technische universität dortmund

IT in Produktion und Logistik

Bachelorarbeit

Entwicklung einer Methode zur Prüfung der Eignung von NoSQL-Datenbanksystemen zur Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie

Christian Matthias Fahrenholz
Immatrikulationsnummer: 158611
Datum der Abgabe: 18.02.2019

Gutachter:
Name des Erstgutachters: Univ.-Prof. Dr.-Ing. Markus Rabe
Name des Zweitgutachters: M. Sc. Joachim Hunker

Technische Universität Dortmund
Fakultät für Maschinenbau
Fachgebiet IT in Produktion und Logistik
http://www.itpl.mb.tu-dortmund.de

Inhaltsverzeichnis

1 Einleitung

Die Automobilindustrie gilt als das beste Beispiel für die Veränderungen in der Welt der Produktion. Durch den Einsatz von neuen Technologien, wie zum Beispiel Elektromobilität oder Leichtbau, und einer personalisierten Produktion nimmt die Produktvielfalt enorm zu (vgl. Bauernhansl 2014, S.13). Gleichzeitig muss sich die Automobilindustrie der Herausforderung stellen, die Wirtschaftlichkeit der Produktion von derartig individualisierten Fahrzeugen zu sichern (vgl. Steegmüller und Zürn 2014, S.103). Es sind die Technologien der Industrie 4.0, vor allem der Einsatz von vernetzten Produktionssystemen, die „den Konflikt zwischen Flexibilisierung und Rationalisierung" lösen sollen (Kellner et al. 2018, S.279).

Mit wachsendem Vernetzungsgrad der Produktion werden die anfallenden Datenmengen immer umfangreicher und unstrukturierter, da sie aus unterschiedlichen Quellen stammen (vgl. Winkelhake 2017, S.52). Trotzdem müssen sie in Echtzeit ausgewertet und analysiert werden (vgl. Meier und Kaufmann 2016, S.13). Diese drei Eigenschaften lassen sich durch den Begriff Big Data zusammenfassen (vgl. Fasel und Meier 2016, S.6). Mit der relationalen Datenbanktechnologie ist es nahezu unmöglich, solche Datenströme zu verarbeiten (vgl. Celko 2014, S.14). Für diesen Zweck wurden NoSQL-Datenbanken entwickelt, da sie für den Einsatz bei Big Data Anwendungen besser geeignet sind (vgl. Vaish 2013, S.11). Die bedeutendsten Unterschiede zu den relationalen Datenbanken sind, dass die Datenspeicherung nicht mehr in Tabellen erfolgt und als Datenbanksprache nicht mehr SQL (Structured Query Language) verwendet wird (vgl. Meier und Kaufmann 2016, S. 18). Die NoSQL-Datenbanken sind in mehrere Typen aufgeteilt, zu denen die Schlüssel-Wert-Datenbanken, die Graphdatenbanken, oder auch die Spaltenfamilien- oder Dokument-Datenbanken zählen (vgl. Fasel und Meier 2016, S.12). Bei ihrem Einsatz müssen NoSQL-Datenbanksysteme die relationalen Datenbanksysteme nicht unbedingt ersetzen, sondern können auch in Verbindung mit ihnen eingesetzt werden (vgl. McCreary und Kelly 2014, S.6).

Das Ziel dieser Arbeit ist die Entwicklung einer Methode, mit der die Eignung verschiedener NoSQL-Datenbanksysteme in Hinblick auf die Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie geprüft werden kann. Für die Methode werden unterschiedliche Bewertungskriterien identifiziert, die an Datenbanken gestellt werden. Diese Kriterien sind überwiegend qualitativ und lassen sich nicht quantitativ beschreiben. Aus diesem Grund wurde für die Methode das Bewertungsverfahren der Nutzwertanalyse gewählt. Bei der Nutzwertanalyse werden qualitative Kriterien durch eine subjektive Einschätzung quantifiziert und dadurch vergleichbar gemacht (vgl. Haag et al. 2011, S.327). Für diese Einschätzung werden mithilfe von Fachliteratur die Eigenschaften und Besonderheiten der unterschiedlichen Architekturansätze für NoSQL-Datenbanken herausgearbeitet. Da nicht alle Kriterien gleichbedeutend sind, wird eine Gewichtung vorgenommen, die sich an den Anforderungen von

vernetzten Produktionssystemen der Automobilindustrie richtet. Die entwickelte Methode soll dem Benutzer eine Handlungsempfehlung bieten, welcher Datenbanktyp am besten zu einer vorliegenden Anwendung passt.

Vor der Auswahl dieses Bewertungsverfahrens werden andere Vorgehen der Technologiebewertung definiert und hinsichtlich ihrer Anwendbarkeit für diese Methode beurteilt. Um die richtigen Bewertungskriterien von NoSQL-Datenbanken auszuwählen und den Erfüllungsgrad der verschiedenen Datenbanktypen bezüglich dieser Bewertungskriterien zu bestimmen, befasst sich ein Abschnitt der Arbeit mit der Theorie von Datenbanken. In diesem Abschnitt werden vor allem NoSQL-Datenbanken mit ihren unterschiedlichen Architekturansätzen thematisiert. In dieser Arbeit werden die Ansätze der Schlüssel-Wert-Datenbanken, der Graphdatenbanken, der Dokument- und der Spaltenfamilien-Datenbanken behandelt. Es wird außerdem herausgearbeitet, was unter Big Data zu verstehen ist und warum dieser Treiber zur Entwicklung von NoSQL-Datenbanken geführt hat. Zum besseren Verständnis dieses komplexen Themas werden zudem die theoretischen Grundlagen der Datenbanken erläutert. Ebenfalls wichtig für die Identifikation und auch die Gewichtung der Bewertungskriterien ist die Kenntnis der Besonderheiten der Automobilindustrie in Hinblick auf die Produktion. Diese Besonderheiten werden im ersten Abschnitt der Arbeit zusammen mit der Beschreibung von vernetzten Produktionssystemen aufgezeigt. Um die Gründe für den Einsatz solcher Produktionssystemen zu erläutern, wird zu aller erst ein Einblick in die Industrie 4.0 gegeben. Im Anschluss an die Methodenentwicklung wird die Methode mittels Anwendung auf selbst gewählte Fallbeispiele validiert. Der erste Anwendungsfall ist ein vernetztes Produktionssystem mit der identischen Gewichtung zu der, die im Kapitel der Methodenentwicklung erarbeitet wurde. Für das zweite Anwendungsbeispiel muss die Gewichtung aufgrund von veränderten Anforderungen des behandelten vernetzten Produktionssystems angepasst werden. Abschließend werden die in dieser Arbeit gewonnenen Erkenntnisse zusammengefasst und es wird ein Ausblick gegeben.

2 Vernetzte Produktionssysteme

Dieses Kapitel beschäftigt in Abschnitt 2.1 mit dem Einfluss der Industrie 4.0 auf Produktionssysteme. Die daraus entstehenden vernetzten Produktionssysteme werden in Abschnitt 2.2 definiert und Abschnitt 2.3 befasst sich mit den Besonderheiten der Automobilindustrie. Auf Basis dieser Aufarbeitung werden die Gewichtungsfaktoren in Abschnitt 4.3.3 verteilt.

2.1 Der Einfluss der Industrie 4.0

Durch den steigenden Grad der Vernetzung durchläuft der industrielle Sektor einen enormen Wandel, weshalb dieser Umbruch als vierte industrielle Revolution, auch „Industrie 4.0“, bezeichnet wird (vgl. Soder 2017, S. 14 f.). Bevor auf die Industrie 4.0 eingegangen wird, ist es sinnvoll die historische Entwicklung der Industrie zu betrachten, um zu verstehen, wie die Revolutionen aufeinander aufgebaut sind (vgl. Bauernhansl 2014, S. 5). Der erste große Wandel der Industrie ereignete sich Ende des 18. Jahrhunderts durch die Entwicklung der Dampfmaschine, wodurch mechanische Produktionsanlagen entstanden. Die zweite Revolution folgte Anfang des 20. Jahrhunderts und ihr Kern liegt in der Einführung der Massenproduktion unter Zuhilfenahme von elektrischer Energie. (vgl. Wagner 2018, S. 3 f.) Durch den Einsatz von Elektronik und der Nutzung der Informations- und Kommunikationstechnologie ließen sich die Prozesse in der Produktion zunehmend automatisieren. Diese Revolution begann Anfang der 60er Jahre und mit ihr wurde die Serienproduktion variantenreicher Produkte möglich. (vgl. Bauernhansl 2014, S. 7) Nach Huber und Kaiser (vgl. 2017, S. 18) wird in der vierten Revolution die reale mit der virtuellen Welt durch „Cyber-Physikalische Systeme“ verbunden und es findet eine weltweite Vernetzung statt. Da eine Kommunikation nicht nur zwischen einer Maschine und dem integrierten System stattfindet, sondern zwischen allen Systemen, sind komplett „neue Produktionsmethoden und -prozesse“ denkbar (Soder 2017, S. 15). Dabei dient das Internet als Technologie, mit der der hohe Vernetzungsgrad zwischen den Produktionsanlagen, ihren Komponenten, den Produkten und ihren Logistikketten erreicht werden kann (vgl. Wagner 2018, S. 4). Die Unternehmen erreichen durch Auflösung starrer Produktionsstrukturen und Umsetzung von Produktionsanlagen, die sich selbst regulieren und autonom handeln können, eminente Verbesserungen in den Bereichen Kosten, Zeit und Effizienz (vgl. Soder 2017, S. 15). Im Verlaufe dieser vier Revolutionen hat sich die Rolle des Menschen den technischen Möglichkeiten angepasst. Er hat sich von einem Maschinenbediener über die Rolle des Spezialisten und des Befähigers zum Gestalter von Maschinen und Wertschöpfung entwickelt. (vgl. Huber 2018, S. 13)

Nach Schlick et al. (vgl. 2014, S. 76) liegt die größte Veränderung in der Vorgehenswiese, wie Ziele erreicht werden. Durch die Industrie 4.0 ist es neben dem Prinzip der Leistungssteigerung möglich, eine Optimierung durch die Reduzierung der Medienbrüche zu erzielen (vgl. Schlick et al. 2014, S. 76). Das Ziel der Produktivitätssteigerung bei gleichzeitiger Einhaltung der geforderten Qualität wurde vor

der Industrie 4.0 einzig durch die Identifikation von Engpässen und anschließender Leistungssteigerung der identifizierten Prozessschritte erreicht. Allerdings lässt sich das theoretische Maximum von Prozessketten nicht erreichen. Der Informationsfluss gewinnt für Prozessketten aufgrund von Trends wie beispielsweise der ansteigenden Produktindividualisierung bis zur Losgröße eins oder der Produktion in globalisierten Wertschöpfungsnetzwerken immer mehr an Bedeutung. (vgl. Schlick et al. 2014, S. 77) Beim Einsatz vieler Systeme, z.B. zur Planung und Produktion, können jedoch Medienbrüche entstehen, da die Systeme eventuell nicht aufeinander zugreifen können. Der Fortschritt ist folglich durch den Menschen limitiert, da dieser für die manuelle Prüfung des Medienbruchs zuständig ist. Es wird durch den Ansatz von Industrie 4.0, Prozesse transparenter zu machen und somit Medienbrüche zu reduzieren, eine Beschleunigung der Prozessketten erreicht. Für diesen Ansatz werden zahlreiche Technologien eingesetzt, die den Grad der Vernetzung erhöhen. (vgl. Schlick et al. 2014, S. 78) Huber (vgl. 2018, S. 6) identifiziert einige technologische Treiber, die mit der Industrie 4.0 verbunden sind und „bestehende Prozesse, Strategien und Unternehmen gravierend verändern“ (Huber 2018, S. 17) können:

- Daten, Software und Smart Data: Die Verarbeitung von Daten durch Software führt zu Smart Data, die Zusammenhänge darstellen und somit von Bedeutung sind (vgl. Huber 2018, S. 18).
- Big Data: Big Data Systeme werten große Datenmengen schnell aus, um zu Erkenntnissen zu gelangen, die Voraussetzung für Unternehmensentscheidungen sind (vgl. Huber 2018, S. 22).
- Künstliche Intelligenz/Maschinelles Lernen: Mithilfe ausreichender Rechenkapazität können Maschinen eigene Kompetenz aufbauen und beispielsweise Muster aus Datensätzen erkennen oder Optimierungen vornehmen (vgl. Huber 2018, S. 27).
- Cyber-Physical Systems (CPS): Produkte, die aus einem intelligenten Zusammenschluss von Hard- und Software bestehen und miteinander kommunizieren können (vgl. Huber 2018, S. 30).
- Sensitive Roboter: Roboter, die mit anderen revolutionären Technologien zusammenwachsen und immer intelligenter und autonomer werden (vgl. Huber 2018, S. 33).
- Additive Manufacturing: flexibles Verfahren, das ein Bauteil durch schichtweises Hinzufügen des Werkstoffes erzeugt (vgl. Huber 2018, S. 40 f.).

Plattformen: Plattformen besitzen keine klare Definition. Sie setzen sich aus einer Kommunikations- und Interaktionsplattform, einem Trustcenter und der physikalischen Logistik zusammen und bieten einem System alle geforderten Funktionalitäten. Ein verbreitetes Beispiel sind die Internet-of-Things-Plattformen. (vgl. Huber 2018, S. 44)

2.2 Begriffsbestimmung vernetzter Produktionssysteme

Produktionssysteme lassen sich in mehrere Komponenten unterteilen. Zu den technischen Elementen zählen die Fabrikhülle, die darin zum Einsatz kommenden Informations- und Automatisierungssysteme und alle benötigten Betriebsmittel. Die Organisation vom Aufbau und Ablauf wird mit den Methoden und Maßnahmen im organisationalen Teil zusammengefasst und die Arbeitsstelle mit dem dafür benötigten Wissen stellt die menschliche Komponente dar. (vgl. Gronau 2014, S. 280 f.) Im Falle von veränderten Anforderungen des Absatzmarktes oder neuer Technologien wird das Produktionssystem „in seinen Elementen, deren Struktur und den stattfindenden Prozessen“ (Kellner et al. 2018, S. 291) an die Veränderungen angepasst. Dabei können entweder Teile des Produktionskonzepts, das ganze Konzept oder die Form des kompletten Unternehmens angepasst werden (vgl. Kellner et al. 2018, S. 291). Dieser Wandel soll am Beispiel der Automobilindustrie verdeutlicht werden. Durch den Bedarf nach Elektromobilität, Leichtbau und einer Produktion, die personalisierte Fahrzeuge regional fabriziert, steigt die Produktvielfalt massiv an. Mit dem Anstieg der Vielfalt sinkt die Anzahl vertriebener Fahrzeuge pro Modell und Variante, wodurch der Bedarf nach einem neuen Produktionskonzept entsteht, welches eine nachhaltige Wertschöpfung sicherstellt und die veränderten Anforderungen erfüllt. Um sich zeitnah und effektiv auf neue Veränderungen einzustellen, müssen Unternehmen flexibler und wandlungsfähiger werden. (vgl. Bauernhansl 2014, S. 13) Nach Bauernhansl (vgl. 2014, S. 15) stellt die Einführung von sich selbst organisierenden und optimierenden Produktionsfraktalen, die untereinander kommunizieren, die Lösung dar.

Die Anpassungsfähigkeit von Produktionssystemen kann durch den Einsatz von Cyber-Physical Systems erhöht werden (vgl. Gronau 2014, S. 279). Solch ein System setzt sich aus mehreren informationstechnischen und physischen Elementen zusammen, die über das Internet verbunden sind. Über diese Plattform werden Daten gesendet, verarbeitet und empfangen. Die Informationsverwertung dient als Grundlage zur Auswahl einer entsprechenden Handlung, welche ebenfalls vom CPS durchgeführt wird. (vgl. Kellner et al. 2018, S. 292) Zu den Komponenten eines CPS gehören unter anderem eine Menge dezentraler Sensoren, die reale Zustände im Prozess erfassen und diese Daten in digitaler Form beliebig zur Verfügung stellen können. Die Erfolgsfaktoren und Potenziale dieser Technologie lassen sich nach Schallow et al. (vgl. 2018, S. 21) auf die Technik und den Menschen aufteilen. Neben der Aufnahme genauer, echtzeitnaher Prozessdaten zur Analyse und Optimierung von Wertströmen und der Vernetzung von Produkten, Ressourcen und Prozessen zur Erzeugung eines echtzeitnahen Abbildes der Fabrik, lassen sich auf der Seite der Technik die CPS für ein effizienteres Fehlermanagement einsetzen. Wenn ein Fehler passiert, kann er sofort identifiziert werden und eine Weiterverarbeitung des defekten Bauteils wird ausgeschlossen, um Folgekosten zu vermeiden. Außerdem wird der Fehler analysiert und es werden Verbesserungsmaßnahmen abgeleitet. Des Weiteren wird der Mensch durch eine fähigkeitsgerechte Aufgabenverteilung unterstützt. (vgl. Schallow et al. 2018, S. 22 f.)

Beispielsweise lassen sich so effizientere und ergonomischere Arbeitsprozesse gestalten (vgl. Schallow et al. 2018, S. 24). Laut Frank und Riess (vgl. 2015, S. 10) werden Cyber-Physische Produktionssysteme (CPPS) zukünftige Szenarien in Produktion und Logistik als intelligente Produkte und Betriebsmittel beherrschen, denn sie ermöglichen eine dezentrale und reaktionsfähige Steuerung dieser beiden Bereiche und setzen vermehrt auf die Informationen von dezentralen Sensoren.

Um die in Kapitel 2.1 beschriebenen Medienbrüche zu verringern, können verschiedene Technologieparadigmen zum Einsatz kommen, die jeweils mit verschiedenen Technologien umgesetzt werden können (vgl. Schlick et al. 2014, S. 59). Das erste Paradigma ist das intelligente Produkt, durch welches alle relevanten Informationen, wie zum Beispiel die Produktionsparameter oder die erforderlichen Konfigurationen für Produktionsanlagen, „zur richtigen Zeit am richtigen Ort" (Schlick et al. 2014, S. 60) bereitgestellt werden, um weiterverarbeitet zu werden. Abgesehen von aktuellen Informationen werden zudem die absolvierten Arbeitsschritte oder die tatsächliche Merkmalsausprägungen gespeichert. Bei Flüssigkeiten oder Kleinstteilen ist es nicht sinnvoll, das Produkt intelligent zu machen, weshalb die nächst größere Transporteinheit mit Intelligenz ausgestattet wird. Falls dadurch ein Werkstückträger zum intelligenten Produkt wird, kann dieser beispielsweise Arbeitspläne enthalten oder aktualisierte Informationen wie Eilaufträge zur Verfügung stellen. (vgl. Schlick et al. 2014, S. 60) Das Paradigma der intelligenten Maschine ist nicht so verständlich wie das intelligente Produkt, da verschiedene Stufen mit der Fokussierung auf unterschiedliche Phasen des Lebenszyklus der Maschine vorliegen. Beispielsweise zählen die Prozesstransparenz und die Optimierung des Qualitätsniveaus zu den Zielen in der Phase des Betriebs. (vgl. Schlick et al. 2014, S. 61) Als letztes Paradigma beschreiben Schlick et al. (vgl. 2014, S. 62) den assistierten Bediener in Form einer Mensch-Maschine-Schnittstelle, z.B. einem Tablet-Computer, als situationsbedingten Filterungsmechanismus, um jederzeit an gewünschte Informationen zu gelangen. Durch diese Paradigmen haben CPPS mit Kommunikationssystemen immer mehr Gemeinsamkeiten. Je größer die Vernetzung der Elemente ist, desto größer ist der Wert des Netzwerkes und je leistungsfähiger die Rechner werden, desto mehr Daten können gewonnen werden. Gepaart mit der Eigenschaft der Dezentralisierung und Autonomie sind dies die Treiber zur Industrie 4.0. (vgl. Bauernhansl 2014, S. 18) Das Resultat dieser Treiber ist die Smarte Fabrik, welche sich mit den CPS organisieren lässt (vgl. Bauernhansl 2014, S. 16). Für eine Fabrik definieren Westkämper und Löffler (vgl. 2016, S. 107) vier grundsätzliche Handlungsfelder:

- High Performance: Nimmt Bezug auf technische, organisatorische und methodische Prozesse der Produktion, die durch den Fortschritt vor Produkten und Prozessen immer leistungsfähiger werden. (vgl. Westkämper und Löffler 2016, S. 104)

- Kompetenz und Wissen: Wissensintegration hat Effizienzsteigerung und Erhöhung der qualitativen Zuverlässigkeit zur Folge, weshalb Digitale Fabriken ununterbrochen neues Wissen aufnehmen (vgl. Westkämper und Löffler 2016, S. 105)
- Effizienz der Ressourcen: Fokus liegt auf Vermeidung von Reibungsverlusten zwischen Prozessen und Schnittstellen und der Abnahme des Ressourceneinsatzes (vgl. Westkämper und Löffler 2016, S. 105 f.)
- Kundennähe: Örtliche Nähe von Fabrik zu Mitarbeitern oder Kunden vorteilhaft, doch belastend für die Umwelt durch Anbindung zu Lieferanten oder Dienstleistern (vgl. Westkämper und Löffler 2016, S. 106)

Die Nachhaltigkeitsziele einer Produktion lassen sich allgemein durch eine „hohe ökonomische, ökologische und soziale Effektivität und Effizienz“ (vgl. Westkämper und Löffler 2016, S. 63) beschreiben. Durch Verbindung dieser Ziele mit den charakterisierten Handlungsfeldern lassen sich vier Typen von nachhaltigen Fabriken der Zukunft identifizieren (Abbildung 2.1). Jede dieser Typen besitzt vor dem Hintergrund der digitalen Umgebung der Industrie 4.0 bestimmte Eigenschaften. (vgl. Westkämper und Löffler 2016, S. 106)

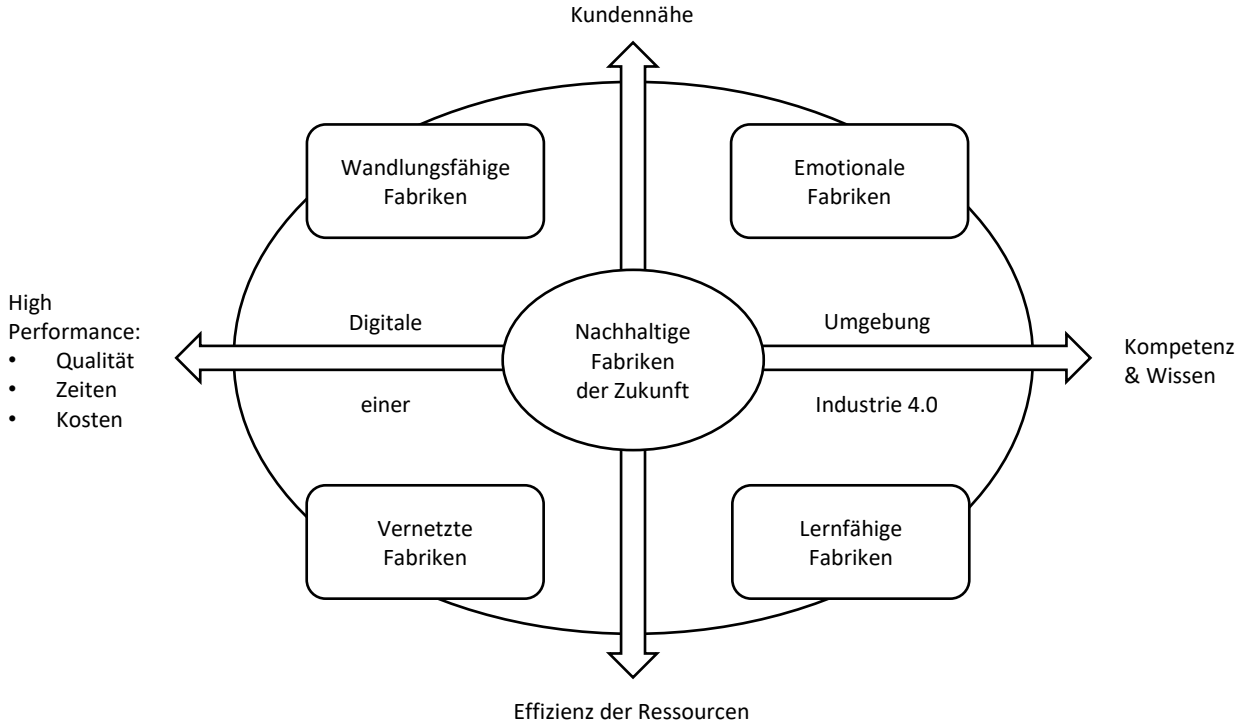

Abbildung 2.1: Typisierung von Fabriken der Zukunft (nach Westkämper und Löffler 2016, Abb 5.24)

Die vier Handlungsfelder werden durch die Pfeile symbolisiert und die verschiedenen Fabriktypen werden je nach Ausprägung dieser Handlungsfelder aufgeteilt, wodurch folgende Fabriktypen entstehen:

- Wandlungsfähige Fabrik: Fabrik, die sich nah am Kunden orientiert und gleichzeitig eine hoch performante Produktion anvisiert (vgl. Westkämper und Löffler 2016, S. 107).
- Vernetzte High-Performance Fabrik: Fabrik, die sich aus gleichzeitigem Bestreben von Effizienz in den Bereichen Produktion und Ressourceneinsatz auszeichnet. Dieser Typ nutzt alle verfügbaren Ressourcen, um das maximale Ergebnis und die höchsten Leistungen zu erhalten. (vgl. Westkämper und Löffler 2016, S. 107)
- Lernfähige Fabrik: Fabrik, die sich durch Wissensintegration und einer steigenden Ressourceneffizienz auszeichnet (vgl. Westkämper und Löffler 2016, S. 107).
- Emotionale Fabrik: Fabrik, die die Kundennähe erfüllt und sich an humanen Kompetenzen orientiert und dadurch eine hohe Kundenbindung besitzt (vgl. Westkämper und Löffler 2016, S. 107).

