

HANSER



Leseprobe

Jonas Freiknecht

Big Data in der Praxis

Lösungen mit Hadoop, HBase und Hive. Daten speichern, aufbereiten,
visualisieren

ISBN (Buch): 978-3-446-43959-7

ISBN (E-Book): 978-3-446-44177-4

Weitere Informationen oder Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-43959-7>

sowie im Buchhandel.

Inhalt

Vorwort	XI
1 Einleitung	1
2 Big-Data	7
2.1 Historische Entstehung	8
2.2 Big-Data – ein passender Begriff?	9
2.2.1 Die drei V	10
2.2.2 Das vierte V – Veracity	13
2.2.3 Der Verarbeitungsaufwand ist big	14
2.2.4 Sicht der Industrien auf Big-Data	14
2.3 Eingliederung in BI und Data-Mining	15
3 Hadoop	19
3.1 Hadoop kurz vorgestellt	20
3.2 HDFS – das Hadoop Distributed File System	21
3.3 Hadoop 2.x und YARN	25
3.4 Hadoop als Single-Node-Cluster aufsetzen	26
3.4.1 Falls etwas nicht funktioniert	39
3.5 Map-Reduce	42
3.6 Aufsetzen einer Entwicklungsumgebung	44
3.7 Implementierung eines Map-Reduce-Jobs	51
3.8 Ausführen eines Jobs über Kommandozeile	63
3.9 Verarbeitung im Cluster	67
3.10 Aufsetzen eines Hadoop-Clusters	69
3.11 Starten eines Jobs via Hadoop-API	81
3.12 Verketten von Map-Reduce-Jobs	94
3.13 Verarbeitung anderer Dateitypen	109
3.14 YARN-Anwendungen	125
3.14.1 Logging und Log-Aggregation in YARN	125
3.14.2 Eine einfache YARN-Anwendung	129

3.15 Vor- und Nachteile der verteilten Verarbeitung	153
3.16 Die Hadoop Java-API	154
3.16.1 Ein einfacher HDFS-Explorer	155
3.16.2 Cluster-Monitor	167
3.16.3 Überwachen der Anwendungen im Cluster	169
3.17 Gegenüberstellung zur traditionellen Verarbeitung	171
3.18 Big-Data aufbereiten	172
3.18.1 Optimieren der Algorithmen zur Datenauswertung	172
3.18.2 Ausdünnung und Gruppierung	174
3.19 Ausblick auf Apache Spark	176
3.20 Markt der Big-Data-Lösungen	178
4 Das Hadoop-Ecosystem	181
4.1 Ambari	182
4.2 Sqoop	183
4.3 Flume	183
4.4 HBase	184
4.5 Hive	184
4.6 Pig	185
4.7 Zookeeper	185
4.8 Mahout	186
4.9 Spark	187
4.10 Data Analytics und das Reporting	187
5 NoSQL und HBase	189
5.1 Historische Entstehung	189
5.2 Das CAP-Theorem	190
5.3 Typen von Datenbanken	191
5.4 Umstieg von SQL und Dateisystemen auf NoSQL oder HDFS	194
5.4.1 Methoden der Datenmigration	194
5.5 HBase	196
5.5.1 Das Datenmodell von HBase	196
5.5.2 Aufbau von HBase	198
5.5.3 Installation als Stand-alone	199
5.5.4 Arbeiten mit der HBase Shell	201
5.5.5 Verteilte Installation auf dem HDFS	203
5.5.6 Laden von Daten	206
5.5.6.1 HBase Bulk Loading über die Shell	207
5.5.6.2 Datenextrakt aus einer Datenbank über Sqoop	209
5.5.7 HBase Java-API	218
5.5.8 Der Umstieg von einem RDBMS auf HBase	242

6 Data-Warehousing mit Hive	245
6.1 Installation von Hive	246
6.2 Architektur von Hive	248
6.3 Das Command Line Interface (CLI)	249
6.4 HiveQL als Abfragesprache	251
6.4.1 Anlegen von Datenbanken	251
6.4.2 Primitive Datentypen	252
6.4.3 Komplexe Datentypen	252
6.4.4 Anlegen von Tabellen	253
6.4.5 Partitionierung von Tabellen	254
6.4.6 Externe und interne Tabellen	254
6.4.7 Löschen und leeren von Tabellen	255
6.4.8 Importieren von Daten	256
6.4.9 Zählen von Zeilen via count	257
6.4.10 Das SELECT-Statement	257
6.4.11 Beschränken von SELECT über DISTINCT	260
6.4.12 SELECT auf partitionierte Tabellen	261
6.4.13 SELECT sortieren mit SORT BY und ORDER BY	261
6.4.14 Partitionieren von Daten durch Bucketing	263
6.4.15 Gruppieren von Daten mittels GROUP BY	264
6.4.16 Subqueries – verschachtelte Abfragen	265
6.4.17 Ergebnismengen vereinigen mit UNION ALL	265
6.4.18 Mathematische Funktionen	266
6.4.19 String-Funktionen	267
6.4.20 Aggregatfunktionen	268
6.4.21 User-Defined Functions	269
6.4.22 HAVING	277
6.4.23 Datenstruktur im HDFS	277
6.4.24 Verändern von Tabellen	278
6.4.25 Erstellen von Views	281
6.4.26 Löschen einer View	281
6.4.27 Verändern einer View	281
6.4.28 Tabellen zusammenführen mit JOINS	282
6.5 Hive Security	284
6.5.1 Implementieren eines Authentication-Providers	290
6.5.2 Authentication-Provider für HiveServer2	294
6.5.3 Verwenden von PAM zur Benutzerauthentifizierung	295
6.6 Hive und JDBC	296
6.7 Datenimport mit Sqoop	314
6.8 Datenexport mit Sqoop	316
6.9 Hive und Impala	317
6.10 Unterschied zu Pig	318
6.11 Zusammenfassung	319

7 Big-Data-Visualisierung	321
7.1 Theorie der Datenvisualisierung	321
7.2 Diagrammauswahl gemäß Datenstruktur	327
7.3 Visualisieren von Big-Data erfordert ein Umdenken	328
7.3.1 Aufmerksamkeit lenken	329
7.3.2 Kontextsensitive Diagramme	331
7.3.3 3D-Diagramme	333
7.3.4 Ansätze, um Big-Data zu visualisieren	334
7.4 Neue Diagrammarten	336
7.5 Werkzeuge zur Datenvisualisierung	340
7.6 Entwicklung einer einfachen Visualisierungskomponente	344
8 Auf dem Weg zu neuem Wissen – aufbereiten, anreichern und empfehlen	357
8.1 Eine Big-Data-Table als zentrale Datenstruktur	360
8.2 Anreichern von Daten	362
8.2.1 Anlegen einer Wissensdatenbank	364
8.2.2 Passende Zuordnung von Daten	364
8.3 Diagrammempfehlungen über Datentypanalyse	368
8.3.1 Diagrammempfehlungen in der BDTable	370
8.4 Textanalyse – Verarbeitung unstrukturierter Daten	376
8.4.1 Erkennung von Sprachen	377
8.4.2 Natural Language Processing	378
8.4.2.1 Klassifizierung	379
8.4.2.2 Sentiment-Analysis	384
8.4.3 Mustererkennung mit Apache UIMA	386
9 Zusammenfassung und Ausblick	405
10 Häufige Fehler	409
11 Anhang	415
11.1 Installation und Verwendung von Sqoop2	415
11.2 Hadoop für Windows 7 kompilieren	421
Literaturverzeichnis	425
Index	429

Vorwort

Die Verfügbarkeit von Daten hat sich in den vergangenen zehn Jahren drastisch verändert. Immer neue Datenquellen, die zunehmende Verbreitung mobiler, internetfähiger Geräte und natürlich alle Entwicklungen, die sich aus dem *Web 2.0* ergeben haben, tragen dazu bei, dass sich Unternehmen heute mit einer sehr viel größeren Datenmenge konfrontiert sehen, die es zu erfassen, zu speichern und auszuwerten gilt, als bisher. Dabei ist es nicht nur die Datenmenge selbst, die den Unternehmen Probleme bereitet, sondern darüber hinaus auch die Struktur und die Art der Daten sowie die Geschwindigkeit, mit der sie anfallen.

