYS"))
                      ,matches,collapse = ", ")
  }
  meta(x, tag = "ENTITY_KEYS") <- matches
  return (x)}
```

Skripte

```
#Skript loadLibraries: Laden der Bibliotheken
#library(ORCH)
library(ORE)
library(ora)
#Library for Text Mining
library(tm)
#library for stems
library(SnowballC)
#library for wordcloud
library(wordcloud)
#library for word-function (split tweet in id and text)
library(stringr)
#library for ddply
library(plyr)
#library for replacement functions gsubfn
library(gsubfn)
#RWeka for NGram and Word-Tokenizer
library(RWeka)

#Skript conntectHive: Aufbau einer Verbindung nach Hive
ore.disconnect()
ore.connect(type="HIVE")
#Da die Daten in Hive nicht indiziert werden, wird eine Warnung geworfen, die mit
folgender Option abgeschaltet wird
options(ore.warn.order=FALSE)
ore.attach()

#Skript prepareTweets: Importiert die Tweets aus Hive und führt verschiedene Be-
reinigungen/Aufbereitungen aus
#Abfrage des Ladetags
datum<-f_get_loading_day()[1,1]
#Hive-View in R bekannt machen
ore.sync(table="v_tweets_tatort_for_r")
ore.attach()
#Laden der Hive-Daten aus der View nach R
tmp <- v_tweets_tatort_for_r$tweet[v_tweets_tatort_for_r$day==datum]
tweets_tatort<-ore.pull(tmp)
tweets_tatort.df<-
data.frame(tweet_id=word(tweets_tatort,1),tweet_text=word(tweets_tatort,2,-1),
stringsAsFactors=FALSE )
#Anlegen eines Korpus
tweets_tatort_corpus                                <-                              Cor-
pus(VectorSource(tweets_tatort.df$tweet_text[1:length(tweets_tatort.df[,1])]))

#Bereinigungen
tweets_tatort_corpus <- tm_map(tweets_tatort_corpus, tolower)
#Stoppmusster entfernen: Urls, Hashtags (#), Usernames (@)
pattern <- paste(f_get_begriffe("STOPPATTERN")[,1], collapse="|")
tweets_tatort_corpus       <-       tm_map(tweets_tatort_corpus,       function(x)
gsub(paste('(',pattern,')\\S*( |$)',sep = ""),' ',x))
#Sonderzeichen und Zahlen entfernen
tweets_tatort_corpus       <-       tm_map(tweets_tatort_corpus,       function(x)
gsub("[[:punct:]]", " ",x))
tweets_tatort_corpus <- tm_map(tweets_tatort_corpus, removeNumbers)
#Stoppwörter entfernen
my_stopwords <- f_get_begriffe("STOPWORD")[,1]
tweets_tatort_corpus <- tm_map(tweets_tatort_corpus, removeWords, my_stopwords)
#Überflüssige Leezeichen entfernen
tweets_tatort_corpus <- tm_map(tweets_tatort_corpus, function(x) str_trim(x))
tweets_tatort_corpus <- tm_map(tweets_tatort_corpus, stripWhitespace)
#Wortstämme bilden (für die deutsche Sprache)
tweets_tatort_corpus  <-  tm_map(tweets_tatort_corpus,  function(x)  f_stemDocument
(x,language = "de"))
#Umlaute auflösen (ä,ö,ü,ß)
tweets_tatort_corpus       <-       tm_map(tweets_tatort_corpus,       function(x)
gsubfn(".",list("ä"="ae","ö"="oe","ü"="ue","ß"="ss"),x))

#Speicherung  der  Twitter-Id  in  den  Korpus-Metadaten.  Keine  Parallelisierung
(mc.cores=1) für eine korrekte Iteration.
i<-0
tweets_tatort_corpus <- tm_map(tweets_tatort_corpus, function(x) {
  i <<- i + 1
  meta(x, "TWEET_ID") <- tweets_tatort.df[i,1]
  meta(x, "SENTIMENT_SCORE") <- 0
```

```
  meta(x, "ENTITY_KEYS") <- ""
  x
},mc.cores=1)