Der Fokus dieser Arbeit liegt auf vernetzten Produktionssystemen. Da die Effektivität und Effizienz eines Systems von der Effektivität und Effizienz der darin zum Einsatz kommenden Leistungseinheiten abhängt, lässt sich für die Produktion der Schluss ziehen, dass das Maximum nur bei verlustfreier Operation und Kooperation der Leistungseinheiten, sowie der Strukturauslegung auf Höchstleistungen erreicht werden kann (vgl. Westkämper und Löffler 2016, S. 127 f.). Jedoch resultiert aus diesem Grad der Vernetzung laut Diesner (vgl. 2018, S. 46) eine neuartige Komplexität in den Bereichen Technik und Organisation. Um die Vernetzung zur Optimierung oder zur Bildung neuer Geschäftsmöglichkeiten zu nutzen, ist es essenziell, dass das Fundament der einer smarten Fabrik (Smart Factory), welches sich aus der Transparenz und Reaktionsfähigkeit zusammensetzt, sowohl vom Fertigungsmitarbeiter als auch vom Management verstanden und danach gehandelt wird (vgl. Diesner 2018, S. 46). Bei der vernetzten Produktion werden nicht nur Netzwerke optimiert und logistisch vernetzt, sondern sie beinhaltet die Vernetzung aller „organisatorischen, technischen und […] wissensbezogenen Wirkungslinien zwischen Leistungseinheiten“ (Westkämper und Löffler 2016, S. 128) und die umfassende Zusammenarbeit aller involvierten Personen (vgl. Westkämper und Löffler 2016, S. 128). Allerdings existieren laut Affenzeller et al. (vgl. 2018, S. 86) noch keine Standards einer umfangreichen Vernetzung. Als wichtige Voraussetzung lässt sich die Verschmelzung von Bereichen beschreiben, die bisher weitgehend autonom waren. Außerdem müssen unterschiedliche Protokolle zusammengeführt und neue Schnittstellen und Standards eingeführt werden (vgl. Seidemann 2015, S. 306). Zu den Erfolgsfaktoren einer Smart Factory gehören die Produktion von hochindividualisierten Produkten in geringen Stückzahlen bis hin zur Losgröße eins, die optimale Ausnutzung von Ressourcen und die hohe Prozessgeschwindigkeit (vgl. Bauernhansl 2014, S. 18). Nach Huber (vgl. 2018, S. 85) lässt sich die Losgröße eins wirtschaftlich ausschließlich durch Prozesse erreichen, die sich durch Transparenz und Steuerung in Echtzeit auszeichnen, weshalb die Prozesse in Zukunft papierlos sein werden. (vgl. Huber 2018, S. 85) Papiergestützte Prozesse können eine Vielzahl von Verschwendungspotenzialen beinhalten, wie zum Beispiel

Suchtätigkeiten oder der redundanten Aufbewahrung von Daten, die bereits in gespeicherter Form vorliegen (vgl. Müller et al. 2011, S. 141). Der Mensch wird sich in der Smart Factory von der Bearbeitung wertschöpfender Prozesse entfernen und zunehmend mehr steuernde Aufgaben übernehmen. Entgegen der allgemeinen Vorstellung wird die Smart Factory nicht menschenleer sein. (vgl. Huber 2018, S. 85)

2.3 Die Besonderheiten der Automobilindustrie

Bevor auf die heutigen Produktionssysteme in der Automobilindustrie eingegangen wird, ist eine Betrachtung der historischen Entwicklung von Produktionssystemen sinnvoll. Aufgrund elementarer Veränderungen der Umwelt durchliefen die Systeme eine Anzahl von Revolutionen, auf die im Folgenden eingegangen wird (vgl. Wildemann 2017, S. 162). Der erste große Wandel wurde durch die Einführung des Taylorismus vollzogen, der die handwerkliche Produktion ablöste. Zu den Kernelementen des Taylorismus zählen die konsequente Arbeitsteilung und Standardisierung von Prozessen und deren Planung, Steuerung und Formalisierung. (vgl. Wildemann 2016, S. 165) Das Produktionssystem von Ford stellt eine Erweiterung zum Taylorismus dar. Durch dieses System wurden mit dem Fließprinzip und der Standardisierung die Grundsteine für die Massenproduktion gelegt. Jeder Mitarbeiter wurde für einen festen Arbeitsplatz eingesetzt, an dem er ein standardisiertes Produkt bearbeitet, das auf einem Fließband transportiert wurde. Durch die Vorgabe der Bandgeschwindigkeit ließ sich der Arbeitsrhythmus von der Leistungsfähigkeit des Menschen entkoppeln. (vgl. Wildemann 2017, S. 166) Allerdings war dieses Produktionssystem kapitalintensiv und unflexibel und wurde durch die Einführung des Toyota Produktionssystems (TPS) revolutioniert (vgl. Wildemann 2017, S. 167). Das TPS setzte auf eine bessere Wirtschaftlichkeit der Produktion, indem jegliche Verschwendungen beseitigt wurden. In diesem Produktionssystem wurde der Mitarbeiter ins Zentrum gestellt, um ihn in den Prozess der kontinuierlichen Verbesserung mit einzubinden. (vgl. Wildemann 2017, S. 167) Dieser Prozess hatte eine schlanke Organisation und eine hohe Flexibilität des Unternehmens zur Folge (vgl. Wildemann 2017, S. 168). Die Verbesserungsmaßnahmen des TPS wurden von vielen Herstellern lediglich im Fertigungssystem umgesetzt und nicht auf das komplette Produktionssystem angewandt, weshalb die erhofften Potenziale nicht erreicht werden konnten. Aus diesem Grund wurden modulare Produktionssysteme entwickelt, die die Produktion in ihre Einzelteile unterteilen und diejenigen Elemente bündeln, die zur Erfüllung einer bestimmten Aufgabe beitragen. (vgl. Wildemann 2017, S. 171) Durch die Zuordnung von bestimmten Arbeitsschritten zu einem Segment wird der Koordinationsaufwand reduziert und es werden Synergiepotenziale freigesetzt. Ein weiterer Bestandteil der Fertigungssegmentierung stellt die modulare Organisation dar. (vgl. Wildemann 2017, S. 172) Dabei wird die Ablauforganisation standardisiert, wodurch Doppelarbeit verhindert wird und die Prozesseffizienz angesichts der Senkung von Koordinations- und Führungstätigkeiten zunimmt. Als Beispiel für solche Standards sind normierte Fertigungsabläufe oder festgelegte Arbeitsrichtlinien zu nennen. (vgl. Wildemann 2017, S. 173) Das

Diagramm in Abbildung 2.2 beschreibt die Rolle des Automobilherstellers in Bezug auf die Wertschöpfungstiefe und die Fahrzeugvielfalt im Verlauf der verschiedenen Revolutionen. In der ersten Revolution wurde der überwiegende Teil der Wertschöpfung von den Herstellern selbst übernommen. Jedoch wurden dadurch nur sehr wenige Fahrzeugvarianten von einem Hersteller gefertigt. Den Extremfall stellt Ford mit der Fertigung von nur einem Modell dar. (vgl. Hüttenrauch und Baum 2008, S. 13) Der Hersteller akzeptierte die Lieferanten als wertschöpfende Partner und übertrug einen Großteil der Wertschöpfung auf selbige. Außerdem konnte sich der Kunde sein Auto selbst zusammenstellen, wodurch die Fahrzeugvielfalt erhöht wurde. (vgl. Hüttenrauch und Baum 2008, S. 28 ff.) Schließlich führt die Modularisierung in der dritten Revolution zu einer immensen Vielfalt der angebotenen Fahrzeuge. Die beteiligten Unternehmen fokussieren sich auf die Wertschöpfung als Ganzes, anstatt auf den Bereich der Produktion. (vgl. Hüttenrauch und Baum 2008, S. 145)

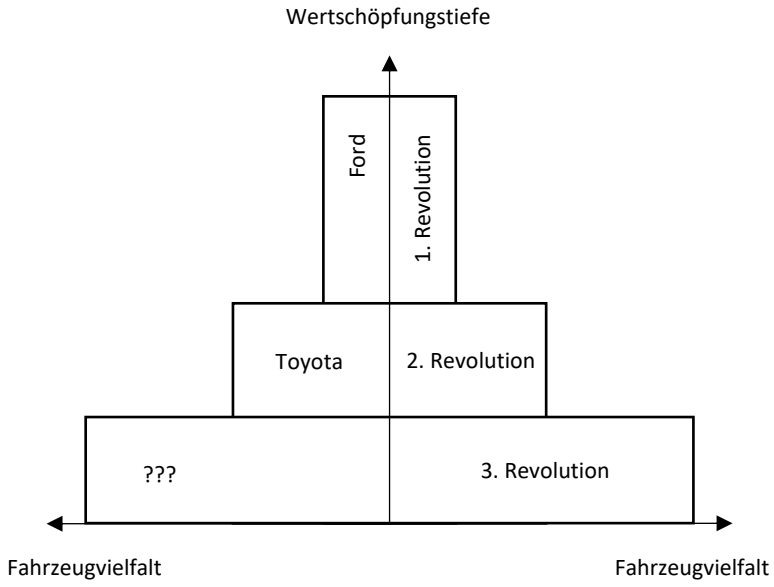

Abbildung 2.2: Positionierung der Automobilhersteller in der dritten Revolution der Automobilindustrie (nach Hüttenrauch und Baum 2008, Abb. 54.)

Göpfert et al. (vgl. 2017, S. 9) leiten aus den aktuellen Entwicklungen neun branchenspezifische Trends für die Automobilindustrie ab. Sie sollen im Folgenden kurz vorgestellt werden:

- Weitere Zunahme der Globalisierung: Weltweiter Ausbau von Wertschöpfungsnetzwerken aus Kunden, Herstellern und Zulieferern (vgl. Göpfert et al. 2017, S. 11).
- Steigende Kundenorientierung: Hersteller sind durch die Sättigung von Märkten und der globalen Konkurrenz gezwungen, sich den Bedürfnissen den Kunden anzupassen (vgl. Göpfert et al. 2017, S. 11).
- Anhaltender Kostendruck: Die Zahlungsbereitschaft der Kunden sinkt und Konkurrenten aus dem Ausland drücken den Preis zusätzlich (vgl. Göpfert et al. 2017, S. 12).

- Anstieg der Bedeutung von Umweltaspekten: Reaktion auf veränderte Anforderungen, die beispielsweise aus der Debatte über den Klimawandel entstanden sind, mit zahlreichen Innovationen (vgl. Göpfert et al. 2017, S. 12).
- Hoher Innovationsdruck/Anstieg des Anteils der Elektronikkomponenten im Fahrzeug: Fahrzeuge benötigen Produktinnovationen, um sich erfolgreich am Markt durchsetzen zu können (vgl. Göpfert et al. 2017, S. 13).
- Neue Wachstumsmärkte: Die klassischen Volumenmärkte Nordamerika und Westeuropa stagnieren und dafür wachsen viele andere Märkte, die erschlossen werden müssen (vgl. Göpfert et al. 2017, S. 14).
- Anstieg der angebotenen Fahrzeugmodelle und -derivate: Zunehmender Entwurf von verschiedenen Fahrzeugklassen, um den Bedürfnissen der Kunden entgegenzukommen (vgl. Göpfert et al. 2017, S. 15).
- Individualisierung der Fahrzeuge hinsichtlich ihrer Ausstattung: Steigende Individualisierungsmöglichkeiten, um dem Kunden sein Wunschauto zu liefern (vgl. Göpfert et al. 2017, S. 16).
- Neuausrichtung der Wertschöpfungskette: Neuausrichtung durch Entwicklungen der Unternehmenszusammenschlüsse, der wachsende Trend des Outsourcings und der zunehmenden Kooperation konkurrierender Hersteller (vgl. Göpfert et al. 2017, S. 17).

Diese Trends resultieren in einer steigenden Komplexität in der Fertigung. Aufgrund des zunehmenden Anstiegs der Fahrzeugvarianten mussten die bisherigen Entwicklungskonzepte überdacht und neue umgesetzt werden. Mit der Einführung des Plattformkonzepts konnten mehrere Fahrzeugmodelle auf einer einheitlichen Plattform konstruiert und gefertigt werden. (vgl. Neubauer 2012, S. 8) Dadurch mussten allerdings die Produktpalette durch plattformübergreifende Modelle ergänzt werden. Aus dieser Forderung wurde die Modulstrategie entwickelt, die eine höhere Produktstandardisierung als das Plattformkonzept besitzt. Die daraus entstandenen Fahrzeuge werden durch einheitliche Baugruppen gekennzeichnet. Volkswagen gilt als der erste Automobilhersteller, der die Komplexitätskosten durch die Einführung der sogenannten Modulbaukästen eindämmte. (vgl. Neubauer 2012, S. 9) Dieses Modell des Modellbaukastens ist bei Kleinserien, beispielsweise einem Rennwagen, jedoch nicht wirtschaftlich und muss für diese Fahrzeugtypen überarbeitet werden. Eine wirtschaftliche Alternative bietet die Fertigung auf Basis einer stabilen Rahmenstruktur, die aus Rohren besteht und mit Beplankungsteilen bestückt werden, die die eigentliche Fahrzeugkontur bilden. Die Beplankungsteile lassen sich bei dem Rahmenstrukturkonzept relativ einfach austauschen, wodurch zum Beispiel ein defekter Rennwagen mit einer neuen Karosserie ausgestattet werden kann, falls der Rahmen unversehrt bleibt. (vgl. Neubauer 2012, S. 9) Allgemein wirkt die Modularisierung der Komplexität entgegen, wodurch neue Fahrzeugtypen immer schneller und günstiger entwickelt werden, wodurch sich der Aufwand, der durch die Module entsteht, schneller bezahlt macht. (vgl. Neubauer 2012, S. 10) Laut Neubauer

(vgl. 2012, S. 10) werden die Fertigungslinien einzelner Modelle durch Mix-Fertigungen unterschiedlicher Fahrzeugtypen abgelöst, welche auf Konzepten basieren, die „einen hohen Grad an Flexibilität bei konstanten Stückkosten sicherstellen“ (Neubauer 2012, S. 10). Obwohl die Produktivitätsoptimierung und die Beherrschung der Variantenvielfalt der Automobilindustrie in den letzten Jahren vorangetrieben wurden, sind die Wertschöpfungsstrukturen in den letzten hundert Jahren nahezu konstant geblieben, da immer noch nach dem Tayloristischen Prinzip gefertigt wird, dessen Kern das Band und der Takt ist. Das Band beschränkt durch dessen Verkettung der Arbeitsschritte die Variantenzahl und -flexibilität und der Takt limitiert die Produktionsmenge und die Flexibilität, was nicht mehr marktkonform ist. Um diese Begrenzung aufzulösen, müssen Band und Takt entkoppelt werden und flexibel vernetzbare, skalierbare Prozessmodule zum Einsatz kommen. (vgl. Bauernhansl 2014, S. 21)

Ein Anwendungsbeispiel für die Smart Factory, das unterschiedliche Technologien der Industrie 4.0 verbindet und in dieser Form von einem deutschen Hersteller umgesetzt wird, ist die vollständige Realisierung einer Montagelinie über fahrerlose Transportfahrzeuge (FTF). Den FTF ist es möglich, sich ohne den Einsatz einer physischen Leitspur frei und sehr positionsgenau durch die Halle zu navigieren. Für die Navigation werden Sensorsysteme zur präzisen Positionsbestimmung innerhalb der Halle genutzt. Dafür wird der aktuelle Standort in Echtzeit mit der Umgebungskarte abgeglichen. Aus diesem Grund hat eine Änderung des Hallenlayouts keine Auswirkung auf die autonome Navigation, wodurch eine schnelle Reaktion auf Veränderungen am Markt oder am Produkt und die zeitnahe Anpassung der Montageabläufe möglich ist. Durch das Hinzufügen oder Entfernen von FTF, lässt sich je nach Auftragslage der optimale Betriebspunkt einstellen. Es kommen außerdem taktungebundene Materialbereitstellungen für die logistische Versorgung und taktungebundene Montagehilfen für den reibungslosen Montageablauf zum Einsatz. Mit den neuen Einsatzmöglichkeiten leisten diese fahrerlosen Transportsysteme einen großen Beitrag zur flexiblen und wandlungsfähigen Gestaltung von Produktionssystemen. (vgl. Wibbe und Rohde 2017, S. 49) Ein weiteres Beispiel für die Implementierung solcher Transportsysteme stellt die Fertigung des neuen Audi A8 in Neckarsulm dar. Da das neue Modell im Vergleich zum Alten an Komplexität zugenommen hat und die Arbeitsumfänge dadurch um 15% gestiegen sind, musste der Montageprozess in eine zweite Halle verlegt werden. In dieser neuen Halle befindet sich das Montageband im Obergeschoss und die Logistik im Erdgeschoss. Für die Beförderung der Bauteile, wie zum Beispiel eines Dachsystems, von der Logistik zum Montageband werden ebenfalls fahrerlose Transportsysteme (FTS) eingesetzt, die mithilfe eines Aufzugs zwischen den Etagen transportiert werden. (vgl. Rumpelt 2016, S. 9)

3 Datenbanksysteme

In diesem Kapitel werden die Grundlagen für die Identifizierung der Bewertungskriterien und die Einteilung der Datenbanktypen auf diese Kriterien gelegt. Dafür soll in Abschnitt 3.1 ein grundlegendes Verständnis von Datenbanken geschaffen werden. Abschnitt 3.2 beschäftigt sich mit den relationalen Datenbanken und in Abschnitt 3.3 wird durch die Beschreibung des Begriffes Big Data und dessen Herausforderungen die Dringlichkeit eines alternativen Datenbankmodells verdeutlicht. Abschnitt 3.4.1 definiert die allgemeinen Informationen zu NoSQL-Datenbanken und geht auf vielfach verwendete Begriffe ein. In Abschnitt 3.4.2 bis 3.4.5 werden die NoSQL-Core-Modelle ausführlich beschrieben. Diese Beschreibung dient der späteren Identifikation und Bewertung der Kriterien. Abschnitt 3.5 befasst sich mit der Möglichkeit, mehrere Datenbanktypen für eine Problemstellung zu kombinieren.

3.1 Grundlagen von Datenbanken

Informationen in Form von Daten gelten in Unternehmen als Produktionsfaktor und Entscheidungsgrundlage, weshalb sie für alle Angehörigen von großer Bedeutung sind. Aus diesem Grund wird an sie ein hoher Qualitätsanspruch gestellt und sie müssen wie auch andere Produktionsfaktoren „geplant, gesteuert, überwacht und kontrolliert“ (Meier und Kaufmann 2016, S. 2) werden. (vgl. Meier und Kaufmann 2016, S. 2) Für die Speicherung und Verwaltung der Daten werden Datenbanksysteme verwendet, die aus einem Datenbankmanagementsystem und einer Anzahl von Datenbanken bestehen (vgl. Saake et al. 2018, S. 9). Abbildung 3.1 soll ein Datenbanksystem mit seinen Komponenten zur Datenspeicherung und -verwaltung aufzeigen. Der Rand des Datenbankmanagementsystems ist dabei die Schnittstelle, auf die mehrere Zugriffe durchgeführt werden. Nach Schicker (vgl. 2017, S. 3) wird eine Datenbank von einem eigenen Datenbankverwaltungssystem (Database Management System, DBMS) verwaltet und in ihr werden Daten gesammelt, die in logischen Beziehungen zueinanderstehen. Somit werden solche Daten, die nicht zusammengehören, getrennt verwaltet (Schicker 2017, S. 3).