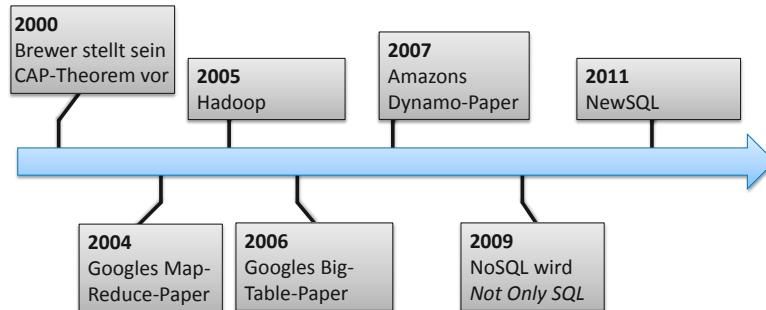
Die meisten Unternehmen sehen sich noch immer in einer klassisch relationalen Welt verhaftet, die sie sich selbst in den letzten Jahrzehnten aufgebaut haben. Über mehr als 40 Jahre hinweg haben relationale Datenbanken und SQL als standardisierte Abfragesprache sowie die dazugehörigen Architekturen die Datenhaltung, -verarbeitung und -auswertung beherrscht. Und auch heute stellt niemand infrage, dass der Großteil der Datenbanken, die wir in den Unternehmen vorfinden, diesem klassischen Paradigma entsprechen: Zeilenorientierte, weitestgehend normalisierte RDBMS-Systeme, die die Daten in Tabellen halten und über Beziehungen miteinander verknüpfen, um auf diese Weise ihre Datenbankschemata aufzubauen.

Viele der Problemstellungen, die sich in den vergangenen zehn Jahren entwickelt haben, passen jedoch nicht in ein solch relationales Modell – was die Anwender zum Teil aber nicht davon abhält zu versuchen, ihre Welt mit eher mäßigem Erfolg in das relationale Modell zu zwingen.

Seit einigen Jahren aber gibt es Alternativen zu den klassischen Architekturen. Als Stichwörter fallen dabei immer wieder Termini wie Hadoop, NoSQL oder auch Map-Reduce. Letztlich aber ist es der Begriff Big-Data, der diese Stichwörter zusammenführt. Ein Begriff, der leider in den vergangenen Monaten zu oft für alles herhalten musste, was mit Daten zu tun hatte. Die hier zu beobachtenden Entwicklungen sind dabei vielfach eng miteinander verwoben, auch wenn sie nicht explizit voneinander abhängen. Getrieben werden sie in allen Bereichen vor allem von großen Internetkonzernen wie Google, Facebook, Yahoo oder Amazon, die maßgeblich an den grundlegenden Konzepten mitgearbeitet haben.

Über den Begriff Big-Data und seine Definition ist in der Fachwelt viel und sehr kontrovers diskutiert worden. Unabhängig davon, ob man von den klassischen drei V (*Volume*, *Variety* und *Velocity*) ausgeht oder noch weitere (wie etwa *Value* oder *Veracity*) einbezieht, lässt sich feststellen, dass es sich bei Big-Data um zumeist große, polystrukturierte Datenmengen

handelt, deren Verarbeitung mit konventionellen Mitteln, wie etwa den traditionellen relationalen Datenbanken, kaum oder gar nicht mehr möglich ist.



Milestones in the Big-Data-Development

Die *Hadoopisierung* in den Unternehmen schreitet darüber hinaus zunehmend voran. Dabei geht es letztlich um die Verarbeitung von Big-Data mit Hadoop oder ähnlichen Frameworks (z. B. *Disco* oder *BashReduce*) in einem verteilten System. Die hier zum Einsatz kommenden Konzepte umfassen etwa das sogenannte Map-Reduce-Verfahren, welches die Verarbeitung großer Datenmengen in einem Cluster ermöglicht, und für die Datenhaltung werden in der Regel NoSQL-Datenbanken eingesetzt. Der Grund dafür, dass im Geflecht von Big-Data und Hadoop NoSQL-Datenbanken eingesetzt werden, ist unter anderem in der Skalierbarkeit der Systeme zu sehen.

Das vorliegende Buch zeigt nun, wie man eine entsprechende Architektur tatsächlich aufbauen kann. Und dies nicht nur theoretisch, sondern sehr praxisnah, stets mit konkreten Beispielen und bei Bedarf auch mit passenden Code-Schnipseln. Die entsprechenden theoretischen Grundlagen und Konzepte werden vorgestellt und in einem nächsten Schritt gleich zur Anwendung gebracht. Angefangen bei einem entsprechenden Hadoop-Cluster mit den typischen in der Praxis häufig zu findenden Tools über die verwendete NoSQL-Datenbank bis hin zur Visualisierung der Ergebnisse. Dabei werden auch die unterschiedlichen Aspekte, die Big-Data ausmachen, behandelt. So wird nicht nur der Größenaspekt allein adressiert, sondern darüber hinaus auch, wie man etwa das Problem der Unstrukturiertheit der Daten angehen kann. Hier werden Methoden aus dem Data- und Text-Mining ebenso vorgestellt wie neue Aspekte, beispielsweise das Natural Language Processing.

Nach der Lektüre bleibt für den Leser nur noch die Frage offen, ob er tatsächlich *Big-Data-Schmerzen* hat, wie Pavlo Baron (2012) es nennt, und die vorgestellten Konzepte einsetzen möchte. Wie diese konkret umzusetzen sind, zeigt Jonas Freiknechts Buch anschaulich und erfreulich praxisorientiert.

Professor Dr. Uwe Haneke, Hochschule Karlsruhe

2

Big-Data

Zu Beginn wurde bereits eine kurze Einführung in das Thema Big-Data gegeben. In diesem Kapitel möchte ich diese noch etwas vertiefen und verschiedene Definitionen vorstellen, die diesbezüglich in den letzten Jahren entstanden sind. Bitte lassen Sie mich zuvor noch einmal den Gedanken der Präsenz von Big-Data in den verschiedenen Industrien aufgreifen, die eben in der Einleitung bereits in den Raum gestellt wurde. Bild 2.1 zeigt eine Statistik, die eine Umfrage von Gartner (Kart, 2012) aus dem Jahre 2012 verbildlicht. In dieser wurde evaluiert, welche Industrien sich mit dem Thema Big-Data in welchem Ausmaß auseinandersetzen.

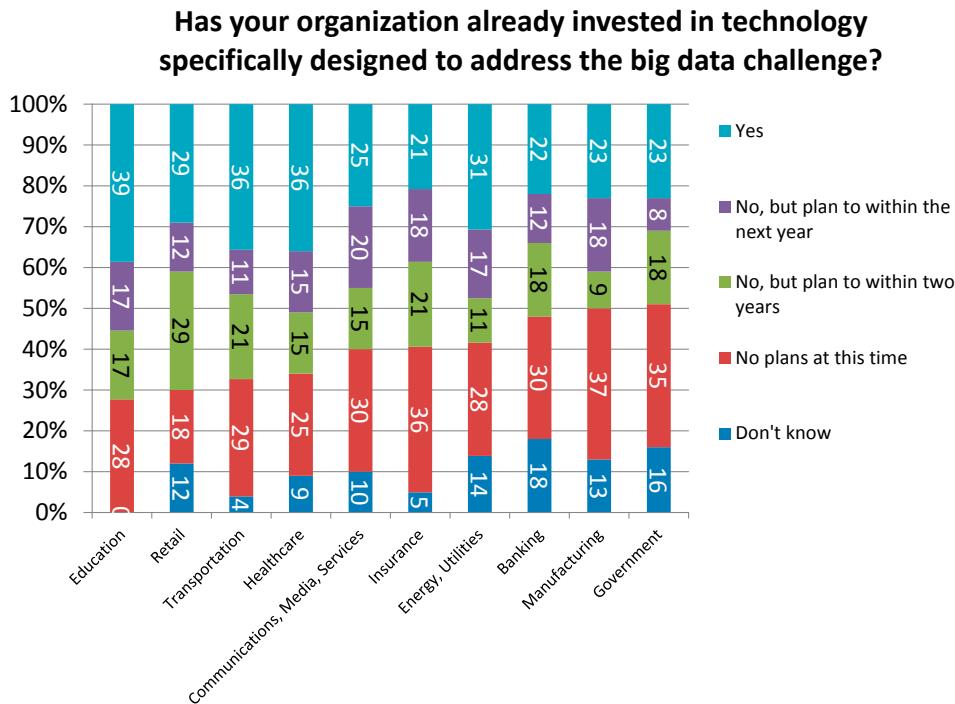


Bild 2.1 Big-Data regt das Interesse sämtlicher Industrien an.

Das wenig verblüffende Ergebnis ist, dass sich alle bereits damit beschäftigt haben, allen voran die Sparte *Education*. Zwar kann man davon ausgehen, dass es sich häufig noch um einfache Evaluationen handelt, die bestimmen sollen, ob der Einsatz passender Technologien im Unternehmen infrage kommt und notwendig ist bzw. sein wird, jedoch ist eine gewisse Grundeinstellung zu erkennen, die aussagt, dass Big-Data als wichtige Neuerung wahr- und ernst genommen wird. Erfolgte noch keine intensivere Auseinandersetzung, so ist sie jedoch für die Hälfte aller Unternehmen und Einrichtungen in den nächsten zwei Jahren geplant. Natürlich bleibt bei vielen eine Restunsicherheit, die dadurch hervorgerufen wird, dass man nicht weiß, ob Big-Data nun ein kurzlebiger Hype bleibt oder nicht. Dem sei jedoch entgegenzusetzen, dass uns die Diskussion rund um große Datenmengen nun schon einige Jahre beschäftigt und immer noch zu der Veröffentlichung neuer Artikel, Bücher und neuer Software führt. Wir können also bald aufhören, von einem bloßen Trend zu sprechen, und beginnen, Big-Data als integralen Bestandteil der heutigen IT zu sehen.