#Skript analyse: Durchführung der eigentlichen Analyse
#1. Berechnung der Sentiment-Werte
#2. Zuordnung von Entitäten
#Teilen des Korpus für eine sicherere Ausführung (Arbeitsspeicher läuft sonst
voll)
len <- length(tweets_tatort_corpus)
mid <- floor(len/2)

tweets_tatort_corpus_a<-tweets_tatort_corpus[1:mid]
tweets_tatort_corpus_b<-tweets_tatort_corpus[(mid+1):len]

#Importieren der benötigten Bibliotheken
bib.df                                                                  <-
f_get_begriffe(c("PHRASE_NEG","PHRASE_POS","NEGATION","INCREASER","DECREASER"))
bib.df$begriff.text <- stemDocument(bib.df$begriff.text, language = "de")

bibneg.df <- bib.df[bib.df$begriff.typ.name == "NEGATION",]
bibcrease.df <- bib.df[bib.df$begriff.typ.name %in% c("INCREASER","DECREASER"),]
bib.df <- bib.df[bib.df$begriff.typ.name %in% c("PHRASE_POS","PHRASE_NEG"),]
bib.df <- aggregate(bib.df$begriff.wert, by=list(bib.df$begriff.text), FUN=mean)
colnames(bib.df)<- c("begriff.text","begriff.wert")

#Erstellung von Bigrammen
bibbi.df <- merge(bib.df,bibneg.df,by = NULL)
#Neutralisierung von Unigramme, die in gefundenen Bigrammen enthalten sind: Bei
Bigrammen mit NEGATION (y) wird 2 mal multipliziert und für INCREASER/DECREASER
(y) der Unigram-Wert zusätzlich abgezogen (x).
bibbigramme_1.df                                        <-                      da-
ta.frame(paste(bibbi.df$begriff.text.y,bibbi.df$begriff.text.x,sep="
"),bibbi.df$begriff.wert.x*bibbi.df$begriff.wert.y*2)
colnames(bibbigramme_1.df)<- c("BEGRIFF_TEXT","BEGRIFF_WERT")
bibbi.df <- merge(bib.df,bibcrease.df,by = NULL)
bibbigramme_2.df                                    <-                      da-
ta.frame(paste(bibbi.df$begriff.text.y,bibbi.df$begriff.text.x,sep="
"),(bibbi.df$begriff.wert.x*bibbi.df$begriff.wert.y)-bibbi.df$begriff.wert.x)
colnames(bibbigramme_2.df)<- c("BEGRIFF_TEXT","BEGRIFF_WERT")
bibbigramme.df <-rbind(bibbigramme_1.df,bibbigramme_2.df)

#Aufbau Trigramme (y=Negation, ''=increaser/decreaser, x = phrase). Ebenfalls
Neutralisierung der gefundenen Bigramme.
bibtri.df <- merge(bib.df,bibneg.df,by = NULL)
bibtri.df <- merge(bibtri.df,bibcrease.df,by = NULL)
bibtrigramme.df                                     <-                      da-
ta.frame(paste(bibtri.df$begriff.text.y,bibtri.df$begriff.text,bibtri.df$begriff.t
ext.x,sep="
"),(bibtri.df$begriff.wert.x*bibtri.df$begriff.wert.y*(1/bibtri.df$begriff.wert))
-               (bibtri.df$begriff.wert.x*bibtri.df$begriff.wert.y)             -
(bibtri.df$begriff.wert.x*bibtri.df$begriff.wert))
colnames(bibtrigramme.df)<- c("BEGRIFF_TEXT","BEGRIFF_WERT")
colnames(bib.df)<- c("BEGRIFF_TEXT","BEGRIFF_WERT")

#Berechnung der Sentiment-Werte für 1-gram,2-gram and 3-gram
tweets_tatort_corpus_a      <-      tm_map(tweets_tatort_corpus_a,      function(x)
f_calc_sentiment (x,bib.df,1), lazy = TRUE)
tweets_tatort_corpus_b      <-      tm_map(tweets_tatort_corpus_b,      function(x)
f_calc_sentiment (x,bib.df,1), lazy = TRUE)

tweets_tatort_corpus_a      <-      tm_map(tweets_tatort_corpus_a,      function(x)
f_calc_sentiment (x,bibbigramme.df,2), lazy = TRUE)
tweets_tatort_corpus_b      <-      tm_map(tweets_tatort_corpus_b,      function(x)
f_calc_sentiment (x,bibbigramme.df,2), lazy = TRUE)

tweets_tatort_corpus_a      <-      tm_map(tweets_tatort_corpus_a,      function(x)
f_calc_sentiment (x,bibtrigramme.df,3), lazy = TRUE)
tweets_tatort_corpus_b      <-      tm_map(tweets_tatort_corpus_b,      function(x)
f_calc_sentiment (x,bibtrigramme.df,3), lazy = TRUE)