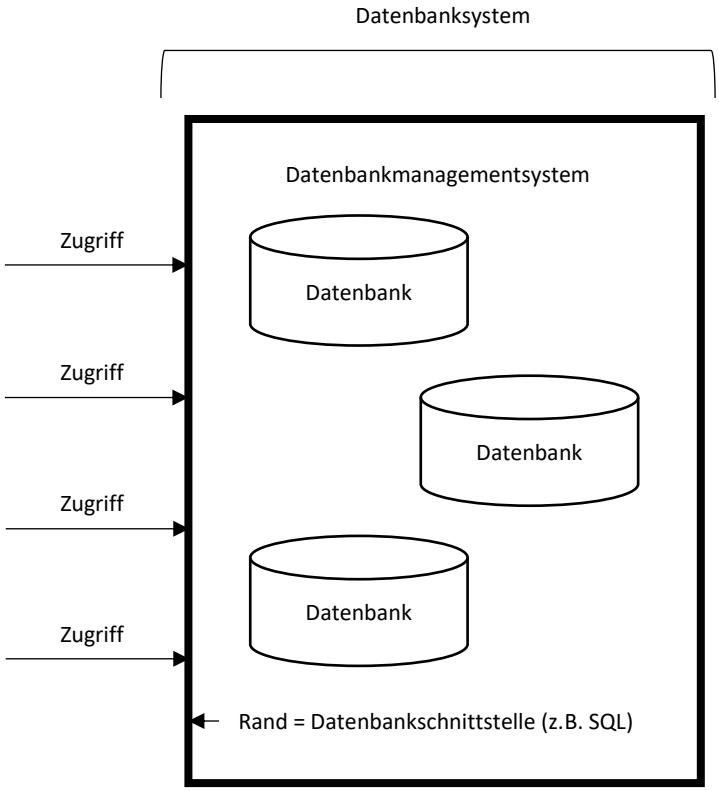

Abbildung 3.1: Abbildung eines Datenbanksystems mit seinen Bestandteilen (nach Schicker 2017, Abb. 1.2)

Anwender können über die logische Schnittstelle des DBMS die Daten auswerten oder verändern (vgl. Meier und Kaufmann 2016, S. 2 f.). Logische Zugriffe werden von dem DBMS in physische umgesetzt. Die physischen Zugriffe bewirken das Lesen oder Schreiben von Daten auf dem Speichermedium. Durch die automatische Umsetzung muss der Benutzer nicht mit der physischen Struktur der Daten vertraut sein. (vgl. Schicker 2017, S. 4) Die physische Speicherstruktur wird nur verändert, wenn das DBMS dadurch leistungsfähiger wird (vgl. Kemper und Eickler 2015, S. 24). Diese Änderung ist die Aufgabe des DBMSs, da es für die optimale physische Speicherung zu sorgen hat (vgl. Schicker 2017, S. 7). Die wichtigste Datenbankschnittstelle ist die Programmiersprache Structured Query Language (SQL), mit der auf relationale Datenbanken zugegriffen werden kann (vgl. Schicker 2017, S. 4). Insgesamt soll der Einsatz eines DBMS zu einer effektiven Verwaltung von großen Datenmengen führen (vgl. Saake et al. 2018, S. 9). Durch das DBMS ist es mehreren Benutzern möglich, gleichzeitig auf die Datenbank zuzugreifen (vgl. Unterstein und Matthiessen 2012, S. 8). Dabei werden die Zugriffsrechte jedes Benutzers von dem DBMS verwaltet und es besteht die Möglichkeit, sensible Daten auszublenden (vgl. Kemper und Eickler 2015, S. 23). In einem Fehlerfall gehen die Daten nicht verloren, sondern können wiederhergestellt werden. Auch die Integrität der Daten wird nicht verletzt, da spezifische Regeln bei dem Umgang mit einer Datenbank gelten. (vgl. Codd 1982, S. 114) Nach Schicker (vgl. 2017, S. 8) bezieht sich die Integrität auf die Korrektheit von Daten. Beispielsweise wird ein atomarer Verarbeitungsvorgang erst ausgeführt, wenn er die Datenbank in einem konsistenten Zustand hinterlässt (vgl. Kemper und Eickler 2015, S. 23). Außerdem sinken durch ein DBMS die Entwicklungskosten, da es eine

komfortable Schnittstelle zur Datenbank darstellt und somit die Zeit für die Entwicklung von neuen Anwendungen und die Anfälligkeit für Fehler verringert (vgl. Kemper und Eickler 2015, S. 23). Der zweite Teil eines Datenbanksystems ist die Datenbank, mit denen Daten strukturiert und dauerhaft gespeichert werden (vgl. Meier und Kaufmann 2016, S. 9). Laut Schicker (vgl. 2017, S. 11) wird im Prinzip zwischen vier Datenbankmodellen unterschieden, die in ihrem logischen Aufbau voneinander abweichen. Die in den 60er Jahren entstandenen hierarchischen und netzwerkartigen Datenbanken wurden durch die relationalen Datenbanken abgelöst, die in den 70er und 80er Jahren entwickelt wurden. (vgl. Saake et al. 2018, S. 11 f.) Ergänzt werden diese Modelle Ende der 90er Jahre durch die objektorientierten Datenbanken, die jedoch keine große Bedeutung erlangen konnten. SQL-Datenbanken werden in der Praxis häufig eingesetzt, doch sie haben Probleme bei der Analyse von heterogenen Daten bei webbasierten Dienstleistungen in Echtzeit. Für diese Probleme werden NoSQL-Ansätze eingesetzt. (vgl. Meier und Kaufmann 2016, S. 3) Diese wurden in den vergangenen Jahren entwickelt, um große Datenmengen zu verwalten (vgl. Schicker 2017, S. 12).

Für eine effiziente Datenspeicherung werden verschiedene Anforderungen an eine Datenbank gestellt. Die redundante Abspeicherung von gleichen Daten kostet Speicherplatz und Verwaltungsaufwand und ist deshalb möglichst zu vermeiden (vgl. Schicker 2017, S. 6). Wenn mehrere Anwender auf eine Datenbank zugreifen, sollen die jeweiligen Zugriffe abgekapselt von den anderen durchgeführt werden. Der Anwender soll nicht bemerken, dass er nicht der einzige Benutzer ist. (vgl. Wiese 2015, S. 4) Zur Kontrolle soll die Datenbank stets Informationen über den derzeitigen Zustand, wie etwa die Benutzeranzahl oder die Auslastung des DBMS. Des Weiteren soll das Systemverhalten immer gesteuert oder verbessert werden können. (vgl. Schicker 2017, S. 10) Eine der Hauptanforderungen an eine Datenbank ist die Möglichkeit, Daten langfristig zu speichern. Es gibt allerdings Datenbanktypen, bei denen nur ein Teil der Daten für die Langzeitspeicherung vorgesehen ist und der Rest auf flüchtigen Speichern, wie dem Hauptspeicher, abgespeichert wird. (vgl. Wiese 2015, S. 4) Eine weitere zentrale Forderung ist die Integrität der Daten. Das DBMS muss beispielsweise für die richtige physische Speicherung sicherstellen. Um logisch korrekt zu sein, müssen redundante Daten den gleichen Wert besitzen und eine Änderung von mehrfach vorhandenen Daten muss sich auf alle Replikate beziehen. (vgl. Schicker 2017, S. 8) Von Datenbanken, die mit enormen Datenmengen umgehen müssen, wird außerdem gefordert, dass sie skalierbar sind. Nur durch die Verteilung der Daten auf ein Netzwerk von Datenbanken und einem hohen Grad an Parallelisierung können diese Daten verarbeitet werden. (vgl. Wiese 2015, S. 3)

Es werden zwischen verschiedenen Zugriffsarten auf Datenbanken unterschieden. Eine Abfrage (query) hat keine Änderung der Datenbank zur Folge und betrifft einen Ausschnitt der Datenbank, der an den Anwender zurückgegeben wird. Bei einer Mutation wird ein Ausschnitt einer Datenbank geändert oder gelöscht. Die Transaktion beschreibt eine Operation, die die Konsistenz der Datenbank

erhält. Als Konsistenz wird die Freiheit von Widersprüchen bezeichnet. Dabei kann die Transaktion aus mehreren Abfragen und Mutationen bestehen. Ein Beispiel für eine Transaktion ist eine Überweisung, die nur ausgeführt wird, wenn eine Summe von einem Konto abgebucht und auf ein anderes Konto eingezahlt wird. Um die Konsistenz zu erhalten, ist eine Transaktion ein atomarer Zugriff. (vgl. Schicker 2017, S. 16 f.) Nach Wiese (vgl. 2015, S. 3) kann sie entweder ganz oder gar nicht ausgeführt werden.

Es existieren verschiedene Modelle zur Konsistenzgewährung. Im Folgenden wird das ACID-Prinzip erläutert, welches strikt von relationalen Datenbanken erfüllt wird. In Kapitel 3.4.1 wird auf das BASE-Prinzip eingegangen, das bei NoSQL-Technologien zum Einsatz kommt. (vgl. Meier 2016, S. 36) Nach Unterstein und Matthiessen (vgl. 2012, S. 99 f.) beschreibt ACID die vier Eigenschaften Atomarität (atomicity), Konsistenz (consistency), Isoliertheit (isolation) und Dauerhaftigkeit (durability), die mit dem Begriff der Transaktion verbunden werden. Atomarität bedeutet, dass eine Transaktion ganz oder gar nicht ausgeführt wird (vgl. Celko 2015, S. 19). Dabei sind die Zwischenzustände von einzelnen Operationen einer Transaktion nicht für andere Transaktionen erkennbar (vgl. Meier und Kaufmann 2016, S. 136). Der Begriff Konsistenz beschreibt die Eigenschaft, dass sich die Datenbank jeweils am Anfang und am Ende einer Transaktion in einem konsistenten Zustand befindet (vgl. Celko 2015, S. 20). Die Isoliertheit fordert, dass mehrere parallel ablaufende Transaktionen dasselbe Ergebnis erzielen, das sich bei einer seriellen Ausführung ergeben würde (vgl. Meier 2018, S. 30). Dafür muss jede Transaktion so ablaufen, „als sei sie allein im System“ (Schicker 2017, S. 19). Die Dauerhaftigkeit bezieht sich laut Meier (vgl. 2016, S. 31) auf die Tatsache, dass der Datenbankzustand nach einer Transaktion dauerhaft ist und erst durch eine folgende Transaktion verändert werden kann. So müssen sich die Daten nach einem Ausfall, wie zum Beispiel einem Rechnerabsturz oder einem Brand, wiederherstellen lassen (vgl. Schicker 2017, S. 19). Mit diesem Konsistenzmodell sollen mehrere Anwender ohne Komplikationen auf die Daten zugreifen können (vgl. Schicker 2017, S. 19).

3.2 Relationale Datenbanken

Nach Meier und Kaufmann (vgl. 2016, S. 4) ist Structured Query Language (SQL) die bedeutendste Datenbankschnittstelle. Diese Programmiersprache wird dafür verwendet, um auf relationale Datenbanken zuzugreifen (vgl. Unterstein und Matthiessen 2012, S. 35), die ihre Daten oder Informationen in einem Relationenmodell abspeichern (vgl. Meier und Kaufmann 2016, S. 3). Bei diesem Modell werden Daten und deren Beziehungen in Tabellen ausgedrückt, wobei „jede Tabelle als Menge ungeordneter Tupel aufgefasst“ wird (Meier und Kaufmann 2016, S. 6). Die Begriffe Relation und Tabelle werden synonym verwendet (vgl. Kemper und Eickler 2015, S. 74). Neben den Zeilen, die den Tupeln der Relation entsprechen, besteht jede Tabelle aus einer Anzahl von unterschiedlichen Attributen, die ebenso wie der Tabellenname eindeutig bezeichnet werden. Dabei steht eine Spalte für ein Attribut. (vgl. Meier und Kaufmann 2016, S. 6) Nach Saake et al. (vgl. 2018, S. 19) muss von einer Tabelle die Integritätsbedingung erfüllt werden, dass ein Attribut existiert, das jedes vorhandene Tupel eindeutig

identifizieren kann. In Abbildung 3.2 ist eine beispielhafte Tabelle zu sehen, in der Daten von einigen Studenten gespeichert wurden.

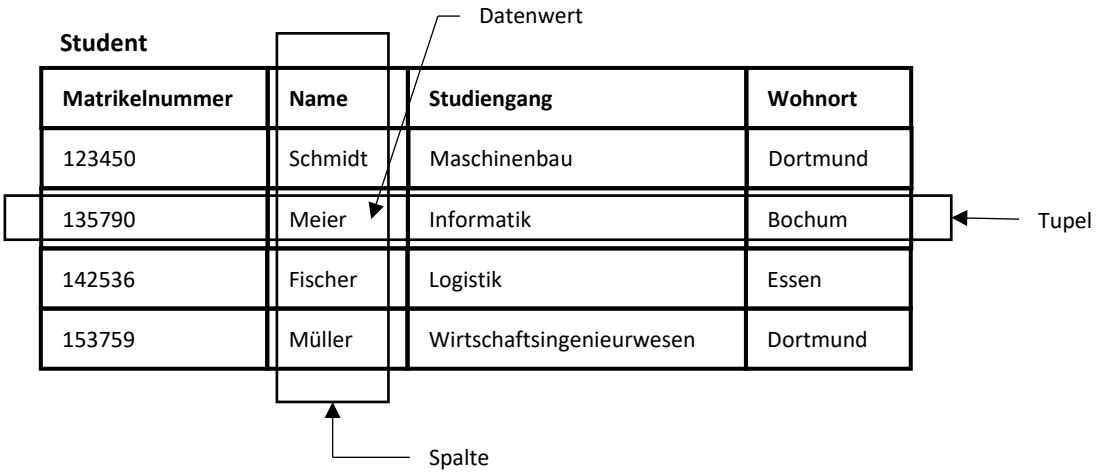

Abbildung 3.2: Beispielhafte Relation (nach Meier und Kaufmann 2016, Abb. 1.3)

Der Identifikationsschlüssel für das vorliegende Beispiel wäre die Matrikelnummer, die jeder Student am Anfang des Studiums zugewiesen bekommt. Es ist möglich, dass identische Datenwerte mehrmals in einer Spalte erscheinen können (Beispiel: Wert „Dortmund“ in Spalte „Wohnort“). Falls in einer Relation kein Attribut zur genauen Identifikation verwendet werden kann, lässt sich der Schlüssel auch durch eine Kombination von Attributen darstellen. Allerdings muss die Anzahl der kombinierten Attribute minimal sein. (vgl. Meier und Kaufmann 2016, S. 5) Zwar ist eine partielle Übereinstimmung von Tupeln zulässig, jedoch dürfen laut Schicker (vgl. 2017, S. 27) keine zwei Tupel mit den identischen Attributwerten existieren.

Bei der Erstellung einer neuen Tabelle werden lediglich die Anzahl der Attribute vorgegeben, in dessen Spalten sich neue Tupel eintragen lassen (vgl. Schicker 2017, S. 25). Die Zeilen und Spalten müssen dabei nicht geordnet werden, da deren Reihenfolgen beim Zugriff auf die Relation nicht ausgenutzt werden (vgl. Schicker 2017, S. 27). Die möglichen Werte eines Attributs werden in einem Wertebereich, der sogenannten Domäne, definiert (vgl. Unterstein und Matthiessen 2012, S. 21). So ist es beispielsweise nicht möglich, dass ein nicht existierender Studiengang oder eine Zahl in der Namensspalte eingetragen wird. Soll der Tabelle ein neues Attribut hinzugefügt werden, muss die Relation laut Schicker (vgl. 2017, S. 27) in eine neue Relation mit dem gewünschten Attribut überführt werden. Bei der Tabellendeklaration kann mit der Angabe „null“ oder „not null“ bestimmt werden, ob fehlende Werte (Nullwerte) für ein Spalte zulässig sind. Ein Nullwert wird benutzt, wenn ein Datenwert nicht bekannt oder nicht zutreffend ist und kann nicht zur Beschreibung eines Schlüsselattributs verwendet werden. (vgl. Saake et al. 2018, S. 23)

Als Schnittstelle zwischen dem Anwendungsprogramm und der Datenbank wird die Abfragesprache SQL benutzt (vgl. Unterstein und Matthiessen 2012, S. 35). Der Anwender muss bei Zugriffen nicht angeben, wie die Auswertung der Daten vorgenommen wird, sondern welche Daten ihn interessieren. Aus diesem Grund wird SQL als deklarativ bezeichnet. (vgl. Kemper und Eickler 2015, S. 113) Zur Verdeutlichung wird in Abbildung 3.3 eine Anfrage an die Relation aus Abbildung 3.2 beschrieben, die sich aus den Klauseln select, from und where zusammensetzt. Die select-Klausel beschreibt den gesuchten Namen der Studenten aus der Tabelle „Student“ (from-Klausel), die in Dortmund wohnen (where-Klausel). (vgl. Saake et al. 2018, S. 212)

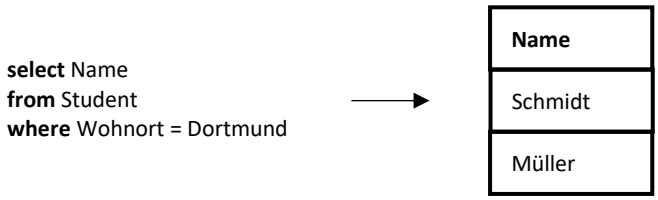

Abbildung 3.3: Beispielhafte Anfrage mit Resultat

Das Datenbanksystem wertet die Anfrage aus und gibt das Ergebnis in einer Tabelle wieder, sodass sich der Benutzer nicht um die Programmierung der Abläufe zur Informationsbereitstellung, wie bei den prozeduralen Datenbanksprachen, kümmern muss (vgl. Meier und Kaufmann 2016, S. 8). Falls die geforderten Eigenschaften durch keine Tupel erfüllt werden, wird eine leere Resultattabelle zurückgegeben (vgl. Meier und Kaufmann 2016, S. 7). Neben den Möglichkeiten, die Daten zu verwalten, besitzt SQL auch Funktionen zur Datenwiederherstellung oder zum Datenschutz (vgl. Meier und Kaufmann 2016, S. 9).

Nach Schicker (vgl. 2017, S. 12) bestehen relationale Datenbanken aus nichts anderem als Tabellen, deren Zusammenhänge über Beziehungen hergestellt werden, die in den Tabellen selbst abgespeichert sind. Der logische Datenbankaufbau kann zu jeder Zeit geändert werden, da sich Tabellen unkompliziert hinzufügen oder löschen lassen. Allerdings müssen für Zugriffe oft Daten aus mehreren Tabellen gelesen und zusammengefügt werden, wodurch die Dauer des Zugriffs verlängert wird und zahlreiche Eingaben benötigt und Ausgaben erstellt werden. Aus diesem Grund ist für den Betrieb relationaler Datenbanken eine hohe Rechnerleistung nötig. (vgl. Schicker 2017, S. 12) Das bei relationalen Datenbanken verwendete Transaktionskonzept ACID hat nicht nur den Vorteil, dass es die Daten konsistent hält, sondern hat auch einen Nachteil. Je größer die Datenmengen werden, desto schwieriger wird es, die Daten in einer relationalen Datenbank zu speichern. Wenn sie nicht mehr auf den Speicher einer Maschine passen, kann man sie auf mehrere Rechner verteilen, was allerdings mit einem großen Aufwand verbunden ist, da das Konzept der relationalen Datenbanken in der Zeit entwickelt wurde, in der ein Programm nur auf einem Rechner lief. Um die ACID-Eigenschaft weiterhin zu erfüllen, verlängert sich dadurch die Latenzzeit. (vgl. Fasel 2016, S. 110) Nach Fasel (vgl. 2016, S. 110) führt die

Datenspeicherung in Tabellen bei analytischen Fragestellungen zu aufwändigen Operationen. Der Programmiercode für diese Abfragen wird immer unübersichtlicher. Für den Anwender wäre ein imperativer Code einfacher zu lesen und zu formulieren, doch von SQL wird ein deklarativer Code benötigt. Ein imperativer Code beschreibt nicht das gewünschte Ergebnis, sondern die Schritte, die dafür benötigt werden. (vgl. Jurney 2017, S. 83) Darüber hinaus sind analytische Operationen spaltorientiert, die Speichermechanismen von klassischen Datenbanken jedoch reihenorientiert. Darum ist der Aufwand der Bearbeitung solcher Operationen höher. (vgl. Fasel 2016, S. 110) Aus diesen Gründen wurden Datenbanken entwickelt, die nicht mehr ACID-konform sind und oft keine SQL-Zugriffsstrukturen besitzen. Die Bezeichnung für diesen neuen Typ lautet NoSQL-Datenbank. (vgl. Fasel 2016, S. 111)

3.3 Big Data und ihre Herausforderungen

Durch den exponentiellen Anstieg der Datenbestände seit den späten 90er-Jahren und der Annahme, dass sie weiterhin ansteigen werden, hat sich der Begriff Big Data gebildet (vgl. McCreary und Kelly 2014, S. 128). Big Data beschreibt umfangreiche Datenbestände, die mit den „herkömmlichen Softwarewerkzeugen kaum mehr zu bewältigen sind“ (Meier und Kaufmann 2016, S. 11). Diese Daten weisen oft keine Struktur aus und kommen aus verschiedenen Quellen (vgl. Meier 2018, S. 5). Nach McCreary und Kelly (vgl. 2014, S. 128) sind sie so umfangreich, dass ein einzelner Prozessor Schwierigkeiten bei der Datenverwaltung bekommt und deshalb liegt eine Lösung von Big Data-Probleme in verteilten Systemen nahe. Die Nutzung von Big Data bring zahlreiche Vorteile mit sich. Nach Huber (vgl. 2018, S. 22) kann die Verarbeitung von größeren Datenmengen für die Entscheidungsfindung ausgenutzt werden, da mehr Erfahrungen über das Unternehmen gewonnen werden. Aus der Datenanalyse können beispielsweise „Markttrends, Kundenvorlieben, Nutzergewohnheiten und Kundenbedarfe“ (Huber 2018, S. 25) rechtzeitig aufgedeckt werden. Außerdem kann eine hohe Produktivitätssteigerung des Werkes erreicht werden, da die Daten zur Optimierung der Fertigungsprozesse dienen (vgl. Bischoff 2017, S. 32). In diesem Feld und der Entwicklung neuer Geschäftsmodelle, die mit den Daten entstehen, nimmt die Automobilindustrie eine Vorreiter-Rolle ein (vgl. Huber 2018, S. 25). Um möglichst viele Daten zu generieren, muss die komplette IT-Landschaft datentechnisch vernetzt werden. Die Vernetzung soll alle Ebenen eines Unternehmens Daten generieren lassen, aus denen anschließend Ableitungen und Querbezüge hergestellt werden. (vgl. Huber 2018, S. 26)

Es ist nicht einfach, für den Begriff Big Data eine einheitliche Definition zu finden. Oft werden für die Charakterisierung die vier V’s Volume, Variety, Velocity und Veracity herangezogen (vgl. Saake et al 2018, S. 662). Nach Meier und Kaufmann (vgl. 2016, S. 13) fügen einige Experten diesen Eigenschaften ein weiteres V, nämlich Value, hinzu. Im Folgenden werden diese Charakteristika erklärt. Mit Volume wird der umfangreiche Datenbestand im Teta- bis Zettabytebereich bezeichnet (vgl. Fasel und Meier 2016, S. 6). Laut Tiwari (vgl. 2011, S. 8) ist es allerdings schwierig, die genaue Größe der Datenmengen in Zahlen zu fassen. Variety bezieht sich auf die unterschiedlichen Datenformate und -modelle unter

die beispielsweise „Text, Bild, Video und Audio“ (Saake et al. 2018, S. 662) fallen. Dabei lassen sich die Daten nach King (vgl. 2014, S. 35) in strukturierte, semistrukturierte und unstrukturierte Daten differenzieren. Die strukturierten Daten haben die gleiche Form und werden in herkömmlichen Datenbanken hinterlegt. Bei unstrukturierten Daten, die keine einheitliche Form aufweisen, bedarf es für deren Analyse gänzlich neue Technologien. Einen Mix aus diesen beiden Arten stellen die semistrukturierten Elemente dar. Sie sind teilweise strukturiert, enthalten aber bereits unstrukturierte Daten, weshalb für dessen Verarbeitung komplexere Verfahren erfordert werden. (vgl. King 2014, S. 35) Der Begriff Velocity nimmt Bezug auf die hohe Geschwindigkeit der Datenauswertung, bis hin zur Echtzeitverarbeitung (vgl. Meier und Kaufmann 2016, S. 13). Nach King (vgl. 2014, S. 35) beinhaltet dieser Begriff ebenfalls die Geschwindigkeit, mit der die Daten anfallen. Der Ausdruck Veracity greift die unterschiedliche Qualität der heterogenen Daten auf. Sie können beispielsweise ungenau, mehrdeutig, oder auch unvollständig sein. (vgl. Saake et al. 2018, S. 662) Die Qualitätsunterschiede haben zur Folge, dass enorme Datenmengen nicht unbedingt eine bessere Auswertungsqualität gewährleisten. Um die Aussagekraft der Daten zu bestimmen, werden spezifische Algorithmen benötigt. (vgl. Meier 2018, S. 7) Die von einigen Experten ergänzte Eigenschaft Value weist auf den Mehrwert für Unternehmen hin, der durch die Verwendung von Big Data Anwendungen geschaffen werden kann (vgl. Meier und Kaufmann 2016, S. 13).