Bevor es in den folgenden Abschnitten an die Begriffsdefinitionen von Big-Data geht, soll zuvor ein kurzer Einblick in dessen historische Entstehung gegeben und skizziert werden, wo die Big-Data-Bewegung ihren Ursprung hat.

■ 2.1 Historische Entstehung

Einen konkreten Ursprung für den Begriff Big-Data gibt es nicht. Zwar ist bekannt (Press, 2012), dass der Begriff selbst von Michael Cox und David Ellsworth verhältnismäßig früh (im Jahr 1997) öffentlich in einem Paper genannt wurde (Cox et al., 1997), jedoch wird auf *Wikipedia* der Ursprung des Begriffs kontrovers diskutiert und andere Quellen werden genannt (Mashey), die behaupten, dass John Mashey den Terminus *Big* bereits 1994 im Zusammenhang mit Datenmengen verwendete. Dies geschah jedoch vorwiegend als stilistisches Mittel, um die Komplexität einiger Teilbereiche der IT hervorzuheben. Darunter waren auch *Big Bandwidth*, *Big Physics*, *Big Latency* und eben auch *Big Data*, sodass der Begriff zwar gleich dem heutigen verwendet wurde, die Bedeutung jedoch nicht oder nur in Teilen übereinstimmte.

Im November 2009 entstand der erste Wikipedia-Artikel zu Big-Data und wurde vom Benutzer John Blackburn prompt wieder gelöscht. Die Begründung war folgende:

"Delete as per nom - it is simply a combination of big and data, dictionary words which have no place here. I'm not even sure it's a neologism, and even if it was it doesn't need an article."
(John Blackburne)

Nach einigen Diskussionen, die besagten, dass *Big Band* oder *Big Bang* ebenso gelöscht werden müssten, wenn Big-Data doch auch nur ein zusammengesetzter Begriff aus einem Adjektiv und einem Substantiv sei, wurde der Artikel letztendlich zugelassen und sogar später (2011) in den *Gartner Hype Cycle* aufgenommen (McBurney, 2012).



Bild 2.2 Frühe Nennung des Begriffs Big-Data auf der IEEE Supercomputing Conference 1996 in Pittsburgh; Foto: Michael Woodacre

Besonders interessant an dieser Betrachtung ist, dass Doug Cutting bereits 2006 unter der Schirmherrschaft von Yahoo an Hadoop arbeitete (Cutting). Die Definition des Begriffs Big-Data hat also von der ersten Open-Source-Implementierung von Hadoop in 2006 bis hin zum Wikipedia-Artikel fünf Jahre benötigt, um zu reifen und um Big-Data als eigenes Teilgebiet der Informatik anzuerkennen.

In den letzten zwei Jahren wurde der Begriff von den bekannten IT-Häusern wie IBM, SAP oder Oracle aufgenommen und wird heutzutage in Zusammenhang mit Speicher, Business-Intelligence (BI), Data-Warehousing (von Data-Warehouse, DWH) und Data Analytics verwendet. *Visible Technologies* diagnostizierte einen Anstieg der Zahl der Begriffsnennungen in Social Media Channels von 2009 bis 2012 um 1211% (Press, 2012). Es ist also kaum mehr möglich, vor dem Begriff die Augen zu verschließen. Wie aber wird nun Big-Data eigentlich definiert?

■ 2.2 Big-Data – ein passender Begriff?

In diesem Abschnitt sollen zunächst verschiedene Quellen herangezogen werden, die Big-Data aus verschiedenen Perspektiven betrachten. Ohne in die Tiefe zu gehen, lässt sich Big-Data so definieren: Big-Data sind Datenmengen, die zu groß für traditionelle Datenbanksysteme sind, eine hohe Halbwertszeit besitzen und in ihrer Form nicht den Richtlinien herkömmlicher Datenbanksysteme entsprechen (Dumbill, 2012). Ziel ist es nun, diese Daten dennoch zu speichern und zu verarbeiten, sodass aus ihnen zeitnah wertvolle, neue Informationen gewonnen werden können. Diese neu gewonnenen Informationen können etwa passende Produktempfehlungen in E-Commerce-Lösungen sein, empfohlene Kontakte

in sozialen Netzwerken oder Artikelvorschläge auf Nachrichtenseiten. Sind diese Daten nun wirklich *big* gemäß der oben gegebenen Definition?

Auf den ersten Blick wirkt es tatsächlich nicht so, denn Freundschaften können als klassische n:m-Relation gespeichert werden, Artikelvorschläge werden anhand der Tags des gelesenen Artikels erstellt und Produktempfehlungen entstehen auf Basis der Produktkategorie der vorher betrachteten Artikel. Zieht man aber nun vom Benutzer generierte Inhalte (*User-Generated Content*) hinzu, trifft man auf Inhalte wie:

- Rezensionen und Bewertungsschreiben für Güter und Dienstleistungen
- Foren-Posts und Kommentare
- Pinnwandeinträge
- Blogseinträge
- (Wissenschaftliche) Artikel
- Tweets

Diese enthalten oft subjektive Meinungen von Benutzern und Konsumenten und sind dementsprechend wertvoll. Allerdings müssen sie vorher analysiert werden, um sie für Maschinen lesbar zu machen und somit dem Datenhalter einen Mehrwert zu liefern. Diese Datenbeschaffenheit geht mit einem Datenumfang einher, der die traditionelle Datenverarbeitung vor eine scheinbar unlösbare Aufgabe stellt. So sammeln Twitter beispielsweise 8 Terabyte an Daten am Tag, Facebook 500 Terabyte und Google verarbeitet pro Tag etwa 20 Petabyte an User-Generated Content.

2.2.1 Die drei V

Die Frage nach der am meisten verbreiteten Definition von Big-Data lässt sich wohl am ehesten durch die drei V beantworten, die der Anbieter von Marktanalysen Gartner 2001 einführte (Lancy, 2001). Zwar wurde hier noch kein Zusammenhang zu Big-Data hergestellt, jedoch erkannte man bereits die zukünftigen Herausforderungen der Datenverarbeitung (hier im E-Commerce-Sektor), die den Big-Data-Begriff später prägen sollten. Es werden nun die einzelnen V genauer beschrieben.

Volume

Hinter *Volume* (deutsch: Volumen) verbirgt sich der Begriff, den man auf den ersten Blick am wahrscheinlichsten erwartet. Wo früher 500 Megabyte große Festplatten eine Sensation waren, sind es heute die Gigabyte großen USB-Sticks in Fingernagelgröße. Dieses Beispiel mag zwar etwas einfach erscheinen, spiegelt aber dennoch die rasante Entwicklung der Speicherhardware wider. Mit den Möglichkeiten steigen nämlich auch die Anforderungen. Wo vor 15 Jahren 20 Gigabyte für einen PC völlig ausreichend waren, ist heute ein bis zwei Terrabyte Speicher der Stand der Dinge. Dieses Wachstum fasst eine Studie von IDC (Gantz et al., 2011) passend in Worte:

“Like our physical universe, the digital universe is something to behold – 1.8 trillion gigabytes in 500 quadrillion ‘files’ – and more than doubling every two years.”

Projiziert man diese Behauptung nun auf die Anforderungen an Suchmaschinen, Reporting, Online-Marketing und irgendwann auch Suchfunktionen auf dem lokalen Computer, so wird schnell klar, dass die Art, die Daten zu speichern und zu verarbeiten, überdacht werden muss. Wann man nun Volumina als groß bezeichnet, ist immer stark von der verfügbaren Hardware abhängig. Öffnet man eine 100 Megabyte große CSV-Datei (*Comma-Separated Value*) in Microsofts Excel, so benötigt der Ladevorgang bereits einige Minuten auf einem durchschnittlich ausgestatteten Desktop-PC. Auf einem mobilen Gerät oder sehr simpler Hardware (*RaspberryPI, FritzBox, Arduino ...*) sind solche Vorgänge überhaupt nicht praktikabel. Klar ist, dass hier die Leistung der CPU (Central Processing Unit) und des RAM (Random Access Memory) den Flaschenhals darstellt. Der Speicher an sich reicht in den meisten Fällen für 100 Megabyte ohne Weiteres aus. Es ist also ersichtlich, dass bei großen Datenmengen die Verarbeitung das *Big* ausmacht (Baron, 2013), nicht der Speicherbedarf (siehe Abschnitt 2.2.3).