#Laden der Entitäten / Kontext-Informationen aus der DB
context.df <- f_get_context()
context.corpus                                  <-                              Cor-
pus(VectorSource(context.df$entitaet.text[1:length(context.df[,1])]))
my_stopwords <- f_get_begriffe("STOPWORD")[,1]
context.corpus <- tm_map(context.corpus, removeWords, my_stopwords)
```

```
context.corpus <- tm_map(context.corpus, function(x) str_trim(x))
context.corpus <- tm_map(context.corpus, stripWhitespace)
context.corpus <- tm_map(context.corpus, function(x) f_stemDocument (x,language =
"de"))
i<-0
context.corpus <- tm_map(context.corpus, function(x) {
  i <<- i + 1
  meta(x, "ENTITY_KEY") <- context.df[i,1]
  meta(x, "TOKEN_COUNT") <- length(WordTokenizer(x))
  x
},mc.cores=1)
context_clean.df <- data.frame(as.character(context.corpus),
                               as.character(meta(context.corpus,type   =   "local",
tag = "ENTITY_KEY")),
                               as.character(meta(context.corpus,type   =   "local",
tag = "TOKEN_COUNT")),stringsAsFactors = FALSE)
colnames(context_clean.df)<- c("text","entity_key","token_count")

#Vergleich für einzelne Wörter (Unigramme)
tweets_tatort_corpus_a      <-      tm_map(tweets_tatort_corpus_a,      function(x)
f_match_context  (x,context_clean.df[context_clean.df$token_count==1,],1),  lazy  =
TRUE)
tweets_tatort_corpus_b      <-      tm_map(tweets_tatort_corpus_b,      function(x)
f_match_context  (x,context_clean.df[context_clean.df$token_count==1,],1),  lazy  =
TRUE)
#Vergleich für zwei aufeinander folgende Wörter (Bigramme)
tweets_tatort_corpus_a      <-      tm_map(tweets_tatort_corpus_a,      function(x)
f_match_context  (x,context_clean.df[context_clean.df$token_count==2,],2),  lazy  =
TRUE)
tweets_tatort_corpus_b      <-      tm_map(tweets_tatort_corpus_b,      function(x)
f_match_context  (x,context_clean.df[context_clean.df$token_count==2,],2),  lazy  =
TRUE)
#Vergleich für drei aufeinander folgende Wörter (Trigramme)
tweets_tatort_corpus_a      <-      tm_map(tweets_tatort_corpus_a,      function(x)
f_match_context  (x,context_clean.df[context_clean.df$token_count==3,],3),  lazy  =
TRUE)
tweets_tatort_corpus_b      <-      tm_map(tweets_tatort_corpus_b,      function(x)
f_match_context  (x,context_clean.df[context_clean.df$token_count==3,],3),  lazy  =
TRUE)

#Erstellung eines DataFrame aus den Metadaten des Korpus
sentiment_a.df  <-  data.frame(as.character(meta(tweets_tatort_corpus_a,type  =  "lo-
cal", tag = "TWEET_ID")),
                               as.character(meta(tweets_tatort_corpus_a,type  =  "lo-
cal", tag = "ENTITY_KEYS")),
                               as.character(meta(tweets_tatort_corpus_a,type  =  "lo-
cal", tag = "SENTIMENT_SCORE")),stringsAsFactors = FALSE)
sentiment_b.df  <-  data.frame(as.character(meta(tweets_tatort_corpus_b,type  =  "lo-
cal", tag = "TWEET_ID")),
                               as.character(meta(tweets_tatort_corpus_b,type  =  "lo-
cal", tag = "ENTITY_KEYS")),
                               as.character(meta(tweets_tatort_corpus_b,type  =  "lo-
cal", tag = "SENTIMENT_SCORE")),stringsAsFactors = FALSE)

#Vergabe von Spatennamen
colnames(sentiment_a.df)<-c('TWEET_ID', 'ENTITAET_KEYS', 'SENTIMENT_WERT')
colnames(sentiment_b.df)<-c('TWEET_ID', 'ENTITAET_KEYS', 'SENTIMENT_WERT')