Nach Saake et al. (vgl. 2018, S. 662) sind herkömmliche Datenbanksysteme nicht für den Einsatz von Big Data ausgelegt. Aus diesem Grund wird die Bearbeitung solcher Daten auf einem Cluster aus zahlreichen, parallel arbeitenden Rechnern durchgeführt und es ist zu überprüfen, ob die Datenintegrität durch das strikte ACID-Prinzip erfüllt werden muss, oder ob stattdessen ein schwächeres Konsistenzmodell verwendet werden kann (vgl. Saake et al. 2018, S. 662 f.). McCreary und Kelly (vgl. 2014, S. 128) beschreiben, dass sich NoSQL-Lösungen für die Bearbeitung von Big Data-Problemen eignen.

Um einen Überblick über Big Data-Probleme zu bekommen, lassen sie sich nach McCreary und Kelly (vgl. 2014, S. 135) in zwei Arten unterteilen. Bei diesen Arten werden die Daten entweder überwiegend angefragt und gelesen (read-mostly) oder zusätzlich durch Schreibzugriffe verändert (read-write). Jeder Problemtyp erfordert zur Bewältigung eine individuelle Kombination von NoSQL-Systemen. Zu den häufigsten Typen zählen Probleme, bei denen die Daten größtenteils gelesen und nur selten verändert werden. Dies ist etwa beim Umgang mit Ereignisprotokollen zu Vorgängen im Unternehmen der Fall oder wenn Dokumente anfallen, die aus Volltext bestehen. (vgl. McCreary und Kelly 2014, S. 135 f.)

3.4 NoSQL-Datenbanken

Dieser Abschnitt beschäftigt sich mit den allgemeinen Grundlagen der NoSQL-Datenbanken (Abschnitt 3.4.1) und geht danach näher auf die vier NoSQL-Core-Modelle ein (Abschnitt 3.4.2 bis 3.4.5). Auf Basis dieser Ausarbeitung werden die Kriterien identifiziert und mit einem Erfüllungsgrad versehen.

3.4.1 Definition

Nichtrelationale Datenbanktechnologien haben mit der Entwicklung des Webs in den 2000er-Jahren immer mehr an Bedeutung gewonnen. Es wurden Datenhaltungssysteme benötigt, die mit den großen Datenmengen der Web-Anwendungen umgehen konnten. Zwar lagen die Stärken von den damals überwiegend verwendeten SQL-Datenbanken in den Bereichen Konsistenz und Datensicherheit, doch durch die steigende Rechenleistung, die für die Überprüfung dieser Faktoren bei großen Datenmengen nötig ist, stießen relationale Datenbanken schneller an ihre Grenzen. (vgl. Meier und Kaufmann 2016, S. 221) Es existiert der Konflikt zwischen der Konsistenzerhaltung und der effizienten Verarbeitung von enormen Datenmengen (vgl. Meier und Kaufmann 2016, S. 222). Deshalb forcierte die „Open Source- und Web Development-Community“ (Meier und Kaufmann 2016, S. 222) die Entwicklung von massiv verteilten Datenbanksystemen. Bei diesen Datenbanksystemen wird nicht auf Atomarität und Konsistenz gesetzt, sondern es wird das Problem der Skalierbarkeit und Verfügbarkeit angegangen (vgl. Vaish 2013, S. 9). Nach Sadalage und Fowler (vgl. 2013, S. 12) gehört die Eigenschaft eines offenen Quelltextes (open-source) zu den Charakteristika von NoSQL-Datenbanken. McCreary und Kelly (vgl. 2014, S. 5) merken an, dass das Konzept von NoSQL auch von kommerziellen Produkten genutzt wird und nicht alle NoSQL-Systeme ein Open Source-Modell besitzen.

Der Begriff NoSQL ist laut Tiwari (vgl. 2011, S. 4) ein Akronym für „Not Only SQL“ und beschreibt solche Datenbanktechnologien, die nicht den Grundsätzen von relationalen Datenbankmanagementsystemen entsprechen. Anders ausgedrückt werden die Daten nicht in Tabellen gespeichert und als Datenbanksprache wird nicht SQL verwendet (vgl. Meier und Kaufmann 2016, S. 18). NoSQL ist weder eine Datenbank oder ein Datenbanktyp, sondern ein Ausdruck, mit dem sich bestimmte Datenbanken von anderen herausfiltern lassen (vgl. Vaish 2013, S. 9). Auf einige NoSQL-Datenbanktypen wird in Kapitel 3.4.2 bis 3.4.5 genauer eingegangen. Der Fokus der NoSQL-Datenbanken liegt auf Realtime-Anwendungen und nicht so sehr auf Analysen. Vaish (vgl. 2013, S. 20) beschreibt, dass keine komplexen Abfragen möglich sind. Sollen die Daten ebenfalls analysiert werden, können die relationalen Datenbanken mit den NoSQL-Datenbanken kombiniert werden. (vgl. Müller 2016, S. 150) Nach Sadalage und Fowler (vgl. 2013, S. 151) sind die beiden Hauptgründe für die Nutzung von NoSQL-Technologien der verbesserte Datenzugriff und die erhöhte Produktivität der Programmierer, da eine Datenbank genutzt wird, die den Bedürfnissen einer Anwendung eher entspricht. In Abbildung 3.4 wird die Grundstruktur eines NoSQL-Datenbanksystems veranschaulicht.

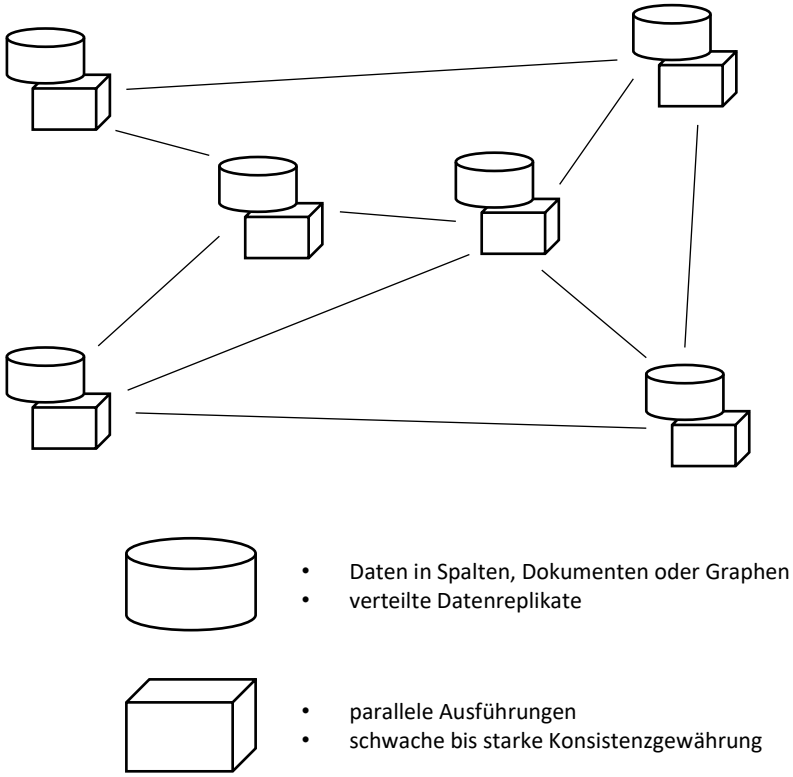

Abbildung 3.4: Grundstruktur eines NoSQL-Datenbanksystems (nach Meier und Kaufmann 2016, Abb. 1.10)

Eine große Veränderung zu den relationalen Datenbanken ist die massiv verteilte Datenhaltungsarchitektur des NoSQL-Datenbanksystems (vgl. Meier 2018, S. 11). Bei der Verarbeitung auf vielen Prozessoren wird eine sehr schnelle Performanz erreicht (vgl. McCreary und Kelly 2014, S. 5). Diese Beschleunigung der Rechnungsvorgänge resultiert aus dem Einsatz von parallelen Auswertungsverfahren, wie zum Beispiel MapReduce (vgl. Meier und Kaufmann 2016, S. 19). Nach McCreary und Kelly (vgl. 2014, S. 132) sind NoSQL-Datenbanken linear skalierbar, da jeder dem Netz hinzugefügter Rechner, bzw. Prozessor für eine Leistungserhöhung sorgt. Aufgrund der modularen Architektur können vorhandene Komponenten ausgetauscht werden (vgl. McCreary und Kelly 2014, S. 20). Des Weiteren können dem Netzwerk, bzw. Cluster weitere Server einfach hinzugefügt werden, was zu einer vergleichsweise kostengünstigen Abarbeitung von vielen Daten führt (vgl. Müller 2016, S. 149). Durch das Hinzufügen von neuen Datenbank-Servern wird außerdem die Kapazität erweitert. Die Daten werden automatisch von der Datenbank über die Server des Clusters verteilt. (vgl. Müller 2016, S. 150) NoSQL-Datenbanken sind auf eine verteilte, horizontale Skalierbarkeit ausgerichtet (vgl. Meier und Kaufmann 2016, S. 222). Es wird durch die Datenverteilung eine hohe Ausfallsicherheit erreicht, da mehrere Replikate der Daten vorhanden sind. Bei einem Ausfall eines Servers, kann problemlos auf einen anderen Server zugegriffen werden. (vgl. Müller 2016, S. 149 f.) Nach Sadalage und Fowler (vgl. 2013, S. 10) operieren NoSQL-Datenbanken nicht nach einem Schema, weshalb beliebige Daten hinzugefügt werden können, ohne die Struktur der Datenbank zu ändern. Bei relationalen Datenbankmanagementsystemen müsste das komplette Datenbankschema geändert werden, was eine Downtime zur Folge hätte (vgl. Müller

2016, S. 150). Jedoch müssen Daten in NoSQL-Datenbanken nicht in relationale Objekte umgewandelt werden, da sie nicht im Relationenmodell gespeichert werden (vgl. Fasel 2016, S. 111). Dadurch ist eine flexible Datenspeicherung möglich, da die Datenstrukturen im Vorfeld nicht „genau definiert werden müssen“ (Fasel 2016, S. 113). Allerdings ist es trotz der Schemaflexibilität für eine Applikation essenziell, die Daten sauber zu modellieren, um langfristig keine Leistungseinbüßen hinnehmen zu müssen (vgl. Fasel 2016, S. 113). Neben den Vorteilen der Flexibilität hat sie den Nachteil, dass dadurch die Datenabfrage und -analyse komplexer wird (vgl. Fasel 2016, S. 111). Angesichts der hohen Verfügbarkeit und Ausfalltoleranz von NoSQL-Datenbanken kann die Konsistenz der Daten nach dem CAP-Theorem nur verzögert gewährleistet werden (vgl. Meier 2018, S. 12). Das führt zum Einsatz eines schwächeren Konsistenzmodells, nämlich BASE (vgl. Meier und Kaufmann 2016, S. 222). Bei diesem Modell werden die Daten auf den einzelnen Servern in einem Cluster zeitlich verzögert aktualisiert. Das führt dazu, dass auf den Servern in der Zwischenzeit verschiedene Versionen der Datenbestände abgespeichert sind. (vgl. Meier und Kaufmann 2016, S. 148) Weitere Vorteile von NoSQL-Datenbanken sind der einfache Zugriff über eine Programmschnittstelle (vgl. Meier und Kaufmann 2016, S. 222) und die reduzierte Entwicklungszeit, da keine komplexen SQL-Abfragen geschrieben werden müssen (vgl. Vaish 2013, S. 11). Allerdings kommen NoSQL-Systeme auch nicht an die Abfragemöglichkeiten von SQL heran, da sie nicht viele Abfragesprachen besitzen. Die Tatsache, dass Daten in NoSQL-Datenbanken nach dem Bedarf der jeweiligen Applikation abgespeichert werden, verhindert die Existenz einer komplexen und mächtigen Abfragesprache. (vgl. Müller 2016, S. 151)

Diese Arbeit beschränkt sich auf die sogenannten vier Core-NoSQL-Modelle, obwohl noch weitere Arten von NoSQL-Datenbanken existieren, die unter dem Begriff Soft NoSQL zusammengefasst werden (vgl. Meier und Kaufmann 2016, S. 222):

- Schlüssel-Wert-Datenbanken (key-value stores) speichern Daten sehr simpel ab und nutzen einen Schlüssel, um auf Datenwerte zuzugreifen (vgl. McCreary und Kelly 2014, S. 6).
- Spaltenfamilien-Datenbanken (column family stores) speichern Daten im Gegensatz zu relationalen Datenbanken in Spalten statt in Reihen ab. Um auf Daten zuzugreifen, werden Reihen- und Spaltenschlüssel genutzt. (vgl. McCreary und Kelly 2014, S. 81)
- Dokument-Datenbanken (document stores) können hierarchische Daten abspeichern (vgl. McCreary und Kelly 2014, S. 6), ohne sie aufzusplitten (vgl. Tiwari 2011, S. 18). Auf diese Daten kann ebenfalls mit einem Schlüssel zugegriffen werden (vgl. McCreary und Kelly 2014, S. 86).
- Graphdatenbanken (graph stores) werden genutzt, um Entitäten und deren Beziehungen abzuspeichern (vgl. Sadalage und Fowler 2013, S. 111).

Durch die Benutzung einer einzigen Datenbanktechnologie lassen sich nur im Ausnahmefall alle Anforderungen eines Geschäfts abdecken. Das Verständnis der verschiedenen Technologien ist demnach sehr wichtig. (vgl. Fasel 2016, S. 113) Eine logische Schlussfolgerung ist der Einsatz von verschiedenen Datenbanktypen für unterschiedliche Umstände (vgl. Sadalage und Fowler 2013, S. 11). Der Begriff Polyglot Persistence (wörtlich: mehrsprachige Persistenz) beschreibt die Kombination verschiedener Datenbankmodelle, um ihre jeweiligen Stärken zu vereinen (vgl. Meier und Kaufmann 2016, S. 222). Nach der Einschätzung von Sadalage und Fowler (vgl. 2013, S. 11) werden die relationalen Datenbanken nicht verschwinden, da sich noch immer die häufigsten Datenbanken sind und durch ihre Eigenschaften wie Vertrautheit oder Funktionsumfang für die meisten Projekte sehr wichtig sind. Die Charakteristika eines relationalen Datenbanksystems werden von NoSQL-Datenbanksystemen nur teilweise erfüllt, weshalb man nicht auf die relationalen Datenbanksysteme verzichten kann. (vgl. Meier und Kaufmann 2016, S. 11) Im Folgenden werden einige Begrifflichkeiten und Verfahren definiert, die in NoSQL-Technologien ihre Anwendung finden.

Datenbankcluster

Ein Cluster besteht aus einer Anzahl von Trägern (rack), die sich aus mehreren gewöhnlichen, handelsüblichen (commodity) Computern zusammensetzen. Ein einzelner Computer wird als Knoten bezeichnet und beinhaltet einen Prozessor (CPU), seinen eigenen Arbeitsspeicher (RAM) und eine Festplatte (disk). Die Träger haben untereinander eine hohe Datenübertragungsrate. (vgl. McCreary und Kelly 2014, S. 20 f.)

Architekturen von verteilten Systemen zum Bewältigen von Big Data Problemen

Nach McCreary und Kelly (vgl. 2014, S. 136) gibt es drei Möglichkeiten, Arbeitsspeicher, Prozessoren und Festplatten zwischen Computersystemen aufzuteilen. Beim geteilten Arbeitsspeicher (shared RAM) greifen viele Prozessoren auf einen Arbeitsspeicher zu. Wenn jeder Prozessor seinen eigenen Arbeitsspeicher besitzt, doch auf ein gemeinsames Speichernetzwerk zugreift, wird das als geteilter Speicher (shared disk) bezeichnet. Bei der letzten Alternative wird keine Ressource geteilt (shared-nothing). Diese Methode setzt wird günstige, handelsübliche Hardware ein und wird bei Big Data-Problemen eingesetzt. (vgl. McCreary und Kelly, S. 136) Sie entspricht der Grundstruktur eines NoSQL-Datenbanksystems (Abbildung 3.4). Von den Kerntechnologien lassen sich nach McCreary und Kelly (vgl. 2014, S. 137) bevorzugt Schlüssel-Wert-Datenbanken und Dokument-Datenbanken auf mehreren verteilten Speichern ablegen (cache-friendly), wohingegen Spaltenfamilien-Datenbanken und Graphdatenbanken nicht cache-friendly sind. Anfragen an einen Graphen lassen sich am effektivsten bearbeiten, wenn der komplette Graph im Arbeitsspeicher gehalten werden kann. Für diesen NoSQL-Datenbanktypen resultiert demnach der Einsatz einer shared RAM-Architektur. (vgl. McCreary und Kelly 2014, S. 137)

Horizontale und vertikale Skalierung

Um die gestiegenen Datenmengen zu bewältigen, bedarf es leistungsstärkerer Ressourcen zur Datenverarbeitung. Dabei gibt es die Möglichkeiten vertikal (scale up) oder horizontal (scale out) zu skalieren. (vgl. Sadalage und Fowler 2013, S. 8) Bei der vertikalen Skalierung wird die Leistungsfähigkeit des Servers, auf dem die Datenbank betrieben wird, erhöht (vgl. Sadalage und Fowler 2013, S. 37). Nach McCreary und Kelly (vgl. 2014, S. 7) war die Anschaffung schnellerer Prozessoren irgendwann keine Alternative mehr, da die Chipdichte der verwendeten Teile so hoch wurde, dass deren erzeugte Wärme nicht mehr entweichen konnte, was zu einer Überhitzung der Chips führte. Aus diesem Grund wurde der Fokus von der Leistungssteigerung der Chips auf das effiziente Zusammenwirken mehrerer Prozessoren gelegt (vgl. McCreary und Kelly 2014, S. 7). Die Methode, bei der die Datenbank auf einem Cluster von mehreren Servern betrieben wird, nennt sich horizontale Skalierung (vgl. Sadalage und Fowler 2013, S. 37). Laut Fasel und Meier (vgl. 2016, S. 7) eignen sich für die Maschinen im Cluster auch ältere und billigere Komponenten, da die Leistungsfähigkeit über die Menge und nicht die Leistung der Maschinen verbessert wird. Deshalb ist das Verfahren der horizontalen Skalierung preisgünstiger als das der vertikalen Skalierung. Ein weiterer Vorteil ist die Ausfallsicherheit, die aus der Fähigkeit des Clusters, Ausfälle von einzelnen Maschinen zu kompensieren, resultiert. (vgl. Sadalage und Fowler 2013, S. 8) Nach Tiwari (vgl. 2011, S. 9) bestehen einige Cluster aus hunderttausenden Servern. Die horizontale Skalierung hat nicht nur positive Seiten, sondern auch einige Schwächen. Dazu gehören die anfallenden Transportkosten und Latenzzeiten, die begrenzte Datenübertragungsrate und die Eigenschaften, dass das Netz unzuverlässig und unsicher sein kann. (vgl. Tiwari 2011, S. 175) Nach Sadalage und Fowler (vgl. 2013, S. 37) ist die bevorzugte Variante die Datenhaltung auf einem Server, da dabei nicht die Komplexitäten der horizontalen Skalierung auftreten.