Velocity

Das zweite V beschreibt die Geschwindigkeit der Daten. Damit wird einmal deren Aktualität angesprochen sowie die Geschwindigkeit der Verarbeitung. Ein passendes Beispiel sind etwa Twitter-Meldungen oder Blog-Posts, die zu Zeiten von Wahlkämpfen verfasst werden. Dabei stellt das Internet ein starkes, aber schwer zu kontrollierendes Werkzeug dar. Bekommt ein Politiker beispielsweise eine schlechte Presse, so muss das bereitstehende Marketing-Team entsprechend zügig reagieren und diese durch korrigierende Meldungen relativieren. Die Betonung liegt auf *zügig*, denn ein einziges Gerücht zur falschen Zeit kann zu ungewollten Auswirkungen führen. So sorgte ein am 23. April 2013 geschriebener Tweet über einen Bombenanschlag auf das Weiße Haus für einen Börsen-Crash an der Wall Street. Bleibt man dem Beispiel der Börse treu, finden sich schnell noch weitere Szenarien, in denen die Geschwindigkeit der Erfassung von Daten eine große Rolle spielt. So sind zum Beispiel Ankündigungen über Firmenübernahmen Gold wert, wenn diese direkt nach der Bekanntgabe verifiziert werden können. Für beide Szenarien gilt: Je schneller die Auswertung stattfindet, desto höher ist der Wert der Information.

Variety

Hier kommt nun die bereits häufig angesprochene Abwesenheit von festen Strukturen und Normalisierungen zur Sprache. Das sicherlich beste Beispiel, um die Datenvielfalt, mit der wir es zu tun haben, zu beschreiben, ist das Internet, das – außer vielleicht den Wiki-Seiten – keine feste Struktur vorweisen kann, aber doch einige Ähnlichkeiten aufweist, die eine maschinelle Verarbeitung ermöglichen. So können etwa bei der Analyse des HTML-Codes die Titel-Tags, z.B. `<h1>`, durchsucht werden, um eine thematische Einordnung des Inhalts vorzunehmen. Jedoch sind HTML-Seiten nicht die einzigen Daten, die verarbeitet werden. IBM befragte Mitte 2012 (Schroeck et al., 2012) einige Unternehmen, die an einer Big-Data-Initiative teilnahmen, welche Quellen sie für ihre Analysen verwenden.

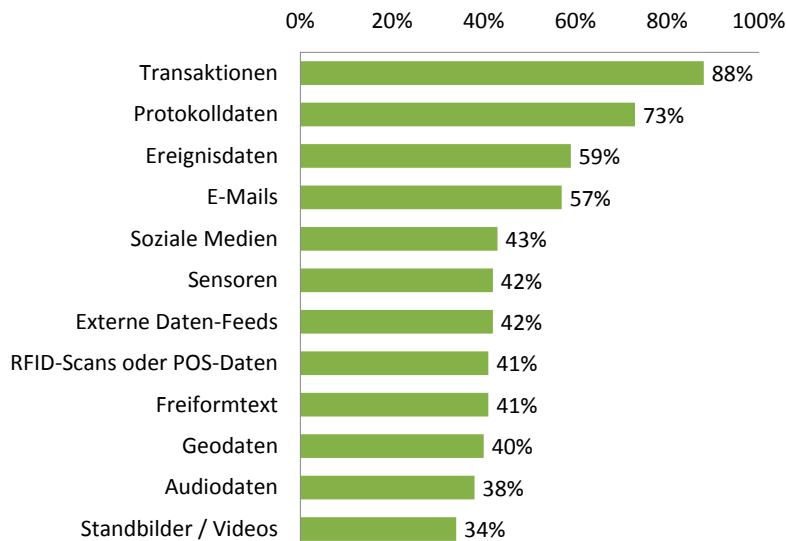


Bild 2.3 Quellen für Big-Data-Analysen auf Basis einer Umfrage von IBM im Jahr 2012

Transaktionsdaten beziehen sich dabei beispielsweise auf den klassischen Börsenhandel, in dem jeder Ver- und Einkauf gespeichert wird. Protokolldaten sind unter anderem Serverlogs, die entsprechend dem Log-Level der jeweiligen Architektur sehr groß ausfallen können und häufig dazu genutzt werden, um Klickpfade durch komplexe Anwendungen zu ermitteln und Benutzer möglichst lange im System zu halten bzw. zu einer bestimmten Aktion (Kauf, Registrierung, Empfehlung) zu bewegen. Ereignisdaten werden etwa in der Automobilindustrie protokolliert, in der Fahrzeugteile produziert, an die Logistik übergeben und verfrachtet werden. Das Feld dieser Daten ist jedoch weder auf die Automobilindustrie noch auf die Produktherstellung im Allgemeinen beschränkt, sondern beschäftigt sich in der IT mit allen Systemen, die bestimmte Ereignisse aufzeichnen und zur Auswertung bereitstellen. So kann etwa die Ausschussquote einer Produktreihe überprüft und ggf. fehlerhafte Teile im gesamten Produktionszyklus, über mehrere Hersteller hinweg, ausfindig gemacht werden. Platz vier auf der Liste belegen E-Mails, die von Mail-Service-Anbietern gescannt, auf Muster von Malware oder Spam durchsucht und für gezielte Produktvorschläge für den jeweiligen Empfänger hin untersucht werden. Ob und wie dieses Vorgehen mit den Datenschutzbestimmungen des jeweiligen Landes und der Moral der Betreiber vereinbar ist, sei einmal dahingestellt. Dass soziale Medien, externe Daten-Feeds und Freitextformen noch keine so starke Beachtung finden, mag daran liegen, dass Restriktionen für die Sichtbarkeit von Daten, etwa in Facebook, die Akquise erschweren oder aber dass die Szenarien für eine Nutzung der Daten noch nicht gefunden sind, um einen produktiven Mehrwert daraus zu ziehen.

Da nun einige Quellen für Big-Data vorgestellt wurden, lässt sich auch gleich auf die Vielgestalt der Formatierung eingehen. Neben klassischen, unformatierten Texten kommen häufig JSON (*JavaScript Object Notation*), XML (*Extensible Markup Language*), HTML- oder sogar Byte-Code vor. Gerade wenn man an den Aspekt der Visualisierung denkt, ist es wichtig, Relationen zwischen einzelnen Datensätzen herzustellen, um diese in Abhängigkeit voneinander zu präsentieren. Was in relationalen Datenbanken über simple Queries erreicht

werden kann, bedarf bei Plain-Text-Analysen eines erheblichen Aufwands (siehe Kapitel 8). Viel früher trifft man jedoch bei der Analyse auf die Herausforderung, die gewünschte Information aus jedem einzelnen der vielen Formate herauszufiltern. Des Weiteren gilt zu bedenken, dass sich Formate im Laufe der Zeit auch ändern können. Gerade bei der Auswertung fremder, externer Datenquellen erfolgt meist keine Benachrichtigung über eine Anpassung der Datenstruktur seitens des Datenhalters. Hier ist es wichtig, Auswertungen entsprechend zu überwachen, um Abweichungen frühzeitig festzustellen.

Ist es denn nun gerechtfertigt, von unstrukturierten Daten zu reden? Schließlich weisen ja viele Datensätze eine Struktur auf, nur eben keine feste, einheitliche. Als besserer Begriff wäre hier vielleicht polystrukturiert anzuführen, wie es der Analyst Mike Ferguson in seinem Blog beschreibt (Ferguson). Im späteren Verlauf des Buches, wenn wir zu den Eingabeformaten für die diversen Diagramme kommen, wird sich zeigen, dass diese Vielgestalt einen großen Teil der Arbeit eines Datenanalysten ausmacht, denn die Interpretation der Eingangsdaten einer Analyse variiert nur allzu häufig. Zu diesem Umstand gesellen sich ebenso Fehler in Daten, die schon vor dem Big-Data-Hype bekannt waren. Nathan Yau (Yau, 2010) benennt sechs davon wie folgt:

- Fehlende Werte
- Falsche Beschriftung
- Inkonsistenz
- Tippfehler
- Werte ohne Kontext
- Verteilte Datensätze (über mehrere Quellen hinweg)

Die Komplexität einer Verarbeitung von Daten, die in mehreren, ggf. unbekannten, Strukturen vor- und diesen sechs Umständen unterliegen, erfordert also einen erheblichen Mehraufwand gegenüber der Aufbereitung von normalisierten Daten, z. B. aus einer relationalen Datenbank.

2.2.2 Das vierte V – Veracity

IBM führt ein viertes V ein, das die Richtigkeit und die Echtheit von Daten beschreibt (Zikopoulos et al., 2013). Zwar steigt die Menge an zur Verfügung stehenden Daten nachweislich an, jedoch werden diese häufig durch generierte Inhalte verfälscht, die da sein können:

- Werbung und Spam, die eine einseitige Sicht auf Personen, Produkte oder Vorkommnisse wiedergeben.
- Per Automatismus übersetzte Texte, die häufig grammatischen, sprachlichen und inhaltlichen Fehler aufweisen.
- Veraltete oder falsch kategorisierte Suchergebnisse oder Forenindizes.
- Gezielte Falschaussagen oder Fehlinformationen.

Das beste Beispiel sind die klassischen Falschmeldungen, die sich im Internet manchmal in wenigen Minuten verbreiten. So streute etwa ein junger Brite am 26. Februar 2012 das Gerücht, dass der Schauspieler Rowan Atkinson gestorben sei (Gardner, 2012). In nur drei

Stunden wurde das Gerücht so schnell verteilt, dass sogar auf Wikipedia der Todestag des Schauspielers eingetragen wurde. Einmal mehr zeigt sich hier die Notwendigkeit, die gesammelten Daten vor der Nutzung zu verifizieren und auszusortieren.