#Zusammensetzen
sentiment.df <- rbind(sentiment_a.df,sentiment_b.df)


#Skript saveResults: Schreiben der Ergebnisse in die DB

#(Neu-)Aufbau einer Verbindung zur Datenbank
ore.disconnect()
ore.connect(user = "tatort", sid = "orcl", host = "10.1.201.83", password = "ta-
tort",
            port = 1521, type = c("ORACLE"))
ore.attach()
#Schreiben der Ergebnisse mit Drop/Create
ore.drop(table = "R_T_TWEETS_SENTIMENTS")
ore.create(sentiment.df,table = "R_T_TWEETS_SENTIMENTS")
#Schließen der Verbindung zur DB
ore.disconnect()
```

Anhang E: Oracle Data Integrator

Schicht	Name	Quelle	Lademethode	Package
Stage	Daten werden durch R geschrieben	R Data Frame	Truncate/ Insert	PCK_REFRESH_ TWEETS
Stage	MAP_S2S_TWEETS_T WEETS	V_TWEETS_TATORT	Truncate/ Insert	PCK_REFRESH_ TWEETS
Stage	MAP_S2S_TWEETS_O RTE	V_TWEETS_ORTE	Truncate/ Insert	PCK_REFRESH_ TWEETS
Stage	MAP_S2S_TWEETS_RE TWEETS	V_TWEETS_TATORT _RETWEETS	Truncate/ Insert	PCK_REFRESH_ TWEETS
Stage	MAP_S2S_TWEETS_N UTZER	V_TWEETS_TATORT _NUTZER	Truncate/ Insert	PCK_REFRESH_ TWEETS
Stage	MAP_S2S_SENTIWS_N EG	SentiWS_v1.8c_Negative.txt	Truncate/ Insert	PCK_REFRESH_ WOERTERBUCH
Stage	MAP_S2S_SENTIWS_P OS	SentiWS_v1.8c_Positive.txt	Truncate/ Insert	PCK_REFRESH_ WOERTERBUCH
Stage	MAP_S2S_TATORT_EPI SODEN	Episoden.csv	Truncate/ Insert	PCK_REFRESH_ TATORT
Stage	MAP_S2S_TATORT_ER MITTLERTEAMS	Ermittlerteams.csv	Truncate/ Insert	PCK_REFRESH_ TATORT
Core	MAP_S2C_TWEETS_T WEETS	S_T_TWEETS_ TWEETS	Incremental Update	PCK_REFRESH_ TWEETS
		S_T_TWEETS_ RETWEETS		
Core	MAP_S2C_TWEETS_O RTE	C_V_TWEETS_ORTE	Incremental Update	PCK_REFRESH_ TWEETS
Core	MAP_S2C_LAENDER	C_V_LAENDER	Incremental Update	PCK_REFRESH_ STAMMDATEN
Core	MAP_S2C_BUNDESLAE NDER	C_V_BUNDESLANED ER	Incremental Update	PCK_REFRESH_ STAMMDATEN
		C_T_LAENDER		
Core	MAP_S2C_ORTE	S_V_ORTE	Incremental Update	PCK_REFRESH_ STAMMDATEN
		C_T_ BUNDESLAENDER		
Core	MAP_S2C_TWEETS_N UTZER	S_T_TWEETS_ NUTZER	Incremental Update	PCK_REFRESH_ TWEETS
Core	MAP_S2C_TWEETS_N UTZER_FOLLOW_HIST	C_T_TWEETS_ NUTZER	Incremental Update	PCK_REFRESH_ TWEETS
Core	MAP_S2C_BIB_BEGRIF FE	S_T_SENTIWS_NEG	Incremental Update	PCK_REFRESH_ WOERTERBUCH
		S_T_SENTIWS_POS		
		P_T_BIB_BEGRIFFE		
		P_T_BIB_BEGRIFF_ TYPEN		

Schicht	Name	Quelle	Lademethode	Package