Distributionsmodelle

Es existieren zwei unterschiedliche Modelle, die Daten auf das Cluster zu verteilen. Diese Modelle sind zum einen das Sharding, bei dem unterschiedliche Teile des Datenbestandes auf mehreren Knoten verteilt werden, und zum anderen die Replikation, bei dem die gleichen Daten auf mehreren Knoten verteilt werden. Das Verfahren der Replikation wird zudem in die Master-Slave Replikation und die Peer-to-Peer Replikation unterschieden. (vgl. Sadalage und Fowler 2013, S. 37) Die Modelle Master-Slave und Peer-to-Peer vergeben unterschiedliche Zuständigkeiten für die verarbeiteten Daten bei einer Anfrage (vgl. McCreary und Kelly 2014, S. 137). Nach McCreary und Kelly (vgl. 2014, S. 28) wird die Datenbank beim Sharding in mehrere Datenblöcke aufgeteilt (shards), welche danach auf das Cluster verteilt werden. Dadurch teilen sich die Abfragen im Idealfall gleichmäßig auf die Knoten auf, wodurch eine geringere Latenzzeit erreicht wird. Um diesem Idealfall möglichst nahe zu kommen müssen die Daten, welche zusammen abgefragt werden, auch auf dem gleichen Knoten abgespeichert werden. Dadurch muss bei einer Abfrage nur auf einen Knoten zugegriffen werden. Außerdem ist es wichtig,

die Belastung gleich zu halten. Es sollten möglichst keine Knoten existieren, auf die öfter zugegriffen wird als auf andere. (vgl. Sadalage und Fowler 2013, S. 38 f.) NoSQL-Systeme besitzen die Funktion, die Daten automatisch auf die Knoten zu verteilen (vgl. McCreary und Kelly 2014, S. 28). Durch dieses sogenannte Autosharding sinken die Betriebskosten (vgl. McCreary und Kelly 2014, S. 20). Der größte Vorteil des Shardings liegt in der Verbesserung der Lese- (read performance) und Schreibleistung (write performance). Allerdings steigt durch dieses Modell nicht die Ausfalltoleranz des Netzes. Bei einem Ausfall eines Knotens werden zwar nicht die Daten auf den anderen Knoten beeinflusst, doch ein Benutzer kann nicht mehr auf die Daten des defekten Knotens zugreifen. Wenn Sharding das einzig eingesetzte Verteilungsmodell ist, sinkt die Ausfalltoleranz, da Ausfälle bei den handelsüblichen Maschinen in Clustern nichts Ungewöhnliches sind. (vgl. Sadalage und Fowler 2013, S. 39 f.) Im Folgenden werden die beiden Replikationsmodelle Master-Slave und Peer-to-Peer erläutert. Das Master-Slave Modell besitzt einen Knoten, der die Rolle des Masters einnimmt und die Schreibzugriffe auf die Daten ausführt. Sobald der Schreibzugriff beendet ist, synchronisiert sich der Master mit den Slaves, welche die übrigen Knoten des Clusters darstellen. (vgl. Wiese 2015, S. 262) Dieses Modell wird bei Datensätzen eingesetzt, die sehr oft gelesen werden, denn Lesezugriffe können von beiden Knotensorten bearbeitet werden. Allerdings ist es sinnvoll, die Leseanfragen auf die Slaves umzuleiten. Die Leseperformanz kann durch Hinzufügen von weiteren Slaves verbessert werden. Sollte der Masterknoten ausfallen, können eingehende Leseanfragen weiterhin von den Slaves verarbeitet werden. (vgl. Sadalage und Fowler 2013, S. 40) Nach Wiese (vgl. 2015, S. 262) wird nach dem Ausfall des Masters ein Slave als sein Nachfolger bestimmt, an den alle zukünftigen Schreibzugriffe geleitet werden. Die Schreibperformanz dieses Modells wird allerdings durch den Master limitiert, da dieser alle Schreibzugriffe bearbeiten und sich mit den Slaves synchronisieren muss. Deshalb ist der Einsatz bei Datensätzen, die schreibintensive Zugriffe erhalten, nicht geeignet. (vgl. Sadalage und Fowler 2013, S. 40) Es besteht die Möglichkeit, dass verschiedene Nutzer von verschiedenen Knoten unterschiedliche Daten lesen, da die Synchronisation der Slaves noch nicht abgeschlossen war. Master-Slave Modelle können somit eventuell inkonsistente Daten halten. (vgl. Sadalage und Fowler 2013, S. 42) Das Problem der Schreibzugriffen löst die Peer-to-Peer Replikation. Die Knoten dieses Typen sind gleichbedeutend, sodass jeder von ihnen Schreibanfragen bearbeiten kann. Das macht Schreib- und Leseanfragen in diesem Modell zu einem hohen Grad skalierbar. (vgl. Tiwari 2011, S. 264) Nach McCreary und Kelly (vgl. 2014, S. 138) resultiert daraus ein höherer Aufwand der Kommunikation und Komplexität, da sich die Knoten laufend synchronisieren müssen. Dieses Modell ist tolerant gegen Ausfälle, da durch den Verlust eines Knotens keine Daten verloren gehen. Um die Performanz zu erhöhen, lassen sich neue Knoten ohne großen Aufwand hinzufügen. (vgl. Sadalage und Fowler 2013, S. 43) Das Peer-to-Peer Modell besitzt das gleiche Konsistenzproblem wie das Master-Slave Modell. Es kann passieren, dass Benutzer veraltete Daten lesen, da diese noch nicht auf dem neuesten Stand sind. Darüber hinaus kann ein Konflikt entstehen, wenn

unterschiedliche Anwender auf jeweils unterschiedlichen Knoten die Daten durch einen Schreibzugriff überschrieben haben, ohne dass die Daten zwischen den Schreibzugriffen synchronisiert wurden. (vgl. Wiese 2015, S. 263) Nach Sadalage und Fowler (vgl. 2013, S. 43) bleibt ein sogenannter write-write Konflikt im Gegensatz zur vorübergehenden Inkonsistenz beim Lesen permanent. Ein Datenbanksystem kann entweder eins der Verteilungsmodelle benutzen oder eine Kombination aus Sharding und Replikation (vgl. Sadalage und Fowler 2013, S. 45). Bei der Verwendung von Sharding und der Master-Slave Replikation wird jeder shard auf mehrere Knoten repliziert, wovon einer zum Master bestimmt wird und die restlichen die Rolle des Slaves einnehmen (vgl. Wiese 2015, S. 262). Nach Sadalage und Fowler (vgl. 2013, S. 43) wird bei Spaltenfamilien-Datenbanken häufig das Sharding mit der Peer-to-Peer Replikation kombiniert.

MapReduce

Zur Verarbeitung von großen Datensätzen haben Dean und Ghemawat (vgl. 2004, S. 137) ein Programmiermodell entwickelt, mit dem sich Berechnungen von zahlreichen Maschinen parallelisieren lassen. Dafür nutzt es die Tatsache aus, dass die Daten auf verschiedene Systeme verteilt sind. Zu Beginn werden in der Map-Phase lokale Teilresultate berechnet, die danach in der Reduce-Phase zu einem globalen Resultat zusammengefügt werden. (vgl. Fasel 2016, S. 125) Nach Meier und Kaufmann (vgl. 2016, S. 180) wird das Verfahren bei NoSQL-Datenbanken neben Abfragen von Datenbankeinträgen auch für Lastverteilungen, Datentransfer oder Kategorisierungen verwendet. Das Vorgehen bei diesem Modell soll durch das einfache Beispiel in Abbildung 3.5 verdeutlicht werden, bei dem zwei verteilte Datenspeicher auf die Häufigkeit verschiedener Begriffe durchsucht werden. In der Map-Phase durchsuchen die beiden Mapfunktionen M1 und M2 die Einträge auf den jeweiligen Datenspeichern und geben Schlüssel-Wert-Paare aus, wobei der Schlüssel für den Begriff steht und der Wert für dessen Häufigkeit in dem Datenspeicher. Diese Paare werden mit einem Algorithmus alphabetisch sortiert und als Zwischenergebnisse ausgegeben. Während der Reduce-Phase werden die Zwischenergebnisse mit dem gleichen Schlüssel zusammengefasst und ihre Häufigkeiten addiert. Für das Endergebnis werden die beiden Listen in einem letzten Schritt vereint und nach Häufigkeiten sortiert. (vgl. Meier und Kaufmann 2016, S. 179)

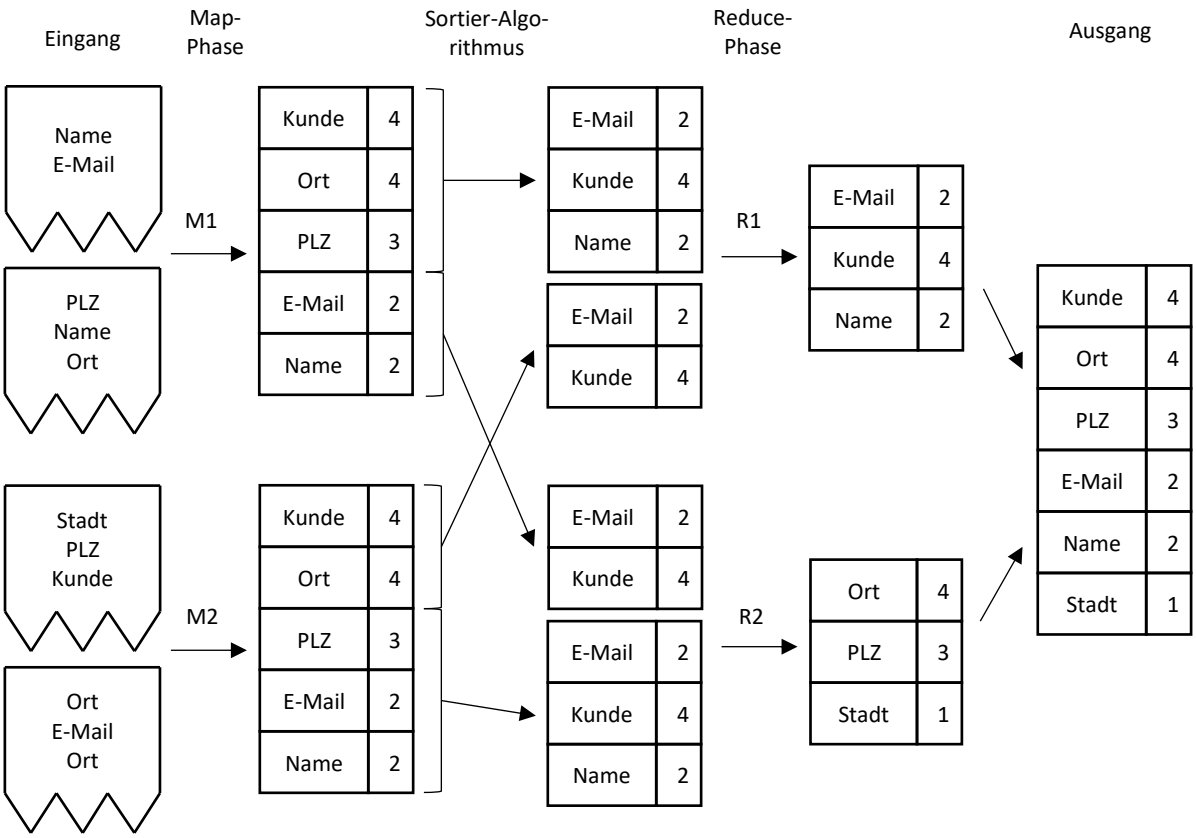

Abbildung 3.5: Beispiel zum MapReduce-Verfahren (nach Meier und Kaufmann 2016, Abb. 5.11 und McCreary und Kelly 2014, Abb. 6.10)

CAP-Theorem

Das von Eric Brewer (vgl. 2000, S. 4) im Jahre 2000 vorgestellte CAP-Theorem besagt, dass sich in verteilten Datensystemen bestenfalls zwei der drei Eigenschaften Konsistenz (Consistency, C), Verfügbarkeit (Availability, A) und Ausfalltoleranz (Partition Tolerance, P) erreichen lassen. Tiwari (vgl. 2011, S. 174) beschreibt die drei Eigenschaften folgendermaßen:

- Konsistenz: Der Zustand, in dem simultane Transaktionen die gleiche Version der Daten lesen oder verändern, es also keine veralteten Daten gibt (vgl. Tiwari 2011, S. 174).
- Verfügbarkeit: Damit ist die ununterbrochene Bereitschaft eines Systems gemeint, sodass es zu jedem Zeitpunkt Abfragen bedienen kann. Ist dies nicht möglich, da es beispielsweise beschäftigt ist, oder bei einem Zugriff nicht reagiert, besitzt das System auch keine Verfügbarkeit. (vgl. Tiwari 2011, S. 174)
- Ausfalltoleranz: Diese Eigenschaft ist gegeben, wenn das System seine Funktionsfähigkeit trotz eines Ausfalls von einigen Knoten im Cluster aufrechterhalten kann (vgl. Tiwari 2011, S. 175). Des Weiteren lassen sich Knoten einfügen oder wegnehmen, ohne dass der Betrieb unterbrochen wird (vgl. Meier und Kaufmann 2016, S. 148).

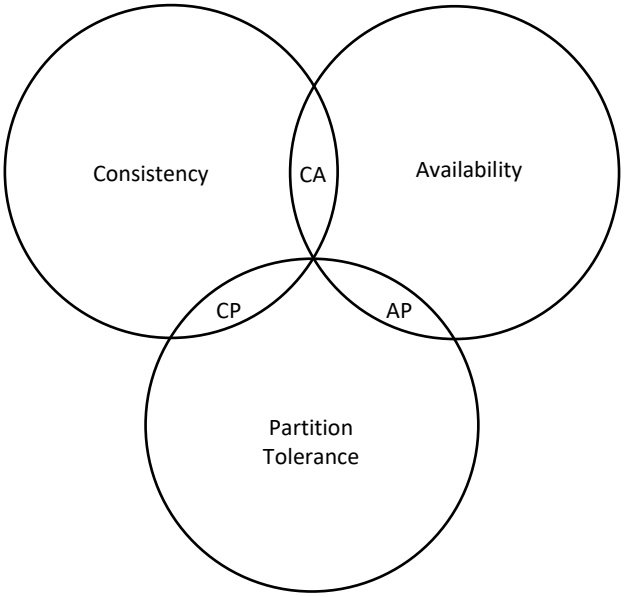

Abbildung 3.6: CAP Theorem (nach Schicker 2017, Abb. 9.1)

Das CAP-Theorem wird in Abbildung 3.6 durch ein Sinnbild dargestellt. Jeder Kreis steht für eine der drei Eigenschaften. Nach Tiwari (vgl. 2011, S. 178) müssen die Erfüllung einer Eigenschaft vernachlässigt werden, um die beiden anderen möglichst hoch zu halten. Aus diesem Kompromiss können drei verschiedene Kombinationen (CA, CP, AP) entstehen, die jeweils unterschiedliche Anwendungsfelder haben (vgl. Meier 2016, S. 34). Die Kombinationen sind in der Abbildung als Schnittmenge der Kreise dargestellt, wobei die Schnittmenge aus allen Kreisen nicht existieren kann, da dies gegen das CAP-Theorem verstoßen würde. Meier und Kaufmann (vgl. 2016, S. 149) haben Beispiele für die Anwendung der Kombinationen beschrieben:

- Kombination aus Konsistenz und Verfügbarkeit (CA): Findet Einsatz an einem Börsenplatz und wird durch relationale Datenbanksysteme sichergestellt, die ACID unterstützen (vgl. Meier und Kaufmann 2016, S. 149).
- Kombination aus Konsistenz und Ausfalltoleranz (CP): Wird in einem Netz von Geldautomaten verwendet, da es ausfalltolerant sein soll und Konsistenz gefordert wird. Für diese Kombination kann mit verteilten und replizierten rationalen oder NoSQL-Systemen gearbeitet werden (vgl. Meier und Kaufmann 2016, S. 149).
- Kombination aus Verfügbarkeit und Ausfalltoleranz (AP): Es werden NoSQL-Systeme gefordert, da „Verfügbarkeit und Ausfalltoleranz durch ein relationales Datenbanksystem nicht zu haben sind“ (Meier und Kaufmann 2016, S.149). Ein Beispiel wäre der Einkaufswagen in einem Onlineshop, dem Bestellungen hinzugefügt werden können, obwohl die Verfügbarkeit des Artikels nicht unbedingt im Inventar überprüft wurde (vgl. Tiwari 2011, S. 180).

Konsistenzsicherung

Die in Kapitel 3.1 vorgestellten Eigenschaften des ACID-Prinzip sind in einem Datenbanksystem zwar wünschenswert, stehen allerdings im Konflikt mit ihrer Verfügbarkeit und Performanz (vgl. Vaish 2013, S. 10). Soll dafür die Verfügbarkeit und Ausfalltoleranz im Vordergrund stehen, ist diese Konsistenzforderung nach Meier und Kaufmann (vgl. 2016, S. 148) nicht in jedem Fall anzustreben. Aus diesem Grund wird bei verteilten Systemen die weichere Konsistenzforderung BASE eingesetzt (vgl. Meier und Kaufmann 2016, S. 148). Nach Brewer (vgl. 2000, S. 4) setzt sich das Akronym BASE aus den Wörtern basically available, soft state und eventually consistent zusammen. Mit der grundsätzlichen Verfügbarkeit (basically available) ist gemeint, dass Daten zwar verfügbar sind, jedoch aufgrund von vorübergehender Inkonsistenz fehlerhaft sein können. Allerdings erreicht das System, wenn keine Eingaben mehr gemacht werden, schlussendlich einen konsistenten Zustand (eventual consistent). Die Zeitspanne, in der das System inkonsistent ist, wird als akzeptabel angenommen. Da sich das System durch die Eigenschaft der schlussendlichen Konsistenz mit der Zeit auch ohne Eingaben ändern kann, befindet es sich in einem lockeren Zustand (soft state). (vgl. Celko 2014, S. 11) Nach McCreary und Kelly (vgl. 2014, S. 27) sind Systeme, in denen das BASE-Prinzip verwendet wird, optimistisch. Entstandene Konflikte werden bei diesem Ansatz erkannt und behoben. Im Gegensatz dazu kann beim pessimistischen Ansatz nur ein Anwender auf das System zugreifen. Für die anderen Anwender wird es für die Dauer des Zugriffes gesperrt. Somit können keine Konflikte entstehen. (vgl. Sadalage und Fowler 2013, S. 48) Diese pessimistische Herangehensweise wird meistens bei relationalen Datenbanken verwendet (vgl. Meier und Kaufmann 2016, S. 155). Angesichts der Tatsache, dass bei Systemen nach dem BASE-Prinzip kein Code für die Sperrung und Entsperrung von Ressourcen geschrieben werden muss, sind diese Systeme schneller und simpler (vgl. McCreary und Kelly 2014, S. 27). Allerdings benötigen diese Systeme laut Celko (vgl. 2014, S. 11) einen globalen Zeitstempel, damit die Knoten bei der Synchronisierung die aktuellsten Datenversionen erkennen.

3.4.2 Schlüssel-Wert-Datenbanken

Die Schlüssel-Wert-Datenbanken (Key/Value Store) sind die simpelsten Typen von NoSQL-Datenbanken (vgl. Fasel 2016, S. 113). Sie bestehen aus Paaren, die sich jeweils aus einem Schlüssel (Key) und einem Wert (Value) zusammensetzen (vgl. Celko 2014, S. 81). Dabei darf jeder Schlüssel nur einmal vorkommen. Er ist somit eindeutig. (vgl. Tiwari 2011, S. 14) Normalerweise nehmen die Schlüssel und Werte nicht die Form von komplexen Datentypen an (vgl. Fasel 2016, S. 113), allerdings werden die abzuspeichernden Datenobjekte nicht in ihrem Typ begrenzt (vgl. Meier und Kaufmann 2016, S. 223).

Eine Schlüssel-Wert-Datenbank lässt sich mit einem Lexikon vergleichen, indem die Einträge die Schlüssel darstellen und die Definitionen der Einträge die Werte. Ist eine bestimmte Definition gesucht, muss nicht das komplette Lexikon gelesen werden, sondern es wird nach dem Eintrag gesucht, der zur

gesuchten Definition führt. Dieses Vorgehen hat eine große Zeitersparnis zur Folge. (vgl. McCreary und Kelly 2014, S. 64)

Schlüssel-Wert-Datenbanken haben lediglich drei Grundbefehle, die in Abbildung 3.7 verdeutlicht sind. Mit dem Befehl „PUT“ besteht die Möglichkeit, neue Paare in die Datenbank einzufügen oder – falls der Schlüssel bereits existiert – den Wert eines bestehenden Paares zu verändern. Der Befehl „GET“ fragt den Wert eines Paares ab und „DELETE“ löscht das jeweilige Paar. (vgl. McCreary und Kelly 2014, S. 68)

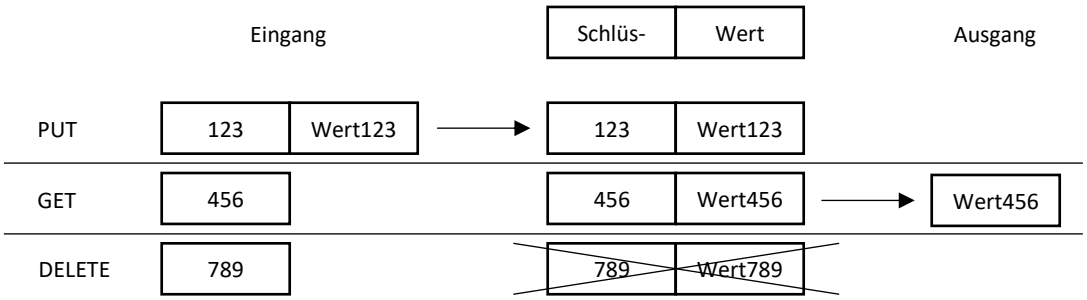

Abbildung 3.7: Die drei Grundbefehle PUT, GET und DELETE (nach McCreary und Kelly 2014, Abb. 4.5)

Zu den Eigenschaften der Schlüssel-Wert-Datenbanken gehört, dass die referentielle Integrität nicht überprüft wird (vgl. Meier und Kaufmann 2016, S. 224). Dies hat zur Folge, dass größere Datenmengen effizient verarbeitet werden können (vgl. Meier und Kaufmann 2016, S. 224). Um eine noch höhere Schreib- und Lesegeschwindigkeit der Daten zu erreichen, besteht die Möglichkeit, eine In-Memory-Datenbank zu verwenden (vgl. Meier und Kaufmann 2016, S. 224). Bei dieser Option wird nur der Arbeitsspeicher genutzt und es muss auf keine Festplatte zugegriffen werden (vgl. Sadalage und Fowler 2013, S. 144). Die Skalierung von Schlüssel-Wert-Datenbanken kann auf mehrere Weisen erhöht werden. Die am meisten verwendete Methode ist das Skalieren durch Sharding (vgl. Sadalage und Fowler 2013, S. 86), bei dem die vorhandenen Daten segmentiert und auf verschiedene Server verteilt werden (vgl. Sadalage und Fowler 2013, S. 38) und somit die Leistungsfähigkeit erhöht wird (vgl. Sadalage und Fowler 2013, S. 86). Ein Datensatz ist einfach aufzuteilen, da die Schlüssel-Wert-Datenbanken auf ein Schema und auf die Überprüfung der referentiellen Integrität verzichten (vgl. Meier und Kaufmann 2016, S. 223). Ein Nachteil dieser Datenbanktypen ist, dass durch den Verzicht auf ein Datenbankschema oft Unordnung in der Datenverwaltung herrscht (vgl. Meier und Kaufmann 2016, S. 225).

Aufgrund ihrer Eigenschaften sind Schlüssel-Wert-Datenbanken sehr einfache Datenbanken (vgl. Sadalage und Fowler 2013, S. 81). Sie eignen sich für die persistente Speicherung von Daten mit einfachem Strukturtyp, „welche als Schlüssel-Wert-Paar dargestellt werden können“ (Fasel 2016, S. 115). Ein anderer Anwendungsfall ist der eines volatilen Zwischenspeichers „für Webseiten und andere Applikationen“ (Fasel 2016, S. 115). Durch die hohe Schreib- und Lesegeschwindigkeit lässt sich durch diesen Datenbankentyp viel Zeit und Geld sparen (vgl. McCreary und Kelly 2014, S. 71). Die

Datenbanken stoßen an ihre Grenzen, wenn mit ihnen Daten verarbeitet werden sollen, die eine Beziehung untereinander haben oder wenn ein Schlüssel anhand des Inhalts seines Wertes ausgesucht werden soll. Es ist nicht möglich, den Wert ohne den zugehörigen Schlüssel zu untersuchen. (vgl. Sadalage und Fowler 2013, S. 87 f.)