2.2.3 Der Verarbeitungsaufwand ist big

Ein weiterer interessanter Ansatz, den Aufwand der Verarbeitung großer Datenmenge als *big* zu sehen, liefert der Autor Pavlo Baron:

„Ich hatte z.B. einen Fall, bei dem es um lächerliche Datenmengen ging, die problemlos auf einen USB-Stick gepasst hätten. Man erwartete allerdings simultane Zugriffszahlen im zweifachen Millionenbereich pro Sekunde. [...] das ist definitiv Big [...].“ (Baron, 2013)

Es ist also aus dieser Perspektive nicht die bloße Größe der Daten, sondern die Komplexität der Aufbereitung und der Informationsgewinnung. Ein gutes Beispiel sind dafür etwa Video-Streams in Kaufhäusern, die das Kaufverhalten von Kunden auswerten sollen. Auch wenn eine einstündige Aufnahme lediglich ein paar Hundert Megabyte groß ist, ist die Schwierigkeit der Implementierung und des Trainierens von situationserkennenden Algorithmen sehr hoch und steht beispielsweise im Gegensatz zu einem einfachen Algorithmus, der lediglich alle *<h1>*-Tags aus einigen Millionen HTML-Seiten auslesen muss. Sind die notwendigen Daten extrahiert, müssen ggf. noch Beziehungen zu anderen Datensätzen hergestellt werden, etwa über einen übereinstimmenden Datumswert, Quellenübereinstimmungen oder, im Optimalfall, über vorliegende IDs. Relationale Daten hingegen verfügen über Schlüsselattribute, die eine Zuordnung von Datensätzen erheblich vereinfachen.

Roger Magoulas von O'Reilly Media gibt eine weitere sehr schöne Definition für Big-Data.

„Big-Data ist, wenn die Daten selbst Teil des Problems werden.“

Dieses Zitat passt besonders gut, da Magoulas gar nicht erst versucht, eine Größendefinition zu geben, sondern einfach sagt, dass man, wenn die Datenmenge für *aktuelle* Verarbeitungsmethoden zu umfangreich wird, von Big-Data spricht. Wenn ein Kunde zu Ihnen kommt und Sie fragt, ob seine Daten nun *big* sind oder nicht, dann antworten Sie doch einfach mit einem der beiden hier gegebenen Zitate.

2.2.4 Sicht der Industrien auf Big-Data

IBM befragte in der bereits in Abschnitt 2.2.1 erwähnten Studie (Schroeck et al., 2012) mehrere Unternehmen nach deren Definition von Big-Data. Eine Auswertung nach Schlagwörtern bestätigte die in den vorigen Abschnitten gegebene Sicht auf den neuen Trend weitestgehend. Auffällig ist, dass der Größenbegriff nicht immer im Sinne von Datengröße verwendet wird. Stattdessen wird etwa in dieser Studie das Schlagwort *Größere Bandbreite an Informationen* verwendet, das sowohl auf große Datenmengen als auch auf mehr oder vielfältigere Informationsquellen hindeutet. 16 Prozent der befragten Unternehmen stellen neue Datenarten und Analysemethoden in den Vordergrund, was wieder für das *big* im Sinne des Verarbeitungsaufwands hindeutet (siehe Abschnitt 2.2.3).

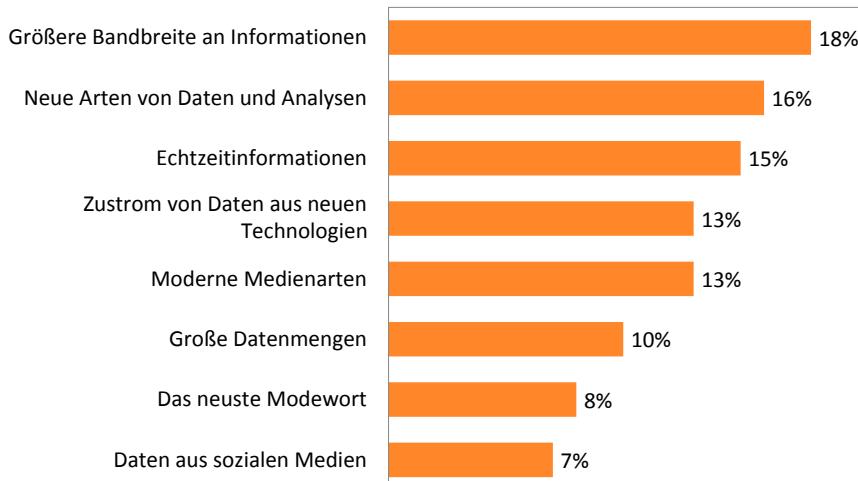


Bild 2.4 Definition von Big-Data (Schroeck et al., 2012)

Überraschend ist, dass Big-Data von einer Mehrheit im Jahre 2012 mehr als Modewort gesehen wurde, als dass es Daten bezeichnet, die aus sozialen Medien stammen. Zwar weist der Begriff *Big-Data* den typischen Buzz-Word-Charakter auf, jedoch ist hier das Ranking entscheidend, dass Daten aus Social-Network-Daten dem Modewortcharakter hintanstellt. Auch diese Studie bestätigt durch die verschiedenen Antworten, dass eine einheitliche Sichtweise auf Definition, Verwendung und Ausprägung des Begriffs noch nicht vorherrscht. Nichtsdestotrotz zeigt sich, dass bestimmte Themengebiete bei einem Eingrenzungsversuch immer wieder auftauchen.

■ 2.3 Eingliederung in BI und Data-Mining

Um die Begriffe BI und Data-Mining in Relation zu Big-Data setzen zu können, gilt es, diese im Vorfeld zu definieren. Kemper, Mehanna & Unger bezeichnen BI recht prägnant als Filter, der Daten in strukturierte Information umwandle (Kemper et al., 2010). Gartner hingegen konstatiert etwas ausführlicher, dass BI ein Überbegriff für Anwendungen, Infrastruktur, Werkzeuge und Best Practices sei, die den Zugriff auf und die Analyse von Informationen ermöglichen, um Entscheidungsfindung und Performance zu erhöhen (Gartner). Hält man sich nun strikt an die Definitionen, besteht der Unterschied zwischen BI und Big-Data darin, dass BI sich auf bereits vorliegende Informationen bezieht, die dazu noch strukturiert sind und sich auf einen eindeutigen Kontext beziehen. Das Ziel von BI und der Big-Data-Bewegung ist jedoch dasselbe, nämlich aus vorhandenen Daten neue Erkenntnisse zu gewinnen, die der Entscheidungsfindung bei vorher definierten Fragestellungen dienen. BI ist jedoch mittlerweile mehr als diese einfache Begriffsdefinition. Es hat sich in den letzten Jahren zu einem festen Prozess samt einem Set aus technischen Werkzeugen entwickelt, um das Berichtswesen in Unternehmen zu automatisieren. Dazu gehören die Datenaufbe-

reitung, die Datenspeicherung in DWHs sowie deren Darstellung aus verschiedenen Perspektiven.

Welche Techniken, Methoden und Arbeitsschritte werden aber nun angewandt, um Informationen aus vorliegenden Daten zu extrahieren? Die Antwort darauf gibt der sogenannte KDD-Prozess (*Knowledge Discovery in Databases*).

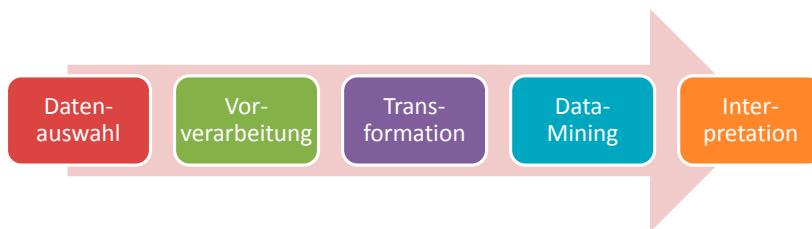


Bild 2.5 Der KDD-Prozess nach (Kononenko et al., 2007)

Der (iterative und interaktive) KDD-Prozess hat das Ziel, gültige, neue, nützliche und verständliche Muster in vorhandenen Daten zu erkennen (Fayyad et al., 1996). Wirft man nun einen Blick auf den vierten Schritt des in Bild 2.5 illustrierten Ablaufs, so ist zu erkennen, dass Data-Mining einen Teil des KDD-Prozesses darstellt. Dieser nimmt gesäuberte, korrigierte und transformierte Daten entgegen, extrahiert aus diesen Muster und Zusammenhänge und gibt diese zur Interpretation frei. Quellen müssen, anders als der Begriff KDD vermuten lässt, nicht zwingend Datenbanken sein, sondern können auch als simple Datensätze gesehen werden, z. B. als Flat-Files, CSV, XML oder Dateisystemstrukturen. Wichtig ist, dass diese bereits im Vorfeld aufbereitet wurden. Zu dieser Aufbereitung (*Preprocessing*) gehören:

- Formattanpassungen (z. B. Datums- und Zahlenformate)
- Korrigieren von Ausreißern (Messfehler, Verarbeitungsfehler, bewusste Falschangabe)
- Auffüllen dünn besetzter Daten



HINWEIS: Ohne nun zu viel verraten zu wollen, möchte ich hier kurz darauf hinweisen, den KDD-Prozess im Hinterkopf zu behalten, wenn wir in Kapitel 8 aus allen behandelten Themen unsere kleine Reporting-Anwendung konstruieren. Sie werden bei deren Entwicklung und Bedienung einige Parallelen feststellen. Ebenso lohnt sich ein Vergleich des KDD-Prozesses mit neuartigen verteilten Verarbeitungsframeworks wie Apache Spark (siehe Abschnitt 3.19), in denen sich einzelne Prozessschritte eins zu eins wiederfinden lassen.