Core	MAP_S2C_BIB_ SYNONYME	S_V_SENTIWS_SYN	Incremental Update	PCK_REFRESH_ WOERTERBUCH
		P_T_BIB_ SYNONYME		
		P_T_BIB_ BEGRIFF_ TYPEN		
		C_V_BIB_KONTEXT_ OHNE_SYN		
		C_T_BIB_BEGRIFFE		
Core	MAP_S2C_TATORT_ ERMITTLER	S_T_TATORT_ ERMITTLER_TEAMS	Incremental Update	PCK_REFRESH_ TATORT
		C_T_TATORT_ TEAMS		
Core	MAP_S2C_TATORT_ TEAMS	S_T_TATORT_ ERMITTLER_TEAMS	Incremental Update	PCK_REFRESH_ TATORT
Core	MAP_S2C_TATORT_ EPISODEN	S_T_TATORT_ EPISODEN	Incremental Update	PCK_REFRESH_ TATORT
		C_T_TATORT_ TEAMS		
		C_T_TATORT_ ERMITTLER		
		C_T_ORTE		
Core	MAP_S2C_TWEETS_ SENTIMENTS	S_V_TWEETS_ SENTIMENTS	Incremental Update	PCK_REFRESH_ TWEETS
		C_T_TWEETS_ TWEETS		
Mart	PR_C2M_D_DATUM	Prozedur mit Row-Generator	Merge	PCK_REFRESH_ STAMMDATEN
Mart	PR_C2M_D_UHRZEIT	Prozedur mit Row-Generator	Merge	PCK_REFRESH_ STAMMDATEN
Mart	MAP_C2M_D_ORTE	C_T_ORTE	Merge	PCK_REFRESH_ STAMMDATEN
		C_T_ BUNDESLAENDER		
		C_T_LAENDER		
Mart	MAP_C2M_D_TATORT _EPSIODEN	C_T_TATORT_ TEAMS	Merge	PCK_REFRESH_ TATORT
		C_T_TATORT_ ERMITTLER		
		C_T_TATORT_ EPISODEN		
Mart	MAP_C2M_D_ TWEETS_NUTZER	C_T_TWEETS_ NUTZER	Merge	PCK_REFRESH_ TWEETS
Mart	MAP_C2M_D_ ENTTITAETEN	C_V_SENTIMENT_ KONTEXT	Merge	PCK_REFRESH_ TWEETS
Mart	MAP_C2M_F_QUOTE N	C_T_TATORT_ EPISODEN	Incremental Update	PCK_REFRESH_ TATORT

Schicht	Name	Quelle	Lademe-thode	Package
		M_D_DATUM		
Mart	MAP_C2M_F_TWEETS	C_T_TWEETS	Merge	PCK_REFRESH_ TWEETS
		M_D_DATUM		
		M_D_UHRZEIT		
Mart	MAP_C2M_F_ SENTIMENTS	C_T_TWEETS_ SENTIMENTS	Incremental Update	PCK_REFRESH_ TWEETS
		C_T_TWEETS_ TWEETS		
		M_D_DATUM		
		M_D_UHRZEIT		

Tabelle 12: ODI-Mapping-Überblick

Anhang F: Oracle Business Intelligence Suite

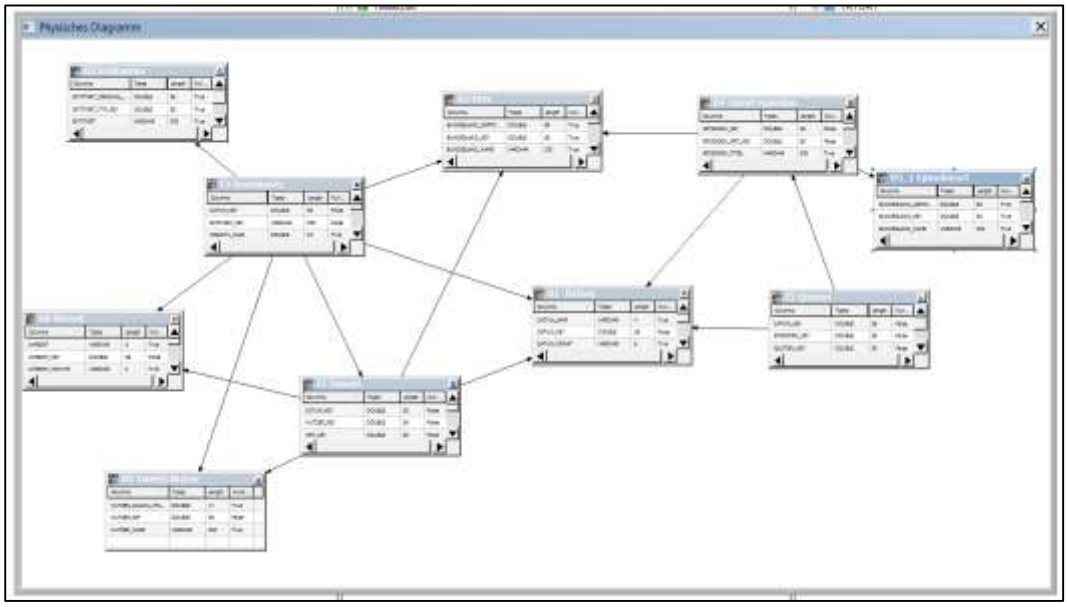

Abbildung 44: OBIEE Physisches Diagramm

Anhang G: Manuelle Sentiment-Validierung

Abbildung 45: Ausschnitt Sentiment-Validierung (negativ)

Abbildung 46: Ausschnitt Sentiment-Validierung (neutral)

Abbildung 47: Ausschnitt Sentiment-Validierung (positiv)

Anhang H: Tatort-Daten

Folge	Titel	Sender	Erstaus-strahlung	Ermittler	Fall	Autor	Regie	Besonderheiten	Stadt	PLZ	Quot (mio)	Quote (%)
918	Wahre Liebe	WDR	28.09.2014	Ballauf und Schenk	61	Maxim Leo	André Erkau	Letzte Folge mit Christian Tasche, der während der Dreharbeiten verstorben ist.	Köln	50667	10,6	30,7
919	Winternebel	SWR	05.10.2014	Blum und Perlmann	27	Jochen Greve	Patrick Winczewski		Konstanz	78462	9,42	26,4