3.4.3 Spaltenfamilien-Datenbanken

Die Typen der Spaltenfamilien-Datenbanken (Column Family Stores) ähneln den relationalen Datenbanken, jedoch weisen zu ihnen den Unterschied auf, dass die Daten nicht in relationalen Tabellen gespeichert werden, sondern in „erweiterten und strukturierten mehrdimensionalen Schlüsselräumen“ (Meier und Kaufmann 2016, S. 226). Diese Schlüsselräume bestehen aus Spaltenfamilien (column families) und einem Reihenschlüssel und ähneln relationalen Objekten (vgl. Fasel 2016, S. 118). Ihren Namen erhalten die Spaltenfamilien-Datenbanken durch die Verwendung von Spaltenfamilien (column families), welche jeweils aus einer Gruppe von ähnlichen Spalten bestehen, die oft zusammen abgefragt werden (vgl. Sadalage und Fowler 2013, S. 99). Aufgrund der Annahme, dass die Spalten zusammen gelesen werden, haben die hinterlegten Daten in einer Spaltenfamilie stets den gleichen Typ (vgl. Meier und Kaufmann 2016, S. 227) und werden „physisch möglichst am gleichen Ort gespeichert“, um die Performance zu verbessern (Meier und Kaufmann 2016, S. 228).

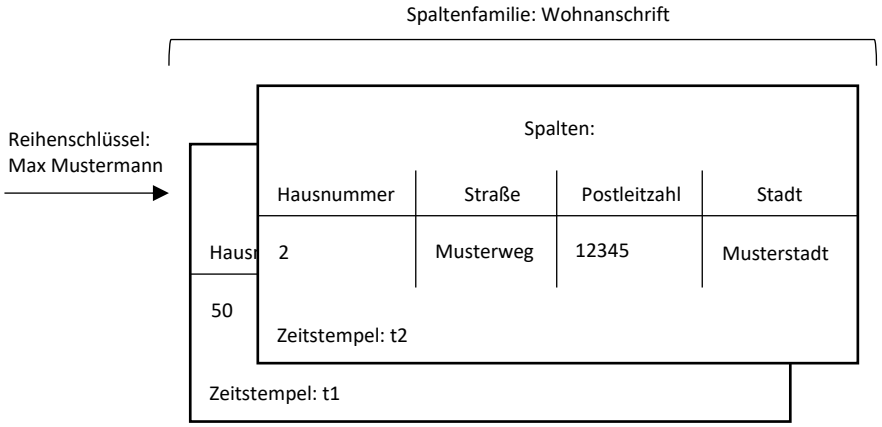

Abbildung 3.8: Spaltenfamilie Wohnanschrift (nach Meier und Kaufmann 2016, Abb. 7.2)

Ein Beispiel für eine Spaltenfamilie ist die Wohnanschrift in einem Adressbuch (Abbildung 3.8). Die Wohnadresse kann verschiedene Spalten enthalten, etwa für die Hausnummer, die Straße, die Postleitzahl oder die Stadt. Dabei ist es nicht wichtig, ob in jeder Spalte ein Wert eingetragen wird. Die Werte in diesen Spalten können als Schlüssel-Wert-Paar gesehen werden (vgl. Tiwari 2011, S. 11). Um die Eindeutigkeit des Schlüssels zu gewährleisten, setzt er sich aus mehreren Attributen zusammen, die in Abbildung 3.9 aufgezeigt sind. Der Reihenschlüssel darf nur einmal vorkommen, da er den Datensatz eindeutig adressieren soll (im Beispiel: Name). Der Kolonnenschlüssel besteht aus je einem Identifikator für die Spaltenfamilie und die jeweilige Spalte (im Beispiel: Wohnanschrift und Straße). (vgl. Fasel 2016, S. 118) Er adressiert somit die entsprechende Eigenschaft des Datensatzes (vgl. Meier

und Kaufmann 2016, S. 227). Bei einer Aktualisierung eines Wertes – zum Beispiel einem Umzug – wird der Wert nicht überschrieben, sondern wird „mit einem neuen Zeitstempel“ in die Zelle aufgenommen, wodurch eine Versionierung entsteht (Fasel 2016, S. 118).

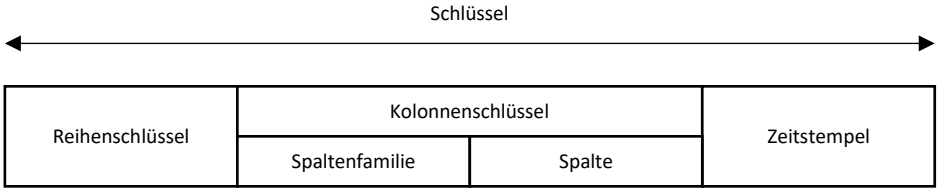

Abbildung 3.9: Zusammensetzung des Schlüssels (nach McCreary und Kelly 2014, Abb. 4.19)

Einer der Vorteile von Spaltenfamilien-Datenbanken ist, dass sie eine sehr hohe Skalierbarkeit durch die Möglichkeit, die Spaltenfamilien auf mehrere Knoten in einem Computercluster zu verteilen besitzen. Im Falle von zunehmenden Datenmengen werden lediglich neue Knoten benötigt, die sich zum Cluster hinzufügen lassen. (vgl. McCreary und Kelly 2014, S. 84) Diese Form von horizontaler Skalierung resultiert in einer Erhöhung der Performance (vgl. Sadalage und Fowler 2013, S. 107). Durch die Verteilung auf viele Knoten und der Eigenschaft, dass ein Cluster keinen Master hat und alle Knoten Peers sind, ist die Verfügbarkeit der Daten ebenfalls sehr hoch, da die anderen Knoten übernehmen können wenn ein Knoten ausfällt (vgl. Sadalage und Fowler 2013, S. 104). Die Flexibilität des Datenbanktyps ist hoch, weil das Einfügen von neuen Spalten einen sicheren Umgang mit unvorhersehbaren Szenarien und neuen Anforderungen zur Folge hat (vgl. Vaish 2013, S. 27) und abgesehen von den Spaltenfamilien „keine Kolonnen und Datentypen definiert werden müssen“ (Fasel 2016, S. 122). Die einzigen festen Schemaregeln sind die Spaltenfamilien, die lediglich durch eine „Änderung des Schemas“ erstellt werden können (Meier und Kaufmann 2016, S. 227).

Die Spaltenfamilien-Datenbanken werden bevorzugt dort eingesetzt, wo die Objektstrukturen komplex und möglichst schemafrei sind (vgl. Fasel 2016, S. 122). Allerdings ist anzumerken, dass sie bei frühen Prototypen nicht die bevorzugte Wahl sind, da sich in diesem Entwicklungsstadium noch viel ändern kann. Eine Änderung bedarf einer Anpassung des Designs der Spaltenfamilien und das würde die Entwickler von wichtigeren Arbeiten abhalten. (vgl. Sadalage und Fowler 2013, S. 109) Des Weiteren sind diese Typen für Anwendungen zu gebrauchen, die eine hohe Leistungsfähigkeit bei Lese- und Schreiboperationen voraussetzen (vgl. Fasel 2016, S. 122). Bei einer hohen Anforderung nach der Konsistenz von Daten, ist diese Datenbank jedoch nicht geeignet (vgl. Sadalage und Fowler 2013, S. 109).

3.4.4 Dokument-Datenbanken

Die Dokument-Datenbanken sind die flexibelsten, mächtigsten und verbreitetsten NoSQL-Datenbanktypen (vgl. McCreary und Kelly 2014, S. 86) und lassen sich als eine Erweiterung der Schlüssel-Wert-Datenbanken sehen, denn die Speicherstruktur mit Schlüssel-Wert-Paaren ist die gleiche (vgl. Fasel 2016, S. 115). Der große Unterschied liegt darin, dass die Dokument-Datenbank die Möglichkeit bietet, gespeicherte Daten zu strukturieren (vgl. Meier und Kaufmann 2016, S. 228). Der Begriff „Dokument“ ist dabei einzugrenzen. Es sind strukturierte Daten in Datensätzen gemeint und nicht beliebige Dokumente, wie zum Beispiel „Video- oder Audiodateien“ (Meier und Kaufmann 2016, S. 229). Die abgespeicherten Dokumente sind unabhängig voneinander, weshalb es in ihrer Struktur eminente Unterschiede geben kann (vgl. Fasel 2016, S. 115). Ein Beispiel für die unterschiedliche Struktur der Dokumente soll in Abbildung 3.10 aufgezeigt werden.

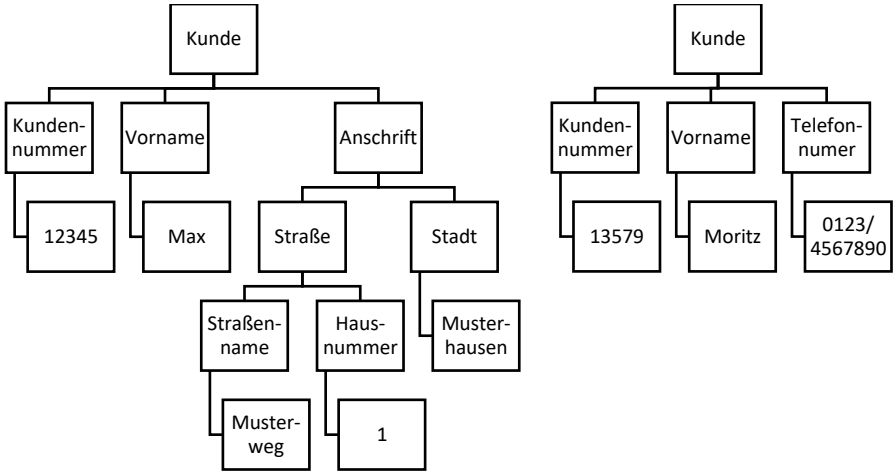

Abbildung 3.10: Zwei Dokumente unterschiedlicher Struktur (angelehnt an McCreary und Kelly 2014, Abb. 4.23)

Es sind zwei Dokumente zu sehen, die als Baumstruktur dargestellt werden. An der Spitze steht der Ursprung, von dem es über Verzweigungen zu den Werten geht. Es fällt auf, dass sich beide Dokumente ähnlich sind, obwohl sie nicht die gleichen Attribute besitzen. Trotz unterschiedlichem Schema können die Dokumente zu einer Dokument-Kollektion zusammengefasst werden (vgl. Sadalage und Fowler 2013, S. 90). Das Zusammenfassen in Kollektionen (document collections) ist eine Möglichkeit, große Dokument-Datenbanken zu managen. Solche Kollektionen können weitere Kollektionen enthalten, wodurch die Flexibilität von diesem Datenbanktyp ansteigt. (vgl. McCreary und Kelly 2014, 88) Für das obige Beispiel könnte die Dokument-Kollektion wie in Abbildung 3.11 aussehen. Es wurden die beiden Kunden zu einer Gruppe „Kundenverzeichnis“ zusammengefasst.

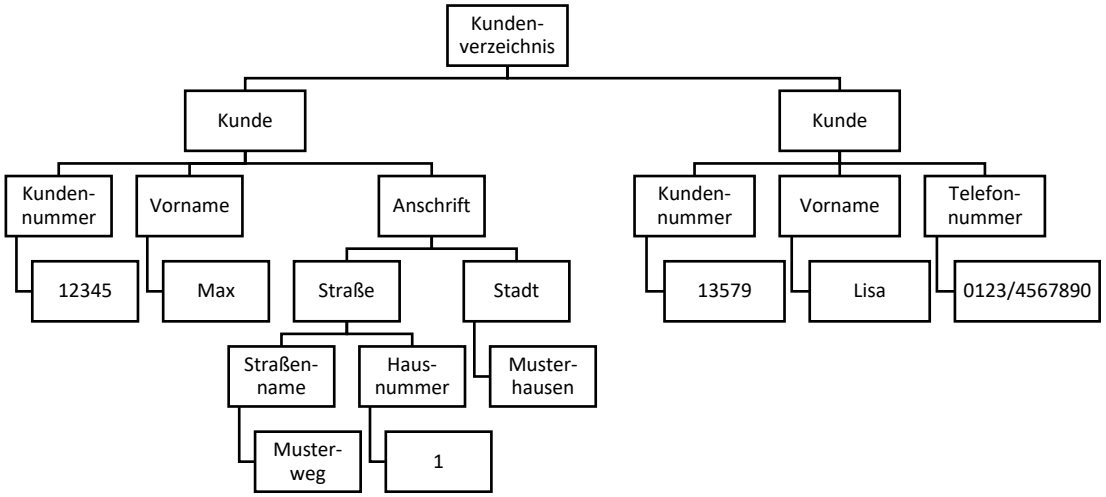

Abbildung 3.11: Kollektion aus zwei Dokumenten (angelehnt an McCreary und Kelly 2014, Abb. 4.23)

Bei Dokument-Datenbanken werden die Attribute, beziehungsweise Eigenschaften (property) jedes enthaltenen Dokuments indexiert, wodurch sich die Datenbanken durchsuchen lassen. Beispielsweise kann sie nach einer Eigenschaft gefiltert werden und als Ergebnis werden alle Dokumente aufgezählt, die diese Eigenschaft besitzen. Dabei kann neben der Rückmeldung, ob die Eigenschaft enthalten ist, auch der exakte Pfad zu dieser ermittelt werden. (vgl. McCreary und Kelly 2014, S. 87) Die Anfragen an diesen Datenbanktyp können mithilfe des MapReduce-Verfahrens (Abschnitt 3.4.1) „parallelisiert und somit beschleunigt werden“ (Meier und Kaufmann 2016, S. 231). Des Weiteren können mit Verweisen auf die Schlüsselwerte von Dokumenten auch Relationen zwischen Dokumenten hergestellt werden (vgl. Fasel 2016, S. 115). Allerdings sind diese Relationen nur einfach und es wird keine komplexe Produktbildung zwischen den Dokumenten erstellt (vgl. Fasel 2016, S. 118). Die Verfügbarkeit wird durch das Replizieren der Daten mit dem Master-Slave Setup verbessert. Dabei ist eine Kopie auf mindestens zwei Servern verfügbar: Einem Master Server und mindestens einem Slave Server. Die Server wählen unter sich den Master aus, an den alle Anfragen gestellt werden. Bei Änderungen werden die Daten an die Slaves weitergeleitet. Wenn der Master Server ausfallen sollte, wird der neue Master unter den verbliebenen Slaves ausgewählt. Aus diesem Grund besitzen Dokument-Datenbanken eine hohe Ausfalltoleranz. (vgl. Sadalage und Fowler 2013, S. 93) Da die Verfügbarkeit und die Ausfall-Toleranz hoch sind, kann die Konsistenz der Daten nicht jederzeit gewährleistet werden (vgl. Meier und Kaufmann 2016, S. 229). Es gibt mehrere Alternativen, die Datenbank zu skalieren. Um die Leseleistung zu erhöhen, werden Slave Server hinzugefügt, an die die Anfragen weitergeleitet werden und die Schreibperformance lässt sich durch das Sharding erhöhen. Dabei werden die Daten auf unterschiedliche Server aufgeteilt (vgl. Sadalage und Fowler 2013, S. 96). Die Schemafreiheit der Dokument-Datenbanken hat einige Vor- und Nachteile. Zu den Vorteilen gehören die hohe „Flexibilität in der Speicherung unterschiedlichster Daten“ (Meier und Kaufmann 2016, S. 229) und die erleichterte Zergliederung der

Datenbasis. Allerdings wird aufgrund des fehlenden Schemas auf referentielle Integrität und Normalisierung verzichtet. (vgl. Meier und Kaufmann 2016, S. 229)

Der entscheidende Vorteil der Dokument-Datenbanken liegt darin, dass sie im Gegensatz zu Schlüssel-Wert-Datenbanken „bessere Indexierungsmöglichkeiten zum Selektieren und Filtern von Datensätzen“ (Fasel 2016, S. 117) bieten und die enthaltenen Werte komplexer sein können. Dadurch lässt sich dieser Typ als vollwertige Datenbank verwenden. (vgl. Fasel 2016, S. 117) Er wurde entwickelt, um für Webdienste eingesetzt zu werden und darum liegt der Fokus auf der „Verarbeitung von großen Mengen heterogener Daten“ (Meier und Kaufmann 2016, S. 229).

3.4.5 Graphdatenbanken

Der letzte Typ der Kerntechnologien im Bereich NoSQL, ist die Graphdatenbank. Sie unterscheidet sich deutlich von den anderen Datenbanktypen, da sie das strukturierende Schema eines Eigenschaftsgraphen besitzt. (vgl. Meier und Kaufmann 2016, S. 237) Der Eigenschaftsgraph besteht aus Knoten, die für Konzepte und Objekte stehen, und gerichteten Kanten, welche für Beziehungen zwischen den Knoten stehen und diese verbinden. Jeweils Knoten und Kanten haben einen Namen und können mit einer Eigenschaft versehen sein, die als Attribut-Wert-Paar mit einem Attributnamen und einem dazugehörigen Wert charakterisiert wird. (vgl. Meier und Kaufmann 2016, S. 14) Durch dieses Schema sind die Beziehungen zwischen den Knoten als Kanten vorhanden (vgl. Meier und Kaufmann 2016, S. 238). Nach Meier und Kaufmann (2016, S. 237) besteht der Unterschied zu relationalen Datenbankmanagementsystemen darin, dass „Datenobjekte zu einem bisher nicht existierenden Knoten- oder Kantentyp direkt in die Datenbank eingefügt werden können“. Dabei muss der Typ der Knoten, beziehungsweise der Kanten, vorher nicht definiert werden. Es handelt sich um ein implizites Schema. (vgl. Meier und Kaufmann 2016, S. 237) Laut McCreary und Kelly (vgl. 2014, S. 72) werden einige Graphdatenbanken auch als Tripel Datenbanken bezeichnet, da sie die Struktur Knoten-Beziehung-Knoten haben.

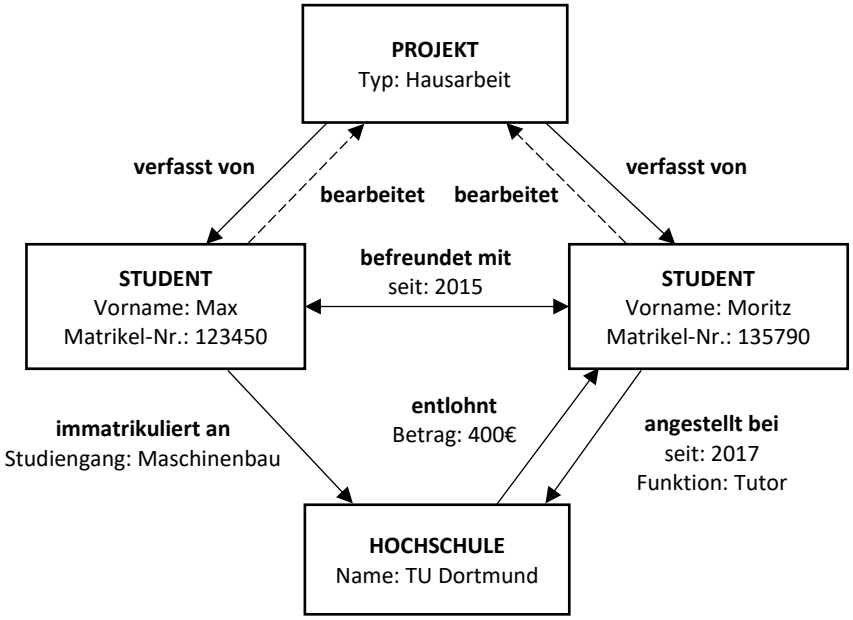

Abbildung 3.12: Beispiel einer Graphdatenbank (angelehnt an Meier und Kaufmann 2016, Abb. 7.6 und Sadalage und Fowler 2013, Abbildung 11.2)

Abbildung 3.12 soll als Beispiel für eine Graphdatenbank dienen. Die Knoten (Rechtecke) und Kanten (Pfeile) haben eine Bezeichnung (fett gedruckte Wörter) und sind eventuell mit Attributen versehen (normale Schriftstärke). Die Beziehung zwischen den beiden Studenten ist bidirektional, da sie in beide Richtungen gültig ist (vgl. Sadalage und Fowler 2013, S. 111). Zwischen zwei Knoten können mehr als eine Beziehung herrschen, wie es zwischen dem Studenten „Moritz“ und der Hochschule „TU Dortmund“ der Fall ist. Er ist dort als Tutor angestellt, wofür er von der Hochschule entlohnt wird. Laut Sadalage und Fowler (vgl. 2013, S. 113) ist es einfach, neue Kanten hinzuzufügen. So könnte beispielsweise die Beziehung „bearbeitet“ erstellt werden, die jeweils von den Studenten zum Projekt gerichtet ist (gestrichelte Linie). Wenn bei der Änderung einer Beziehung Abwärtskompatibilität gefragt ist oder der Graph nicht sofort komplett abgeändert werden soll, werden neue Kanten zwischen den Knoten erstellt und die alten Kanten gegebenenfalls zu einem späteren Zeitpunkt gelöscht (vgl. Sadalage und Fowler 2013, S. 131). Meier und Kaufmann (vgl. 2016, S. 238) beschreiben diese Datenmanipulationen als Graph-Transformationen.

Laut Meier und Kaufmann (vgl. 2016, S. 238) wird die referentielle Integrität vom Datenbankverwaltungssystem sichergestellt. Die gespeicherten Daten bleiben durch Transaktionen konsistent. Ihre Verwendung stellt sicher, dass jede Beziehung einen Start- und Endknoten besitzt. Vorhandene Knoten können demnach nur gelöscht werden, wenn keine Beziehungen an ihnen anliegen. (vgl. Sadalage und Fowler 2013, S. 114) Laut Fasel (vgl. 2016, S. 122) ist die horizontale Skalierbarkeit im Allgemeinen schwierig, da sich Graphen nicht einfach auf mehrere Maschinen verteilen lassen. Nach Sadalage und Fowler (vgl. 2013, S. 10) besitzen Graphdatenbanken ein ähnliches Verteilungsmodell wie relationale Datenbanken, wodurch die Verteilung erschwert wird. Ein weiterer Grund der schlechten

Skalierbarkeit ist die enge Verbindung der Knoten (vgl. McCreary und Kelly 2014, S. 73). Wenn der Graph auf mehrere Maschinen verteilt sein sollte, geht das Durchlaufen von diesem außerdem zu Lasten der Performance (vgl. Sadalage and Fowler 2013, S. 119). Nach Meier und Kaufmann (vgl. 2016, S. 239) wird das Sharding von heutigen Datenbanken nicht unterstützt und es gibt keine effiziente Methode für das Zerlegen eines Graphen in Teilgraphen, da die existierenden Algorithmen mit exponentiellem Aufwand verbunden sind. Anders hingegen ist die Situation bei der Skalierung zur Verbesserung der Leistungsfähigkeit von Abfragen und Leseanfragen aus. Dafür werden Slave-Server hinzugefügt, von denen der Datensatz nur gelesen werden kann. Die Schreibanfragen werden zum Master-Server geleitet. (vgl. Sadalage und Fowler 2013, S. 119) Der Datensatz wird dabei auf die Slave-Server repliziert (vgl. McCreary und Kelly 2014, S. 73). Nach Sadalage und Fowler (vgl. 2013, S. 119) ist diese Art der Skalierung sinnvoll, wenn der Datensatz zu groß für den Arbeitsspeicher einer Maschine ist, aber immer noch klein genug, um komplett auf mehreren Maschinen repliziert zu werden. Dank der Replikation auf andere Server wird eine hohe Verfügbarkeit der Daten sichergestellt (vgl. Sadalage und Fowler 2013, S. 115). Eine weitere Eigenschaft dieses Graphen ist die Schemaflexibilität, da sich neue Knoten und Kanten effektiv in ihn einbinden lassen (vgl. Fasel 2016, S. 124). „Die Kosten, den Graphen abzufragen," steigen dabei nicht (Fasel 2016, S. 122). Auch bei größeren Datensätzen bleibt der Aufwand für Abfragen konstant. Begründet ist das durch die indexfreie Nachbarschaft von Graphdatenbanken. Dieses Merkmal beschreibt die Fähigkeit von Datenbanksystemen, zu jedem Knoten die direkten Nachbarn zu finden, ohne dass alle Kanten in Betracht gezogen werden müssen. (vgl. Meier und Kaufmann 2016, S. 238) Nach Sadalage und Fowler (vgl. 2013, S. 113) werden die meisten Werte aus Graphdatenbanken aus den Beziehungen zwischen den Knoten abgeleitet.