Stellt man nun die drei Begriffserklärungen BI, Data-Mining und Big-Data einander gegenüber, so erkennt man schnell einige Gemeinsamkeiten sowie Unterschiede.

Einerseits werden die Daten bei der Big-Data-Verarbeitung vorher nicht aufbereitet und in einem eigens dafür eingerichteten System, wie beim Data-Mining dem DWH, abgelegt. Das macht den Prozess performanter und schlanker, erfordert allerdings wesentlich flexiblere, aufwendigere Algorithmen zur Mustererkennung. Ein weiterer Nachteil der Big-Data-Verarbeitung gegenüber dem Data-Mining ist, dass Daten bei Letzterem strukturiert in einem

DWH abgelegt sind und somit auch aus anderen Perspektiven betrachtet werden können (Stichwort *Online Analytical Processing Cube*). Soll also in einer Grafik der Gewinn eines Unternehmens samt aller Töchterfirmen angezeigt werden anstatt wie bisher ohne, so müssen im Reporting-Werkzeug lediglich vom Benutzer die bereits ermittelten Gewinne der Töchterfirmen dem Gesamtgewinn hinzugefügt werden. Im Falle von einer Umsetzung auf Basis von Hadoop, müsste die Auswertung programmatisch angepasst werden, sie ist also weniger flexibel, was Änderungen in Abfragen angeht¹. Die Verarbeitung der Daten zeigt deutliche Übereinstimmungen bei Big-Data und Data-Mining. Zwar nutzt Big-Data den Map-Reduce-Algorithmus zum Extrahieren der Daten, jedoch werden für Klassifizierung, Clusteranalyse etc. dieselben Algorithmen (*K-Means*, *CLARA*) verwendet. Die Apache Foundation hat den Vorteil der verteilten Verarbeitung für Data-Mining bereits aufgegriffen und implementiert eine entsprechende Lösung mit Namen Apache Mahout (siehe Abschnitt 4.8).

Zu guter Letzt sollen die Resultate der drei Vorgänge betrachtet werden. Diese weisen auf den ersten Blick starke strukturelle Unterschiede auf. Der Benutzer eines BI-Reporting-Werkzeugs wünscht sich ein Diagramm, das einige Zahlenmengen so darstellt, dass er darin Informationen erkennt, die ihm vorher nicht zugänglich waren. Big-Data wird im Allgemeinen durch den Map-Reduce-Prozess auf Schlüssel-Wert-Paare reduziert und danach auf dem traditionellen Weg (Reporting-Werkzeuge, Geschäftsanwendungen ...) weiterverarbeitet. Neue, flexiblere Frameworks wie YARN oder Spark sind jedoch stark im Kommen. DM ermittelt laut der Beschreibung des KDD-Prozesses Muster und Zusammenhänge, die dann als Rohdaten ebenso kontextabhängig aufbereitet werden müssen. So liefern diese aggregierte Einkäufe einer bestimmten Artikelgruppe über einen Zeitraum t sowie die Seitenaufrufe der Artikel xyz dieser Gruppe. Ein Algorithmus erkennt hier etwa einen Zusammenhang von Seitenaufrufen und Käufen über eine positive Kovarianz. Um dem Benutzer diese Information zugänglich zu machen, müssen diese in irgendeiner Form aufbereitet werden, z.B. als Tabelle, als Information in Textform oder als Diagramm. Auffällig ist, dass sich alle drei Ergebnisse in der fachlichen Betrachtung ähneln, denn jeder Einzelne hat das Ziel, neu gewonnene Informationen zu liefern, die der Entscheidungsfindung dienen. Der technische und fachliche Weg dorthin unterscheidet sich jedoch mitunter sehr.

Bild 2.6 (auf der nächsten Seite) zeigt noch einmal die in den Begriffsdefinitionen gefundenen Übereinstimmungen der drei Themenbereiche BI, Big-Data und Data-Mining. Diese Betrachtung war nun sehr theoretisch. In der Praxis ergänzen sich Big-Data, DWH und BI natürlich. So liefern Big-Data-Technologien die Daten für das Reporting und stellt Schnittstellen zur Verfügung, um die großen Datenmengen ähnlich einem DWH abzurufen. Die bisherige DWH-Infrastruktur wird mit großer Wahrscheinlichkeit noch viele Jahre parallel zu den neuartigen Methoden für die Verarbeitung von Big-Data existieren.

¹ Ausnahmen gibt es auch hier. So können SQL-ähnliche Abfragen über Apache Hive auch dynamisch zur Laufzeit angepasst werden. Dafür ist es allerdings notwendig, dass die Daten in einem festen Schema vorliegen, was jedoch nicht immer dem Normalfall entspricht.

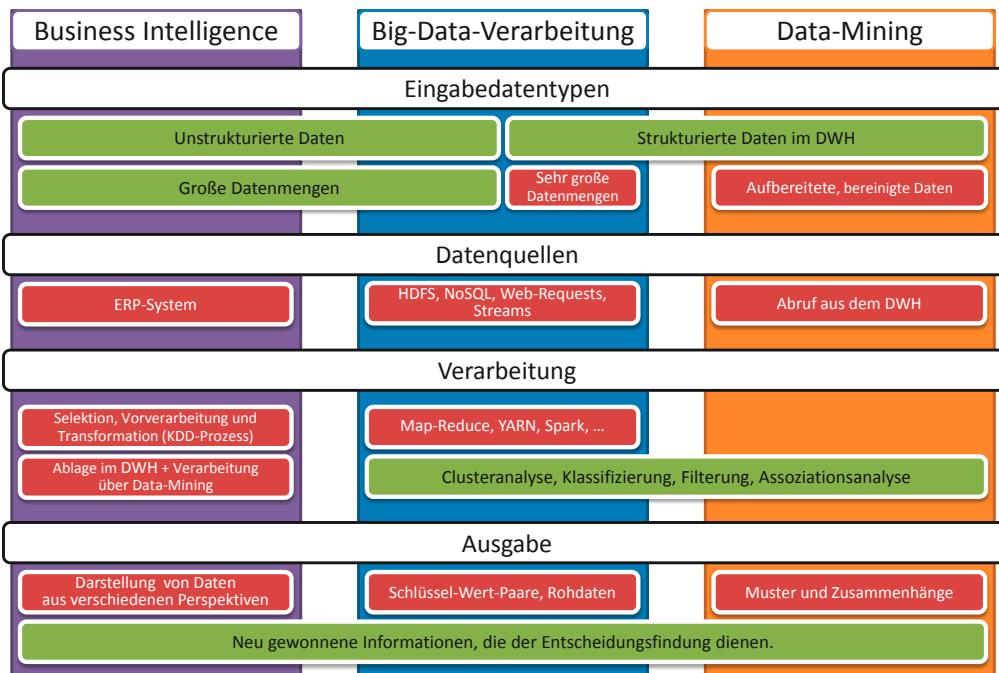


Bild 2.6 Definitionsvergleich von BI, Big-Data und Data-Mining



PRAXISTIPP: Der Trend bei der Bereitstellung großer Datenmengen und traditioneller Daten im DWH oder in relationalen Datenbanken geht dahin, dass alle Datenquellen über ein einheitliches Interface angeboten werden. So werden Abfragen nur noch an eine Datenbank-Engine gerichtet, die die Queries an die entsprechende Stelle weiterleitet. Die eigenständigen Systeme dahinter verlieren bei einem solchen sogenannten föderierten Informationssystem (*Federated Database System*) weder ihre Selbstständigkeit, noch müssen deren Daten zwischen den verschiedenen Systemen ausgetauscht werden.