920	Im Schmerz geboren	HR	12.10.2014	Murot	4	Michael Proehl	Florian Schwarz	Premiere und Bernd Burgemeister Fernsehpreis beim Filmfest München 2014, Medienkulturpreis beim Festival des deutschen Films 2014, neuer Leichenrekord	Wiesbaden	65183	9,29	26
921	Blackout	SWR	26.10.2014	Odenthal und Kopper	60	Eva und Volker A. Zahn	Patrick Winczewski	Open-Air-Vorpremiere mit den Hauptdarstellern beim SWR Sommerfestival am 30. Mai 2014 in Mainz	Ludwigshafen am Rhein	67059	10,4	29
922	Vielleicht	RBB	16.11.2014	Stark	37	Klaus Krämer	Klaus Krämer	Letzter Fall für Hauptkommissar Stark, Ausstrahlung im Rahmen der ARD-Themenwoche	Berlin	10115	9,86	27,1
923	Eine Frage des Gewissens	SWR	23.11.2014	Lannert und Bootz	15	Sönke Lars Neuwöhner, Sven Poser	Till Endemann		Stuttgart	70173	10,41	29,9
924	Die Feigheit des Löwen	NDR	30.11.2014	Falke und Lorenz	4	Friedrich Ani	Marvin Kren		Oldenburg	26121	9,18	26,1
925	Der sanfte Tod	NDR	07.12.2014	Lindholm	22	Alexander Adolph	Alexander Adolph		Hannover	30159	10,19	27,8
926	Der Maulwurf	MDR	21.12.2014	Funck, Schaffert und Grewel	2	Michael B. Müller, Leo P. Ard	Johannes Grieser	letzter Fall des gesamten Teams	Erfurt	99084	8,47	18,5
927	Weihnachtsgeld	SR	26.12.2014	Stellbrink und Marx	4	Michael Illner	Zoltan Spirandelli		Saarbrücken	66111	6,48	19,9
928	Das verkaufte Lächeln	BR	28.12.2014	Batic und Leitmayr	69	Holger Joos	Andreas Senn		München	80331	9,71	27,4

Folge	Titel	Sender	Erstaus-strahlung	Ermittler	Fall	Autor	Regie	Besonderheiten	Stadt	PLZ	Quot (mio)	Quote (%)
929	Der Irre Iwan	MDR	01.01.2015	Lessing und Dorn	2	Murmel Clausen, Andreas Pflüger	Richard Huber		Weimar	99423	8,87	23,9
930	Deckname Kidon	ORF	04.01.2015	Eisner und Fellner	34	Max Gruber	Thomas Roth		Wien	1004	8,44	22,8
931	Hydra	WDR	11.01.2015	Faber, Bönisch, Dalay und Kossik	5	Jürgen Werner	Nicole Weegmann		Dortmund	44135	9,11	25
932	Die Sonne stirbt wie ein Tier	SWR	18.01.2015	Odenthal und Kopper	61	Harald Göckeritz	Patrick Winczewski		Ludwigshafen am Rhein	67059	9,45	25,7
933	Borowski und der Himmel über Kiel	NDR	25.01.2015	Borowski und Brandt	24	Rolf Base-dow	Christian Schwochow	Uraufgeführt am 27. September 2014 beim Filmfest Hamburg	Kiel	24103	10,67	28,5
934	Freddy tanzt	WDR	01.02.2015	Ballauf und Schenk	62	Jürgen Werner	Andreas Kleinert		Köln	50667	10,49	28,1