Die Besonderheit von Graphdatenbanken liegt darin, dass sich Beziehungen einfach darstellen, abfragen und bearbeiten lassen (vgl. Vaish 2013, S. 45). Im Gegensatz zu relationalen Datenbanken besitzen Graphdatenbanken ein Datenmodell, mit dem komplexe Beziehungen besser verarbeitet werden können (vgl. Sadalage und Fowler 2013, S. 10) Deshalb werden sie beispielsweise bei „sozialen Beziehungen zwischen Menschen, Verkehrsverbindungen zwischen Orten oder Netzwerktopologien zwischen verbundenen Systemen verwendet" (Vaish 2013, S. 43). In diesen Fällen sind die Beziehungen der Datensätze untereinander wichtiger als die einzelnen Datensätze (vgl. Meier und Kaufmann 2016, S. 238). Wenn es allerdings keine Beziehungen zwischen den Objekten gibt, besteht kein Anwendungsfall für Graphdatenbanken. (vgl. Vaish 2013, S. 45) Außerdem sind sie nicht für analytische Abfragen geeignet, die über viele Knoten und Attribute hinweg gehen (vgl. Meier und Kaufmann 2016, S. 124). Wenn aufgrund einer veränderten Eigenschaft alle Objekte, beziehungsweise eine Teilmenge davon, aktualisiert werden müssen, so ist dieser Datenbanktyp ebenfalls unpraktisch, da es keine einfache Operation dafür gibt (vgl. Sadalage und Fowler 2013, S. 121).

3.5 Polyglot Persistence

Jeder der beschriebenen NoSQL-Datenbanktypen wird für bestimmte Szenarien eingesetzt. Durch die Gruppierung von verschiedenen Datenbanken in einer Anwendung lassen sich die Vorteile der einzelnen Typen kombinieren. Dieses Vorgehen wird als Polyglot Persistence beschrieben. (vgl. Tiwari 2011, S. 271) Der Begriff lässt sich nach Meier und Kaufmann (vgl. 2016, S. 222) mit „mehrsprachige Persistenz“ wörtlich ins Deutsche übersetzen. Die Nutzung von mehreren Lösungen für ein Problem ist zwar komplexer als die Nutzung einer einzigen Datenbank, jedoch überwiegen die dadurch entstehenden Vorteile den höheren Aufwand (vgl. Sadalage und Fowler 2013, S. 148). Bei der Auswahl des richtigen Datenbanktyps ist es wichtig zu wissen, wie die verschiedenen Ansätze funktionieren und wie mit ihnen gearbeitet wird (vgl. Sadalage und Fowler 2013, S. 138). Ein Beispiel für die Nutzung mehrerer Datenbanken soll Abbildung 3.13 geben. Ein Onlineshop hat verschiedene Möglichkeiten, die anfallenden Daten zu speichern. Wenn ein Benutzer die Webseite besucht und verschiedene Waren bestellen möchte, können die Daten zu seiner Sitzung und dem Inhalt seines Warenkorbs in einer Schlüssel-Wert-Datenbank abgespeichert werden, da sie nach dem Besuch nicht mehr gebraucht werden. Die bei einer abgeschlossenen Bestellung anfallenden Daten können in einer relationalen Datenbank hinterlegt werden. Soll dem Benutzer während des Einkaufes beispielsweise vorgeschlagen werden, welche anderen Produkte von Kunden bestellt wurden, die sich ebenfalls für das Produkt entschieden haben, lässt sich dies durch Einsatz einer Graphdatenbank erreichen. (vgl. Sadalage und Fowler 2013, S. 134 f.)

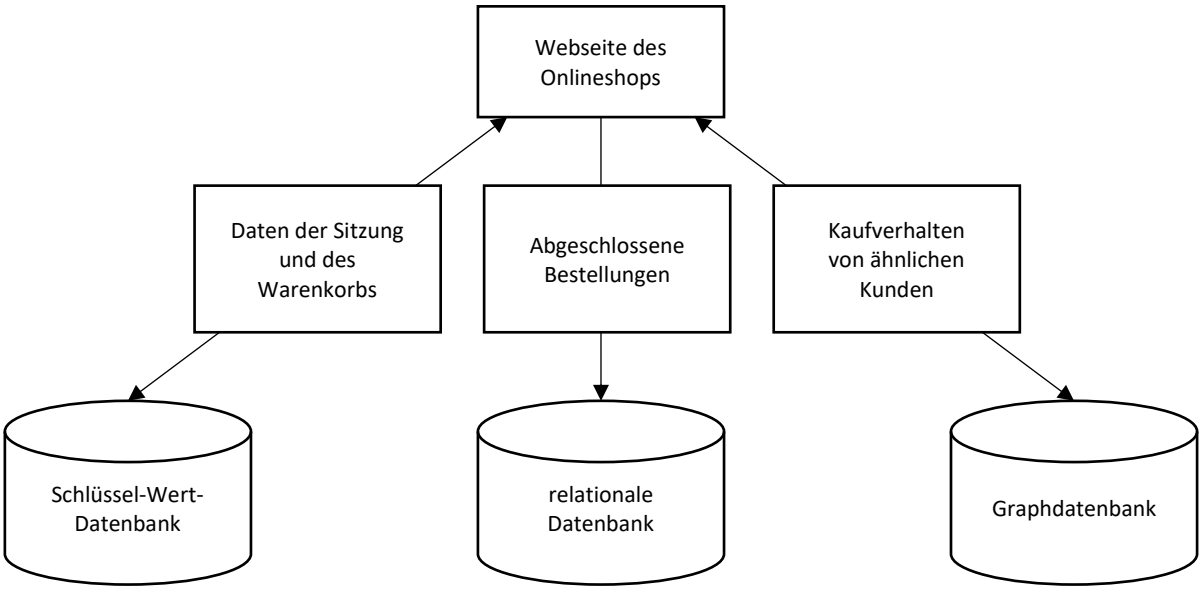

Abbildung 3.13: Beispiel für Polyglot Persistence (angelehnt an Sadalage und Fowler 2013, Abb. 13.2 und 13.3)

Nach Sadalage und Fowler (vgl. 2013, S. 138 f.) gibt es trotz der vielseitigen Einsatzmöglichkeiten einige Bedenken seitens der Unternehmen, zu denen beispielsweise der Sicherheitsaspekt oder die Frage der Kompatibilität zwischen dem NoSQL-Tool und dem Data-Warehouse-System zählt. Obwohl die NoSQL-Technologie durch Polyglot Persistence viele Vorteile bietet, sollten auch andere Datenbanktechnologien in Betracht gezogen werden (vgl. Sadalage und Fowler 2013, S. 146).

4 Methode zur Prüfung der Eignung von NoSQL-Datenbanksystemen zur Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie

Dieses Kapitel beschäftigt sich detailliert mit der Methodenentwicklung zur Prüfung der Eignung von NoSQL-Datenbanksystemen zur Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie. Diese Methode beschränkt sich auf die Prüfung der vier NoSQL-Core-Modelle. In Abschnitt 4.1 wird beschrieben, wie neue Technologien bewertet werden können und es werden einige Bewertungsverfahren vorgestellt. Aus diesen Bewertungskriterien wird in Abschnitt 4.2 eines ausgewählt, das als Grundlage für die zu entwickelnde Methode benutzt wird. Der erste Schritt ist die Identifikation möglicher relevanter Kriterien, die auf Basis der Erkenntnisse aus Kapitel 2 und 3 gewählt werden. Danach werden die Kriterien gefiltert, sodass eine Auswahl erstellt wird, mit der die Nutzwertanalyse durchgeführt werden kann. (Abschnitt 4.3.1) In Abschnitt 4.3.2 werden die Merkmalsausprägungen der vier NoSQL-Datenbanktypen in Bezug auf die Bewertungskriterien bestimmt. Danach werden die Kriterien in Abschnitt 4.3.3 nach ihrer Relevanz mit Gewichtungsfaktoren versehen. Die Verteilung dieser Faktoren basiert auf den Anforderungen, die von vernetzten Produktionssystemen an Datenbanken gestellt werden. Im letzten Schritt wird die Nutzwertanalyse durchgeführt. Das Ergebnis der Nutzwertanalyse liefert eine Grundlage zur Entscheidungsfindung.

4.1 Definitionen unterschiedlicher Methoden zur Technologiebewertung

Um die Wettbewerbsfähigkeit eines Unternehmens zu steigern, müssen die innovativen Technologien des technischen Fortschritts untersucht werden. Die Implementierung von Innovationen hat beispielsweise die Steigerung der Produktqualität oder Leistungsfähigkeit von Produktionsmitteln zur Folge. Das Technologiemanagement beschäftigt sich mit der Planung zur langfristigen Sicherung von Unternehmen und der Stärkung ihrer Marktposition. (vgl. Klappert et al. 2011, S. 6) Bei dieser Aktivität steht die „gezielte Änderung einer Technologie, eines Produktes oder der eingesetzten Produktionstechnologie“ (Klappert et al. 2011, S. 6) im Fokus. Eine weitere Aufgabe ist es, die Änderung zur richtigen Zeit und zu akzeptablen Kosten durchzuführen. Dabei werden lediglich Technologien implementiert, die zur Abarbeitung von aktuellen und zukünftigen Leistungen nötig sind. (vgl. Klappert et al. 2011, S. 5) Nach Schuh et al. (vgl. 2011, S. 15) besteht das Technologiemanagement aus sechs Grundaktivitäten, die miteinander vernetzt sind:

- Technologiefrüherkennung
- Technologieplanung
- Technologieentwicklung
- Technologieverwertung
- Technologieschutz

- Technologiebewertung

Im Folgenden wird auf die Technologiebewertung eingegangen. Dieser Bereich ist eine Voraussetzung zur effizienten Umsetzung des Technologiemanagements. Er ermittelt für ein bestimmtes Bewertungsobjekt den Grad der Erfüllung von Zielzuständen. Auf Basis dieser Bewertung können Entscheidungen über Technologien, bezüglich ihrer Entwicklung, ihrer Einführung oder ihrem Einsatz getroffen werden. (vgl. Schuh et al. 2011, S. 17) Für unterschiedliche Entscheidungssituationen können dafür verschiedene Methoden der Technologiebewertung genutzt werden. Diese unterteilen sich in eher quantitative oder eher qualitative Verfahren. Qualitative Methoden werden hauptsächlich in der Technologiefrüherkennung eingesetzt und quantitative Methoden meist in den Phasen der Technologieplanung, -entwicklung oder -verwertung. (vgl. Haag et al. 2011, S. 324) Der Unterschied dieser Verfahren liegt in der Art der Bewertungskriterien des untersuchten Bewertungsobjektes. Es ist möglich, eine Merkmalsausprägung in Worten (qualitativ) oder in Zahlen (quantitativ) auszudrücken. Eine zusammenfassende Bewertung von den unterschiedlichen Arten der Kriterien kann problematisch sein. (vgl. Hoffmeister 2008, S. 276) Quantitative Bewertungsverfahren, wie zum Beispiel die Methoden der Investitions- und Wirtschaftlichkeitsrechnung, haben durch ihre Beschränkung auf monetäre oder ökonomische Kriterien oftmals nicht die Möglichkeit, ein Problem vollkommen zu erfassen oder zu lösen. (vgl. Hoffmeister 2008, S. 277) Wenn sich die Kriterien nicht quantitativ ausdrücken lassen, da sie beispielsweise keine monetäre Beschreibung zulassen, muss die Entscheidung aufgrund einer überwiegend qualitativen Bewertung getroffen werden (vgl. Haag et al. 2011, S. 316) Es wird eine Auswahl von Methoden zur Technologiebewertung beschrieben, wovon eine die Grundlage für die in dieser Arbeit zu entwickelnde Methode darstellen soll.

Argumentenbilanz

Diese Bewertungsmethode ist das simpelste und bekannteste Verfahren und stellt die Vorteile und Nachteile der zu untersuchenden Technologie in Listen dar. Sie wird bei Problemen eingesetzt, die sich nicht monetär oder quantitativ beschreiben lassen. Allerdings lassen sich die Kriterien durch ihre verbale Beschreibung nicht präzisieren. Es lässt sich mit dieser Methode lediglich ein grober Überblick über diverse Technologien schaffen. (vgl. Haag et al. 2011, S. 324 ff.)

Checklisten Methoden

Bei Checklisten lassen sich mit geringem Aufwand alle wichtigen Kriterien eines bestimmten Themas auf pragmatische Weise erfassen. Das Ergebnis ist eine Handlungsempfehlung, die aus einer Priorisierung der untersuchten Technologien resultiert. Bei dieser Ordnung ist die Technologie mit den am meisten erfüllten Kriterien am relevantesten. Die Kriterien lassen sich bei dieser Methode frei auswählen und eine Checkliste ist einfach durch zusätzliche Kriterien ergänzbar. Schwachstellen dieses Verfahrens sind die nicht definierte Zahl der zu berücksichtigenden Kriterien und die Möglichkeit, dass

Kriterien untereinander umstrittene Wirkungsrichtungen besitzen können. (vgl. Haag et al. 2011, S. 326 f.)

Nutzwertanalyse

Die Nutzwertanalyse ist eine Methode zur Beurteilung von mehreren Technologien, um diese Alternativen „entsprechend den Präferenzen des Entscheidungsträgers bezüglich eines multidimensionalen Zielsystems zu ordnen“ (Zangemeister 1976, S. 45). Zur Ordnung der Technologien dient der Nutzwert, der sich für jedes untersuchte Element aus der Bewertung der Kriterien zusammensetzt und keine Dimension besitzt (vgl. Zangemeister 1976, S. 45). Die identifizierten Kriterien sind qualitativ und zudem unabhängig voneinander zu wählen. Sie werden durch subjektive Einschätzung quantifiziert und lassen sich aus diesem Grund vergleichen. (vgl. Haag et al. 2011, S. 327) Bei dieser Einschätzung wird festgelegt, in wie weit ein Kriterium von einer Technologie erfüllt wird, und je nach Erfüllungsgrad wird der Technologie für dieses Kriterium ein Skalenwert zugeteilt (vgl. Haag et al. 2011, S. 329). Bei dieser Methode kann zudem die Relevanz von einzelnen Kriterien durch Einsatz des Gewichtungsfaktors berücksichtigt werden (vgl. Haag et al. 2011, S. 328). Sie findet vorzugsweise in Entscheidungssituationen Anwendung, die keine monetäre Bewertung zulassen oder die aus vergleichbaren Technologien besteht oder bei denen mehrere entscheidungsrelevante Merkmale zu beachten sind (vgl. Hoffmeister 2008, S. 279).

Kosten-Nutzen-Analyse

Bei der Kosten-Nutzen-Analyse werden die Kosten für die Implementierung der Technologie mit dem Nutzen, der durch die Einführung der jeweiligen Technologie erreicht wird, verglichen. Besitzt eine Technologie qualitative Bewertungskriterien, werden sie zur Beurteilung in Geldgrößen umgewandelt. Die Differenz des monetären Nutzens der Technologie und ihrer Anschaffungskosten wird als Nettonutzen bezeichnet. Auf der Basis dieses Wertes lassen sich verschiedene Technologien vergleichen. Jedoch ist die Umwandlung von qualitativen Auswirkungen in monetäre Größen schwierig, weshalb der praktische Nutzen dieser Methode oftmals geringer ist, als angenommen. (vgl. Haag et al. 2011, S. 329 f.)

Total Cost of Ownership

Diese Methode betrachtet nicht nur die Anschaffungskosten, sondern alle über den kompletten Lebenszyklus der Technologie anfallenden Ausgaben. Somit werden auch die Bereiche der Technologienutzung, -verwaltung und -wartung berücksichtigt. Die Stärke des Verfahrens liegt in der Aufteilung der anfallenden Kosten auf verschiedene Bereiche des Lebenszyklus. Dadurch kann zum Beispiel identifiziert werden, dass eine in der Anschaffung kostspieligere Technologie durch ihre geringen Verwaltungskosten auf lange Sicht rentabler ist als ihre preisgünstigeren Alternativen. Jedoch werden bei

dieser Methode sämtliche qualitative Faktoren außenvorgelassen, weshalb auf eine Kombination verschiedener Bewertungsmethoden zu setzen ist. (vgl. Haag et al. 2011, S. 342 ff.)

4.2 Auswahl des geeigneten Bewertungsverfahrens

Um ein geeignetes Bewertungsverfahren für die Methode der Eignungsüberprüfung verschiedener NoSQL-Datenbanktypen zu finden, wird in einem ersten Schritt auf die in Kapitel 4.1 vorgestellten Verfahren eingegangen. Auf Basis dieser Bewertung wird ein Verfahren ausgewählt, das als Grundlage für die zu entwickelnde Methode benutzt wird. Es sei angemerkt, dass in dieser Arbeit die unterschiedlichen Datenbanktypen im Fokus stehen und nicht die Tools, die auf dem Markt angeboten werden.

Argumentenbilanz

Bei diesem Verfahren ist es von Vorteil, dass sich qualitative Kriterien gegenüberstellen lassen. Diese Eigenschaft ist bei der Bewertung von Datenbanktypen essenziell, da die verwendeten Kriterien ausschließlich qualitativ sind. Da diese Methode jedoch lediglich einen Überblick über die verschiedenen Datenbanktypen geben kann und die Typen nicht in Hinblick auf ihre Fähigkeit, die Daten in vernetzten Produktionssystemen der Automobilindustrie abzuspeichern, eingeordnet werden, eignet es sich nicht als Bewertungsverfahren.

Checklisten Methoden

Obwohl dieses Verfahren eine Handlungsempfehlung bereitstellt, ist deren Aussagekraft dennoch in Frage zu stellen. Es könnte bloß untersucht werden, ob ein Datenbanktyp das Kriterium erfüllt oder nicht. Der Grad der Erfüllung würde dabei ebenso unberücksichtigt bleiben, wie die Gewichtung des Kriteriums. Bei einer detaillierten Bewertung ist eine Gewichtung der Kriterien nötig, um deren Relevanz zu beurteilen. Darüber hinaus könnte das Ergebnis verfälscht werden, wenn die Kriterien nicht alle Anforderungen von Datenbanken abdecken oder wenn sich mehrere Kriterien in ihrer Kernaussage ähneln, bzw. in Beziehung zueinander stehen. Dies würde zu einer Verschiebung des Ergebnisses führen. Aufgrund dieser Argumente lässt sich zusammenfassen, dass diese Methode ebenfalls nicht als Bewertungsverfahren in Frage kommt.

Nutzwertanalyse

Bei der Überprüfung der Eignung von NoSQL-Datenbanktypen existieren eine Vielzahl von entscheidungsrelevanten Kriterien, die sich nicht durch monetäre Größen beschreiben lassen. Außerdem haben die verschiedenen Datenbanktypen im Groben den gleichen Aufbau und lassen sich somit durch Kriterien vergleichen. Diese Eigenschaften stimmen mit den Einsatzgebieten von Nutzwertanalysen überein. Nicht jeder Datenbanktyp erfüllt ein Kriterium gleich gut, weshalb die Einführung eines Erfüllungsgrades von Vorteil ist. Ebenso ist die Möglichkeit, den Bewertungskriterien durch den Einsatz eines Gewichtungsfaktors unterschiedliche Relevanz zuzuteilen, bei einer Bewertung überaus wichtig.

Mit dieser Eigenschaft lassen sich beispielsweise die Anforderungen von den vernetzten Produktionssystemen der Automobilindustrie berücksichtigen. Somit ist die Nutzwertanalyse als Bewertungsverfahren für NoSQL-Datenbanktypen geeignet.

Kosten-Nutzen-Analyse

Da eine typbasierte Bewertung erstellt werden soll, sind die Anschaffungskosten für die Implementierung einer neuen Datenbank nicht abzusehen. Neben diesen Kosten müssten die aufgewendeten Ressourcen berücksichtigt werden, die zur erfolgreichen Umsetzung der neuen Struktur nötig wären. Dieser Umfang lässt sich allerdings ebenso wenig bestimmen wie die Nutzen, die durch die neue Datenbank entstehen. Aus einer höheren Zugriffsgeschwindigkeit auf die Daten, der damit einhergehenden höheren Geschwindigkeit von Analysen oder der erreichten Ausfallsicherheit durch die Verteilung auf viele Rechner lassen sich keine monetären Werte ziehen. Es wäre zwar interessant, die gleichen Aufgaben von einer relationalen Datenbank und einer NoSQL-Datenbank lösen zu lassen, um daraus Schlüsse zur Leistungssteigerung ziehen zu können, jedoch ließen sich diese Ergebnisse wohl nicht pauschalisieren, da jedes System durch unterschiedliche Faktoren beeinflusst wird. Aus diesen Gründen ist die Kosten-Nutzen-Analyse, ebenso wie alle anderen quantitativen Verfahren nicht anwendbar.

Total Cost of Ownership

Von dieser Methode wird der Ansatz verfolgt, dass sie nicht nur die Anschaffungskosten berücksichtigt, sondern auch die laufenden Kosten für Nutzung, Verwaltung und Wartung. Da sich die Datenbanktypen nicht durch monetäre Werte beschreiben lassen und dieses Verfahren sämtliche qualitativen Werte ausschließt, ist es ebenfalls kein geeignetes Bewertungsverfahren.