Index

Symbole

\$HOME 31
\${JAVA_HOME} (Maven) 136
.bashrc 32
/etc/hostname 74, 83
/etc/hosts 70

A

AE *siehe* Analysis-Engine
Aktienkurse 362
Alternativen zu Hadoop 178
Amazon 179
Ambari 80, 182
AMRMClient 146
AMRMClientAsync 146
Anreichern 357, 362, 367
Apache Commons 155, 303
Apache Oozie 108
Apache POI 114
Apache Sentry 290
Application-Master 25, 74, 130, 143
ApplicationsManager 74
ApplicationSubmissionContext 141
Aufbereitung 16
Ausdünnung 328
Azkaban 108

B

BashReduce 178
Beispieldaten 5
BigTable 189
Breadcrumbs 162

Brewer's Theorem *siehe* CAP-Theorem
Bubble-Charts 341
Bucketing 278

C

Calendar-Chart 338
CapacityScheduler 141
CAP-Theorem 190
Cascading 108
Cascading Style Sheet 161
Cassandra 179, 190, 197
Channel (Flume) 183
Chord-Chart 338
Chord-Diagramm 341
Choroplethenkarte 323
Chrome Experiments 333
CLI *siehe* Command Line Interface
Column-Family 196, 216, 219, 221, 223
Combine-Phase 42, 43
Commodity-Hardware 20, 42, 189
commons-fileupload 165
Configuration 135, 140, 157, 220
Configured 97
ContainerLaunchContext 140
ContainerRequests 146
ControlledJobs 108
copyFromLocal 23, 77
copyToLocal 23
core-default.xml 140
core-site.xml 34, 140
CouchBase 179
CouchDB 190
cp 23

D

DAG *siehe* Directed Acyclic Graph
 Data Driven Documents *siehe* D3.js
 Data Lake 21
 Data-Node 21, 34, 69
 Data-Scientist 44, 360
 Data-Visualizer 325
 Data-Warehouse 245, 319, 357
 Datenkompression 37
 Datenlokalität 20
 Datenmigration 194
 Denormalisierung 242
 Deployment Descriptor 82
 Derby 192
 dfs.datanode.data.dir 35
 dfs.datanode.name.dir 35
 dfs.permissions 35
 dfs.replication 35
 Diagrammempfehlungen 370
 Directed Acyclic Graph 108
 Disco 178
 Document Object Model *siehe* DOM
 Document-Store *siehe* Dokumentenorientierte Datenbank
 Dokumentenorientierte Datenbank 192
 DOM 350
 Domäne (Glassfish) 48
 Doug Cutting 9
 Driver 57, 95
 DVD 5

E

Echtheit von Daten *siehe* Veracity
 Eclipse 44
 Entwicklungsumgebung 44
 Ereignisdaten 12
 ext4 21

F

Fehler 39
 Fehlererkennung 173
 Fehler in Daten 13
 Fehlertoleranz 173
 FileSplit 118
 FileStatus 145
 FileSystem 154

final 35
 Flare-Chart 336
 Flume 183
 Föderiertes Informationssystem 18
 Formale Sprachen 378
 Formatieren 37
 Fourth Extended Filesystem *siehe* ext4
 Fragen 5
 Fremdschlüssel (Sqoop) 218
 fs.file.impl 135
 fs.hdfs.impl 135
 Full Profile (JavaEE) 48
 Fully distributed 27

G

GenericOptionsParser 59
 getApplications 170
 getNodeRepots 169
 Github 367
 Glassfish 46
 Goals (Maven) 53
 Google File System 21
 google-gson 355
 Graphen-Datenbanken 192

H

Hadoop 19
 Hadoop Distributed File System *siehe* HDFS
 Hadoop-Ecosystem 181
 Hadoop Process Definition Language 186
 HADOOP_USER_NAME 89
 HBase 184, 193

- Autosharding 199
- BinaryComparator 231
- BinaryPrefixComparator 231
- BitComparator 231
- Bulk Loading 207
- Bytes.toBytes 222
- Cell 224
- ColumnCountGetFilter 230
- ColumnPaginationFilter 230
- ColumnPrefixFilter 229
- ColumnRangeFilter 230
- Comparator 231
- CompareFilter 231
- count 203

- create 202
- delete 203
- Delete 228
- deleteall 203
- DependentColumnFilter 230
- disable 203
- Distributed-Mode 203
- drop 203
- enable 203
- EQUAL 231
- Family 224
- FamilyFilter 229
- Filter 229
- FilterBase 231
- FilterList 232
- FirstKeyOnlyFilter 230
- get 202
- Get 225
- GREATER 231
- GREATER_OR_EQUAL 231
- Hadoop-JARs 204
- Hot Spotting 198
- InclusiveStopFilter 230
- KeyOnlyFilter 230
- LESS 231
- LESS_OR_EQUAL 231
- list 203
- MultipleColumnPrefixFilter 229
- MUST_PASS_ALL 232
- MUST_PASS_ONE 232
- NO_OP 231
- NOT_EQUAL 231
- NullComparator 231
- PageFilter 229
- Paging 226, 234
- PrefixFilter 230
- Pseudo-Distributed-Mode 203
- put 202
- Put 226
- Qualifier 224
- QualifierFilter 230
- RegexStringComparator 231
- Region 198
- Region-Server 198
- Row 224
- RowFilter 230
- Row-key 196
- scan 202
- Schema 197
- setReversed 226, 234
- Shell 201
- SingleColumnValueExcludeFilter 230
- SingleColumnValueFilter 229
- Stand-alone 199
- SubstringComparator 231
- TimestampFilter 229
- Value 224
- ValueFilter 230
- Vergleichsoperatoren 231
- Web-Interface 205
- HBaseAdmin 220
- HBaseConfiguration 220
- hbase-default.xml 220
- hbase-env.sh 200
- hbase-site.xml 200, 205, 220
- HCatalog 210, 246
- HColumnDescriptor 222
- HDFS 20, 21
- hdfs-site.xml 34, 73
- Heat-Map 329
- Herunterladen (HDFS) 155
- HFiles 207
- History Server 69
- Hive 184
 - ADD COLUMNS 280
 - Aggregatfunktionen 268
 - ALTER 278
 - Architektur 248
 - Arithmetische Operatoren 260
 - Authentication-Provider 289, 312
 - Auto-Increment 269, 270
 - Autorisierung und Authentifizierung 289
 - Benutzer 285
 - Benutzerverwaltung 313
 - Bucketing 263
 - Case-Sensitivity 251
 - CHANGE 279
 - CLI 314
 - CLUSTER BY 262
 - Command Line Interface 249
 - COMMENT 251, 278
 - Compiler 248
 - COUNT() 257
 - CREATE DATABASE 251
 - CREATE EXTERNAL TABLE 255
 - CREATE ROLE 286

- CREATE TABLE 253
- CREATE TABLE ... LIKE 253
- CREATE TABLE ... PARTITIONED BY ... 254
- CROSS JOIN 283
- DESCRIBE FUNCTION 267
- Directed Acyclic Graph 248
- DISTINCT 260
- DISTRIBUTE BY 262
- Driver 248
- DROP 281
- DROP ROLE 286
- DROP TABLE 255
- Execution Engine 248
- EXPLAIN 258
- Externe Tabellen 254
- FULL OUTER JOIN 283
- GRANT 288
- Grantor 287
- GRANT ... TO ROLE 287
- GROUP BY 264
- Gruppen 285
- HAVING 277
- HCAT_HOME 314
- hive-default.xml.template 247
- HiveQL 245, 251
- hiveserver 249
- hiveserver2 249, 299
- hive-site.xml 247, 285, 294, 296
- Hive-Web-Interface 249
- IF NOT EXISTS 253
- Import 314
- IMPORT 256
- INSERT INTO TABLE 256
- Interactive Shell Mode 250
- JDBC 296
- JOIN 282
- JPam 295
- KERBEROS 295
- Komplexe Datentypen 252
- LDAP 295
- LEFT OUTER JOIN 283
- LEFT SEMI JOIN 283
- Lightweight Directory Access Protocol
siehe LDAP
- LIMIT 258
- LOAD DATA 256
- LOCATION 254
- Logging 275
- Logische Operatoren 259
- Mathematische Funktionen 266
- Metadaten 309
- Metastore 248
- ORDER BY 261
- OVERWRITE 257
- Paging 296, 299, 306, 310, 312
- PAM 295
- Partitionierte Tabellen 254
- Partition Pruning 261
- PasswdAuthenticationProvider 291
- Pluggable Authentication Module
siehe PAM
- Primitive Datentypen 252
- Privilegien 285
- REPLACE 280
- ResultSet 300
- ResultSetMetaData 311
- REVOKE 288
- RIGHT OUTER JOIN 283
- Rolle 285
- Security 284
- SELECT 257
- SELECT .. AS 258
- SELECT ... WHERE 258
- setCatalog 301
- SET TBLPROPERTIES 281
- SHOW DATABASES 251
- SHOW FUNCTIONS 267
- SHOW GRANT 287
- SHOW ROLES 288
- SHOW TABLES 253
- SORT BY 261
- Stinger-Initiative 246
- String-Funktionen 267
- Subquery 265
- Tabelle klonen 253
- Temporäre Funktionen 271
- TRUNCATE TABLE 255
- UDFType 271
- UNION 265
- USE 251
- User-Defined Functions 269
- View 281
- HiveQL 108, 184
- Hochladen (HDFS) 155
- HPDL *siehe* Hadoop Process Definition Language