935	Chateau Mort	SWR	08.02.2015	Blum und Perlmann	28	Stefan Dähnert	Mark Rensing		Konstanz	78462	9,38	25,6
936	Blutschuld	MDR	15.02.2015	Saalfeld und Keppler	20	Stefan Kornatz	Stefan Kornatz		Leipzig	4103	9,4	25,8
937	Das Haus am Ende der Straße	HR	22.02.2015	Steier	7	Michael Proehl, Erol Yesilkaya	Sebastian Marka	Letzter Fall für KHK Steier	Frankfurt am Main	60311	9,37	22,9
938	Grenzfall	ORF	08.03.2015	Eisner und Fellner	35	Rupert Henning	Rupert Henning		Wien	1004	9,59	26,9
939	Die Wiederkehr	RB	15.03.2015	Lürsen und Stedefreund	31	Matthias Tuchmann und Stefanie Veith	Florian Baxmeyer		Bremen	28195	10,61	28,9
940	Das Muli	RBB	22.03.2015	Rubin und Karow	1	Stefan Kolditz	Stephan Wagner	Uraufgeführt am 9. März 2015 im Kino Babylon mit Vorstellung des neuen Teams	Berlin	10115	10,19	27,1
941	Borowski und die Kinder von Gaarden	NDR	29.03.2015	Borowski und Brandt	25	Eva Zahn und Volker A. Zahn	Florian Gärtner		Kiel	24103	9,43	25,4

Folge	Titel	Sender	Erstaus-strahlung	Ermittler	Fall	Autor	Regie	Besonderheiten	Stadt	PLZ	Quot (mio)	Quote (%)
942	Frohe Ostern, Falke	NDR	06.04.2015	Falke und Lorenz	5	Thomas Stiller	Thomas Stiller	Uraufgeführt am 13. September 2014 beim Internationalen Filmfest Oldenburg. „Event"-Tatort zu Ostern 2015, Falke und Lorenz ermitteln in Hamburg	Hamburg	20095	8,49	23,8
943	Der Himmel ist ein Platz auf Erden	BR	12.04.2015	Voss und Ringelhahn	1	Max Färberböck, Catharina Schuchmann	Max Färberböck	Erster Fall des neuen Teams für Franken	Nürnberg	90402	12,11	33,7
944	Dicker als Wasser	WDR	19.04.2015	Ballauf und Schenk	63	Norbert Ehry	Kaspar Heidelbach		Köln	50667	10,73	30,6
945	Niedere Instinkte	MDR	26.04.2015	Saalfeld und Keppler	21	Sascha Arango	Claudia Garde	Letzter Fall für Saalfeld und Keppler.	Leipzig	4103	10,06	28,3
946	Schwerelos	WDR	03.05.2015	Faber, Bönisch, Dalay und Kossik	6	Benjamin Zakrisson Braeunlich	Züli Alada		Dortmund	44135	9,43	25,9
947	Kälter als der Tod	HR	17.05.2015	Brix und Janneke	1	Michael Proehl, Matthias Tuchmann	Florian Schwarz	Erster Fall des neuen Frankfurter Teams.	Frankfurt am Main	60311	9,89	28,4

Tabelle 13: Tatort Episodendaten

Anhang I: Datenträger

E R K L Ä R U N G

Ich versichere an Eides Statt, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, 23.06.2015 Martin Frisch