HTableDescriptor 222
HTML5 333, 342
HttpFS 24
HttpServlet 165
Hue 249
hung_task_timeout_secs 216
HWI *siehe* Hive-Web-Interface

I

ifconfig 40
Impala 246, 317
ImportTsv 208, 219
include file (JSP) 353
Industrien 7
Infografik 322, 341
In-Memory 176
In-Memory-Datenbanken 192
InputFormat 110, 111
InputFormat-Klassen 60
InputSplit 110, 111, 118
InputStreamReader 346
IPv6 31
isSplittable 115

J

JAAS *siehe* Java Authentication and Authorization Service
Java Authentication and Authorization Service 292
Java Database Connectivity 249
Java Persistence API *siehe* JPA
Java Runtime Environment 29
JavaScript-Validator 347
JDBC *siehe* Java Database Connectivity
JDK 45
jdk.tools 219
JobControl 108
JobTracker 25
Join 244
JPA 355
Jps 39
JRE *siehe* Java Runtime Environment
JSON 346
JSON.parse 351

K

KDD-Prozess *siehe* Knowledge Discovery in Databases
Key-Value-Datenbank 191
KeyValueInputFormat 61
KeyValueTextInputClass 60
Klassifikator 379, 382
Klassifizierung 379
Knowledge Discovery in Databases 16
Kompression *siehe* Datenkompression
Kontextsensitive Diagramme 331

L

Lambda-Expressions 177
Language-Detection *siehe* Sprachenerkennung
Latenzzeit 245
LingPipe 379
LoadIncrementalHFiles 208
LocalResource 140
LocalResourceVisibility 140
LocI-Methode 322
Log-Aggregation 126
Logging (Hadoop) 125, 130
ls 23

M

Machine-Learning 186
Mahout 186
Mapper 55, 97, 100
Map-Phase 42, 43
mapred 59
mapred-site.xml 35, 38
Map-Reduce 20, 42
mapreduce.task.timeout 216
Maschine-Learning 384
Maven 50
Maven-Assembly-Plug-in 53, 131
MD5 198
Mehrdeutigkeiten 379
META-INFO 135
Miscro 178
mkdir 23
MongoDB 179
mv 23
MySQL 192
MySQL-Server 211

N

Name-Node 21, 34, 69
 Nashorn 344
 Natural Language Processing *siehe* NLP
 Neo4j 179, 192
 NLP 377
 Node-Manager 69, 146
 Normalform 242
 NoSQL 189
 Not only SQL *siehe* NoSQL
 NullOutputFormat 113

O

ODBC *siehe* Open Database Connectivity
 Oozie 186
 Open-Data 367
 Open Database Connectivity 249
 OpenNLP 173, 358, 379
 - Abhängigkeiten 380
 - DoccatModel 381
 - DocumentCategorizerME 383
 - getAllResults 383
 - getBestCategory 383
 OpenSSH-Client 71
 OpenSSH-Server 30
 OpenStreetMap 336
 OutputCollector 112
 OutputFormat 110, 113

P

Partitioner 112
 Partitions 112
 PDFBox 114
 PDFInputFormat 114
 Perspektive (Eclipse) 46
 PhantomJS 344
 Pig 185, 318
 Pig Latin 108, 185, 318
 polystrukturiert 13
 Primärschlüssel 242
 Project-Facet 82, 102, 155, 219, 298
 Projekt importieren (Eclipse) 51
 Pseudo distributed 27

Q

Queue 141

R

Random 350
 Random Read/Write 245
 RDBMS 194, 195
 Recommendation-Engine 369
 RecordReader 110, 111
 RecordWriter 110, 119, 121
 Redirect 241
 Reduce-Phase 42, 43
 Reducer 56, 98, 101
 Region-Server 206
 Regular-Expression 231, 260
 Relational Database Management System
siehe RDBMS
 Relationale Datenbank 191
 Relationen herstellen 357
 Repliken 22
 Reporter 112
 Resilient Distributed Datasets 177
 Resource-Manager 25, 69, 74, 130, 140, 141,
 146, 248
 ResourceTracker 74
 REST-API 80
 rm 23
 Rowan Atkinson 13

S

Sandbox 27
 SAP-Hana 190
 SAXParserException 40
 Scala (Programmiersprache) 176
 Scalable Vector Graphics *siehe* SVG
 Scale-out 193
 Scale-up 193
 Scheduler 74
 Schemafreiheit 189
 Scoop 209
 scrollen (Ubuntu) 65
 Secure Shell *siehe* SSH
 Sekundärschlüssel 242
 Sensordaten 362
 Sentiment-Analysis 377, 379, 384
 SequenceFileInputFormat 61

SequenceFileOutputFormat 113
 setInputFormatClass 60
 setJarByClass 92
 setMapperClass 60
 setOutputFormatClass 60
 setOutputKeyClass 61
 setOutputValueClass 61
 setReducerClass 60
 SFTP 297
 Shark 177
 Shuffling 112
 SingleColumnValueFilter 228, 229, 230, 232
 Single Point of Failure 25, 68
 Sink (Flume) 183
 Social Media 12
 Source (Flume) 183
 Spaltenorientierte Datenbank 191
 Spark 176, 192
 Split-Phase 42
 Splits 110
 Sprachen 377
 Sprachenerkennung 378
 Sqoop 183, 198, 219, 227, 240, 246, 314
 Sqoop2 209, 415
 sqoop-env.sh 210
 sqoop-env-template.sh 210
 SSH 30
 SSH File Transfer Protocol *siehe* SFTP
 Stack Traces 65
 Standalone 27
 start-all.sh 38
 Starten eines Map-Reduce-Jobs 64
 StoreFiles 209
 Storm 176
 StringBuilder 238
 sudo 29
 SVG 342
 systemPath (Maven) 136

T

Tag-Cloud 328, 336
 Talend Open Studio 109
 Testdaten 95, 347
 Textanalyse 357, 376
 TextInputFormat 61
 Text-Mining 376
 TextOutputFormat 113

Thrift 207, 249
 ToolRunner 59
 touchz 24
 Trainingsdaten 173, 379, 381
 Transaktionsdaten 12
 Tree-Map 337

U

Ubuntu Server 27
 UDF *siehe* User-Defined Functions
 UIMA 358, 377, 386
 - Abhängigkeiten 388
 - Analysis-Engine 393
 - AnnotationIndex 393
 - CAS 402
 - Eclipse-Plug-in 387
 - getDocumentText 393
 - JCas 390
 - JCasGen 389
 - Primitive 394
 - Testkonfiguration 398
 - Type System Definition 388
 Unstructured Information Management
 Architecture *siehe* UIMA
 Upload-Servlet 163
 URL-Dekodierung 160, 162
 URL-Enkodierung 160, 162

V

V *siehe* VVV
 Validator 365, 377
 Variety 11
 Velocity 11
 Veracity 13
 Verarbeitung (Big-Data) 171
 Video-Tutorials 5
 View (Eclipse) 46
 Visual Analytics 326
 Visualisierung 321
 - 3D-Diagramme 333
 - Assoziation 332
 - Audio 334
 - Aufmerksamkeit lenken 329
 - Circos 341
 - Computertomographie 333
 - D3.js 341, 342, 343, 344, 353

- Datameer 341
 - Datenstrukturen 327
 - Datumswerte 338
 - Diagrammarten 336
 - Dimensionen 334
 - Frameworks 341
 - Geografische Daten 328
 - Hierarchische Daten 327
 - infogr.am 341
 - InfoViz 341
 - Interaktion 330
 - Interaktivität 334
 - JGraph 342
 - Klassische Werkzeuge 340
 - Kontextsensitivität 335
 - Lineare Daten 327
 - Magnetresonanztomographie 333
 - Matlab 341
 - m-n-Relation 336
 - Netzstruktur 327
 - Precog 341
 - Processing 341
 - ReportGrid 341
 - RGB 329
 - R-Project 341
 - Tabellarische Daten 327
 - Visualisierungsempfehlungen 357
- VMware Player 27
 Volume 10
 Vorverarbeitung 189
 VVV 10

W

- waitForCompletion 61, 93, 106
 WebGL 333

- Web Graphics Library *siehe* WebGL
 Web-Interface 40, 64, 127, 150
 Web Profile (JavaEE) 48
 Wikipedia 8
 Windows-Binaries (Hadoop) 86, 412
 WinSCP 50, 63
 winutils.exe 409
 Word-Cloud 336
- X**
- XAMPP 211
 XML-Validator 388
- Y**
- YARN 20, 25
 YarnClient 138, 154, 168
 YarnConfiguration 140
 YARN-Kompatibilität 35
yarn.nodemanager.aux-services 36
yarn.nodemanager.aux-services.mapreduce.shuffle.class 36
yarn.nodemanager.delete-debug-delay-sec 36
yarn.nodemanager.delete.debug-delay-sec 151
yarn.nodemanager.vmem-pmem-ratio 36
yarn-site.xml 36, 74
 Yet Another Resource Negotiator
siehe YARN

Z

- Zookeeper 185, 200, 220
 - Quorum 220, 233