Aus der Bewertung der verschiedenen Bewertungsmethoden lässt sich der Schluss ziehen, dass eine qualitative Methode gewählt werden muss, da die quantitative Methoden zur Bewertung von verschiedenen Datenbanktypen nicht geeignet sind. Es wäre zwar wünschenswert vor dem Einsatz einer neuen Technologie zu wissen, ob sich der daraus entstehende Aufwand rentiert und wie groß die Einsparpotenziale sind, jedoch lässt sich dies nicht umsetzten. Von den bewerteten qualitativen Verfahren eignet sich die Argumentenbilanz, um einen Überblick über die verschiedenen Datenbanktypen zu erlangen. Für eine Entscheidung, welcher Typ bei einer Anwendung eingesetzt werden soll, bietet sie allerdings nicht genug Grundlagen. Einen Schritt weiter geht die Checklisten Methode, mit der eine grobe Priorisierung der Datenbanken erstellt werden kann. Allerdings ist dieses Verfahren noch sehr ungenau, da es dem Anwender nicht die Möglichkeit bietet, die verschiedenen Kriterien zu gewichten. Außerdem wird bei einem Kriterium nur unterschieden, ob es erreicht wird oder nicht. Eine Abstufung, die den Grad der Erfüllung widerspiegelt, ist nicht existent. Obwohl die Einführung einer neuen Technologie, die sich nicht durch eine Wirtschaftlichkeitsrechnung bewerten lässt, stets mit einem gewissen Risiko behaftet ist, sollte das Risiko dennoch möglichst gering sein. Besonders die Automobilindustrie ist

durch ihre vernetzten Fabriken, die auf den Einsatz hoch-performanter Produktionssysteme setzten, darauf bedacht, bei der Einführung ein möglichst geringes Risiko einzugehen und das lässt sich durch den Einsatz von einer Argumentenbilanz oder einer Checklistenmethode nicht eingrenzen. Aus diesen Gründen wird die Nutzwertanalyse als favorisiertes Bewertungsverfahren für NoSQL-Datenbanktypen ausgewählt. Sie soll die unterschiedlichen Datenbanktypen möglichst genau in Hinblick auf die Eignung zur Datenspeicherung beschreiben.

Sollen allerdings statt NoSQL-Datenbanktypen deren Tools analysiert werden, bietet das von Cooper et al. (vgl. 2010, S. 143) entwickelte Yahoo! Cloud Serving Benchmark (YCSB) eine Alternative. Dieses Framework erleichtert den Leistungsvergleich von NoSQL-Datenbanksystemen (vgl. Cooper et al. 2010, S. 143). Mit dem YCSB Client lassen sich unterschiedliche Betriebszustände durch unterschiedliche Auslastungen (workloads) simulieren und auswerten. Durch einen Vergleich der Ergebnisse kann der Anwender das für seinen Anwendungsfall passende Datenbanksystem auswählen. (vgl. Cooper et al. 2010, S. 154)

4.3 Entwicklung einer eigenen Methode zur Prüfung der Eignung von NoSQL-Datenbanksystemen zur Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie

In den folgenden Kapiteln wird auf Basis der Nutzwertanalyse eine Methode zur Prüfung der Eignung von NoSQL-Datenbanksystemen zur Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie entwickelt. Diese Methode bezieht sich auf die vier Core-NoSQL-Modelle Schlüssel-Wert- (siehe Abschnitt 3.4.2), Spaltenfamilien- (siehe Abschnitt 3.4.3), Dokument- (siehe Abschnitt 3.4.4) und Graphdatenbank (siehe Abschnitt 3.4.5). Die Bewertungskriterien, deren Gewichtung und Erfüllungsgrade werden aufbauend auf den in den Kapiteln 2 und 3 erarbeiteten Erkenntnissen identifiziert und erläutert. Das Ergebnis der Methode bietet eine Empfehlung, welcher Datenbanktyp bei vernetzten Produktionssystemen der Automobilindustrie eingesetzt werden sollte.

4.3.1 Bestimmung der Kriterien

Sollen NoSQL-Datenbanken zur Datenspeicherung in vernetzten Produktionssystemen verwendet werden, werden einige Anforderungen an die Datenbanken gestellt. Diese Anforderungen werden in Form von Kriterien formuliert, durch die sich die vier Modelle beschreiben und bewerten lassen. Zuerst wird eine Anzahl von Kriterien beschrieben, die für die Zielerreichung relevant sein könnten. Zu der Beschreibung soll ein selbst gewähltes Beispiel die Wichtigkeit des Kriteriums unterstreichen. Danach werden die beschriebenen Kriterien gefiltert, sodass in der Nutzwertanalyse lediglich jene eingesetzt werden, die zur eindeutigen Beschreibung der alternativen Datenbankmodelle dienen. Dadurch sollen unter anderem redundante Kriterien gestrichen werden. Damit die Nutzwertanalyse nicht zu kompliziert wird, werden nur so viele Kriterien wie nötig verwendet.

Administrationsaufwand

Der Administrationsaufwand einer Datenbank beschreibt die anfallenden verwaltungstechnischen Belastungen. Ein Beispiel für dieses Kriterium stellt der Aufwand, der bei der Implementierung des neuen Systems entsteht, oder das Hinzufügen eines neuen Rechners in das Computercluster dar (siehe Abschnitt 3.4.1). Allerdings lässt sich dieser Aufwand nicht auf einen Datenbanktypen verallgemeinern und ist zudem abhängig von anderen Faktoren, beispielsweise der Infrastruktur im Unternehmen. Deshalb wird dieses Kriterium nicht für die Nutzwertanalyse verwendet.

Ausfalltoleranz

Dieses Kriterium steht für die Eigenschaft des Datenbanksystems, seine Funktionsfähigkeit trotz Ausfalls im Computercluster aufrecht zu erhalten (siehe Abschnitt 3.4.1). Ein Ausfall kann durch defekte Bauteile in den Rechnern entstehen oder auch durch äußere Einflüsse, wie einem Stromausfall oder sogar einem Brand. Die Ausfalltoleranz ist für den effizienten Betrieb von vernetzten Produktionssystemen sehr wichtig. Falls durch Ausfälle nicht mehr auf das Datenbanksystem zugegriffen werden könnte, müsste der Betrieb im Produktionssystem stillgelegt werden, was zu den schlechtesten anzunehmenden Fällen gehört. Dieses Kriterium wird für die Nutzwertanalyse verwendet.

Entwicklungsreife

Aus der Entwicklungsreife könnten Rückschlüsse auf den Zustand der Qualität des Datenbanktechnologie getroffen werden. Handelt es sich um einen etablierten Datenbanktypen könnte dies bedeuten, dass er frei von Fehlern ist, viele Funktionen besitzt und mit anderen Systemen kompatibel ist. Ist ein Datenbanktyp noch nicht so alt, würde er demnach nicht so ausgereift sein wie der etablierte Datenbanktyp. Allerdings wurden die unterschiedlichen Datenbankmodelle für unterschiedliche Anwendungsfälle entwickelt (siehe Abschnitt 3.5). Aus diesem Grund kann ein Datenbanksystem eines nicht so weit entwickelten Typs besser für die vorliegende Anwendung gedacht sein, als ein Datenbanksystem von einem entwicklungsreiferen Typ, das jedoch wegen der Typcharakteristiken nicht zum Anwendungsfall passt. Als Beispiel wird die Speicherung von Daten gewählt, die in Beziehung zueinander stehen. Angenommen, der Typ der Graphdatenbanken besitzt eine niedrigere Entwicklungsreife als der Typ der Schlüssel-Wert-Datenbanken, eignet er sich trotzdem besser als die Schlüssel-Wert-Datenbank, da dieser Typ nicht für den Anwendungsfall entwickelt wurde. Des Weiteren hängt dieses Kriterium stark vom eingesetzten Tool ab und wird deshalb nicht in die Nutzwertanalyse übernommen.

Fachwissen

Dieses Kriterium beschreibt das zur Verfügung stehende Fachwissen. Darunter fallen Fachbücher, Veröffentlichungen oder auch der Support der Open Source-Community (siehe Abschnitt 3.4.1). Existieren beispielsweise viele Lehrbücher über einen Datenbanktypen, lassen sich auftretende Probleme einfacher beseitigen. Ist ein Datenbanktyp gut erforscht und somit viel über ihn bekannt, zeugt das von

einer hohen Entwicklungsreife. Allerdings kommen bei diesem Vergleich die gleichen Einschränkungen wie bei dem Kriterium der Entwicklungsreife auf. Das Fachwissen bezieht sich in den meisten Fällen wohl nicht auf einen Datenbanktypen, sondern auf eine bestimmte Software. Da die Tools nicht im Fokus dieser Arbeit stehen, wird auf dieses Kriterium bei der Nutzwertanalyse verzichtet.

Funktionsumfang

Mit dem Funktionsumfang werden die Anzahl der Funktionen eines Datenbanktypen eingegangen. Eine andere Beschreibung für dieses Kriterium wäre das der Anfragemächtigkeit. Wenn sich an eine Datenbank eine Vielzahl von Abfragen stellen lässt und die Daten somit auf mehrere Wege analysiert werden können, ist der Datenbanktyp besser für die Datenspeicherung in vernetzten Produktionssystemen gedacht. Des Weiteren sind mit diesem Kriterium die Möglichkeiten gemeint, die die jeweilige Datenbank durch den zu speichernden Datentyp bietet. Komplexe Datentypen lassen sich vielfältiger verarbeiten als simple Strukturen (siehe Abschnitt 3.4.5). Beispielsweise können sich aus mehreren Schlüssel-Wert-Paaren keine Beziehungen ableiten (siehe Abschnitt 3.4.2). Dieses Kriterium ist für die Datenspeicherung von Relevanz und wird deshalb in die Nutzwertanalyse übernommen.

Konsistenzsicherung

NoSQL-Datenbanken besitzen im Allgemeinen mit BASE ein lockereres Konsistenzmodell als relationale Datenbanken. Dies ist erforderlich, um eine massive Verteilung auf mehrere Rechner möglich zu machen (siehe Abschnitt 3.4.1). Allerdings ist die Konsistenz immer noch eine wichtige Eigenschaft bei der Datenspeicherung und soll auch in dieser Analyse nicht außen vor gelassen werden. Als Beispiel soll die autonome Warenanlieferung von fahrerlosen Transportsystemen dienen. Wenn eine sich selbst steuernde Anlage Bedarf an neuen Bauteilen meldet und eines der zahlreich eingesetzten Transportsysteme den Auftrag annimmt, da es gerade keinen anderen Auftrag abarbeitet, könnte es bei inkonsistenter Datenführung passieren, dass noch ein zweites Transportsystem den Auftrag annimmt, da der Auftrag aus der Sicht dieses System noch nicht von dem ersten Transportsystem angenommen wurde (siehe Abschnitt 3.4.1). Im Endeffekt muss eines der beiden Systeme die Bauteile wieder ins Lager bringen. Es ist erkennbar, dass inkonsistente Daten möglichst zu vermeiden sind, weshalb das Kriterium der Konsistenz in die Nutzwertanalyse mit aufgenommen wird.

Leistungsfähigkeit (reads)

Dieses Kriterium beschreibt die Fähigkeit einer Datenbank, Leseanfragen performant und schnell zu verarbeiten. Da bei den häufigsten Anwendungsfällen von Big Data-Problemen die Daten überwiegend gelesen und nur selten verändert werden (siehe Abschnitt 3.3), ist dieses Kriterium immens wichtig für die Datenspeicherung in vernetzten Produktionssystemen (siehe Abschnitt 3.4.1). Wenn beispielsweise ein autonomer Roboter, der Schweißprozesse ausführt und ständig seine Position abfragt, eine verzögerte Antwort bekommt kann es passieren, dass die Schweißnaht falsch gesetzt wird und der

Fehler danach korrigiert oder das betroffene Bauteil im schlimmsten Fall entsorgt werden muss. Deshalb gehört die Leistungsfähigkeit, Leseanfragen performant zu beantworten, zu den Kriterien der Nutzwertanalyse.

Leistungsfähigkeit (writes)

Die Leistungsfähigkeit, Schreibanfragen in einer hohen Geschwindigkeit zu beantworten, ist ebenfalls sehr wichtig für die Datenspeicherung in vernetzten Produktionssystemen. Auch wenn diese Art von Anfragen nicht so häufig vorkommt, wie die der Leseanfragen, muss sie trotzdem möglichst hoch sein (siehe Abschnitt 3.3). Ist diese Eigenschaft nicht gegeben, entsteht eine Differenz zwischen den realen und digitalen Daten. Wenn Daten nicht mehr möglichst echtzeitnah sind, können die Prozesse in einer vernetzten Produktion nicht mehr effektiv ausgeführt werden (siehe Abschnitt 2.2). Außerdem erhöht sich durch die längere Bearbeitungszeit einer Schreibanfrage die Wahrscheinlichkeit eines write-write Konflikts (siehe Abschnitt 3.4.1), der permanente Auswirkungen auf das System hat. Die Leistungsfähigkeit, Schreibanfragen performant zu beantworten, wird in den Kriterienkatalog für die Nutzwertanalyse aufgenommen.

Realisierung von Beziehungen

Es existieren viele Anwendungsfälle, bei denen die anfallenden Daten in Beziehungen zueinander stehen (siehe Abschnitt 3.4.5). So müssen fahrerlose Transportsysteme beispielshalber die Position der anderen FTS aus dem Schwarm kennen, um nicht mit ihnen zusammenzustoßen (siehe Abschnitt 2.3). Um ein weiteres Beispiel zu nennen, könnten sie auf Basis der Auftragslage der anderen FTS entscheiden, ob sie einen Auftrag annehmen oder nicht. Deshalb wird bei der Nutzwertanalyse auf dieses Kriterium eingegangen.

Schemaflexibilität

Das Kriterium der Schemaflexibilität von Datenbanken bezieht sich auf die unterschiedlichen Datentypen, die in einer Datenbank abgespeichert werden können (siehe Abschnitt 3.4.1). Da eine Eigenschaft von Big Data die Variety ist, also das Vorkommen unterschiedlicher Datenformate und -modelle, die sich jeweils in strukturierte, semistrukturierte und unstrukturierte Daten einteilen lassen, besteht der Bedarf nach einer hohen Schemaflexibilität (siehe Abschnitt 3.3). Zum Beispiel sollen in einer Datenbank zur Qualitätskontrolle Prüfberichte und im Falle eines Fehlers auch Bilddateien abgespeichert werden. Dies lässt sich nur erreichen, wenn sich verschiedene Datenstrukturen auf der Datenbank hinterlegen lassen. Aus diesem Grund wird die Schemaflexibilität von Datenbanken in die Nutzwertanalyse übernommen.

Skalierbarkeit (reads)

Dieses Kriterium beschreibt die Möglichkeit der Skalierbarkeit, um Leseanfragen zu verbessern. Dies kann entweder durch das Sharding und die Replikation mit der Master-Slave- und der Peer-to-Peer-Methode erreicht werden (siehe Abschnitt 3.4.1). Je größer der Grad der Skalierung ist, desto mehr Elemente können beispielsweise in ein vernetztes Produktionssystem integriert werden. Da bei den meisten Anwendungen von Big Data Leseanfragen gestellt werden, ist dieses Kriterium eminent für die Beschreibung von verschiedenen Datenbanktypen.

Skalierbarkeit (writes)

Im Gegensatz zum letzten Kriterium beschäftigt sich dieses Kriterium mit der Möglichkeit der Skalierbarkeit, um Schreibanfragen zu verbessern. Dafür können die Verteilungsmodelle des Shardings und der Replikation nach dem Peer-to-Peer-Verfahren eingesetzt werden (siehe Abschnitt 3.4.1). Es können beispielsweise in einem vernetzten Produktionssystem mehr Sensoren eingesetzt werden, die ihren Zustand laufend aktualisieren. Dieses Kriterium wird ebenfalls in den Katalog für die Nutzwertanalyse übernommen.

Verbreitung

Aus der Verbreitung eines Datenbanktyps könnten Rückschlüsse auf dessen Mächtigkeit gezogen werden. Wenn ein gewisser Typ öfter benutzt wird als ein anderer, hat dies eventuell die Ursache, dass er besser für eine Anwendung geeignet ist. Allerdings ist es wahrscheinlicher, dass sich damit die Anwendungsfälle identifizieren lassen, welche am häufigsten vorkommen, da jeder Datenbanktyp für einen unterschiedlichen Anwendungsfall entwickelt wurde (siehe Abschnitt 3.5). Hinzu kommt, dass beispielsweise jedes Produktionssystem unterschiedlich groß ist und verschiedene Anforderungen an eine Datenbank stellt (siehe Abschnitt 2.2). Aus diesen Gründen wird die Verbreitung nicht in der Nutzwertanalyse verwendet.

Verfügbarkeit

Das Kriterium der Verfügbarkeit bezieht sich auf die ununterbrochene Bereitschaft eines Systems. Kann ein Datenbanksystem zu einem Zeitpunkt eine Anfrage nicht beantworten oder zeigt bei der Anfrage keine Reaktion, so ist es nicht verfügbar (siehe Abschnitt 3.4.1). Die Relevanz dieses Kriteriums wird durch das Beispiel deutlich, dass selbststeuernde Anlagen wissen müssen, welche Prozessschritte sie an dem nächsten Fahrzeug auszuführen haben. Werden ihre Anfragen nicht beantwortet kann es zum Produktionsstopp kommen, der den schlechtesten erdenklichen Fall beschreibt. Die Verfügbarkeit wird in der Nutzwertanalyse berücksichtigt.

Nachdem alle Kriterien beschrieben und auf ihre Relevanz für die Nutzwertanalyse geprüft wurden, sollen die ausgewählten Kriterien zur Übersichtlichkeit kurz zusammengefasst werden:

- Ausfalltoleranz
- Funktionsumfang
- Konsistenzsicherung
- Leistungsfähigkeit (reads)
- Leistungsfähigkeit (writes)
- Realisierung von Beziehungen
- Schemaflexibilität
- Skalierbarkeit (reads)
- Skalierbarkeit (writes)
- Verfügbarkeit

Der nächste Schritt in der Erstellung der Methode ist die Bewertung des Erfüllungsgrades der einzelnen Datenbanktypen in Hinblick auf die ausgewählten Kriterien.

4.3.2 Bewertung der Kriterien

In Kapitel 4.3.1 wurden die Kriterien so festgelegt, dass sie die verschiedenen Modelle möglichst gut beschreiben. Um die unterschiedlichen Ausprägungen der Datenbanken in den definierten Kriterien quantitativ messbar zu machen (siehe Abschnitt 4.1) werden die verschiedenen NoSQL-Datenbanktypen in die Kriterien eingeordnet. Dabei werden für den jeweiligen Grad der Erfüllung eines Kriteriums bestimmte Punktzahlen vergeben. Nach Haag et al. (vgl. 2011, S. 329) wird für vage Aussagen eine Skala mit drei Erfüllungsgraden verwendet, bei genaueren Informationen eine Skala mit fünf Erfüllungsgraden und im Ausnahmefall wird auch eine Skala mit zehn Erfüllungsgraden eingesetzt. Es ist dabei zu beachten, dass die Zuteilung eines Erfüllungsgrades „mit einem relativ großen subjektiven Bewertungsspielraum erfolgt“ (Haag et al. 2011, S. 329). Da die Kriterien qualitativ sind, wird auf die Genauigkeit der Skala mit zehn Erfüllungsgraden verzichtet. Allerdings wurde durch die Aufarbeitung der Theorie in Kapitel 3 die fundierte Grundlage für eine Skala mit fünf Erfüllungsgraden gelegt. Die verbale Beschreibung der Erfüllungsgrade mit der dazugehörigen Punktzahl ist in Tabelle 4.1 verdeutlicht.

Tabelle 4.1: Zuordnung von Erfüllungsgrad und Punktzahl

Erfüllungsgrad des Bewertungskriteriums	Punkte
Besonders niedrig	1
Niedrig	2
Neutral	3
Hoch	4
Besonders hoch	5

Im Folgenden werden die Merkmalsausprägungen der vier NoSQL-Datenbanktypen in Bezug auf die in Kapitel 4.3.1 identifizierten Kapitel bestimmt. Dabei basieren die Einordnungen auf den Erkenntnissen aus Kapitel 2 und 3 und sie werden für jedes Kriterium verbal hergeleitet und unter jedem Kriterium der Übersicht halber in einer Tabelle festgehalten. Aus Platzgründen werden die Schlüssel-Wert-Datenbanken zu „Schlüssel-Wert“, die Spaltenfamilien-Datenbanken zu „Spaltenfamilien“, die Dokument-Datenbanken zu „Dokument“ und die Graphdatenbanken zu „Graph“ abgekürzt. Die Punktzahlen 1 und 5 werden lediglich für einen besonders hohen, bzw. niedrigen Erfüllungsgrad verwendet. Dadurch soll sichergestellt werden, dass die maximale, bzw. minimale Ausprägung eine Besonderheit des Datenbanktyps darstellt. Es sei angemerkt, dass auf Basis der Theorie in Abschnitt 3.4.1 die Ausprägung eines Merkmals hergeleitet werden kann, beispielsweise mit dem CAP-Theorem, wenn in der Literatur keine Angabe zu diesem Kriterium gefunden wird.

Ausfalltoleranz

Der Erfüllungsgrad bei diesem Kriterium unterscheidet sich durch den Einsatz verschiedener Distributionsmodelle, die eine unterschiedliche Wirkung auf die Ausfalltoleranz haben (siehe Abschnitt 3.4.1) Durch das bei den meisten Schlüssel-Wert-Datenbanken eingesetzte Sharding kann die Ausfalltoleranz nicht erhöht werden. Wenn ein Rechner ausfällt, sind die darauf befindlichen Daten nicht mehr zugänglich. (siehe Abschnitt 3.4.2) Spaltenfamilien-Datenbanken setzten das Sharding häufig in Kombination mit der Peer-to-Peer Replikation ein. Diese Verbindung garantiert besonders hohe Ausfalltoleranz. (siehe Abschnitt 3.4.3) Bei den Dokument-Datenbanken wird die Replikation mit dem Master-Slave-Verfahren eingesetzt. Wenn einer der Master ausfallen sollte, wird der neue Master unter den verbliebenen Slaves ausgewählt und somit eine hohe Ausfalltoleranz bereitgestellt. (siehe Abschnitt 3.4.4) Die Graphdatenbanken nutzen das gleiche Distributionsmodell wie die Dokument-Datenbanken. (siehe Abschnitt 3.4.5) Allerdings ist die Ausfalltoleranz nicht als gleichwertig einzustufen, da die Graphdatenbanken bereits in den Kriterien der Konsistenzsicherung und der Verfügbarkeit sehr gute Erfüllungsgrade besitzen und nach dem CAP-Theorem nicht alle Werte gleichzeitig werden können.

(siehe Abschnitt 3.4.1). Zusammenfassend ist festzuhalten, dass Spaltenfamilien-Datenbanken das Kriterium der Ausfalltoleranz in einem besonders hohen Maße erfüllen und danach die Dokument-Datenbanken kommen, die aus genannten Gründen höher einzustufen sind, als die Graphdatenbanken. Das Schlusslicht bilden die Schlüssel-Wert-Datenbanken. Es ergibt sich folgende Punkteverteilung für das Kriterium der Ausfalltoleranz (Tabelle 4.2).

Tabelle 4.2: Punkteverteilung für die Ausfalltoleranz

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
2	5	4	3

Funktionsumfang

Die verschiedenen NoSQL-Datenbanken lassen sich anhand ihres Funktionsumfangs unterscheiden. Schlüssel-Wert-Datenbanken besitzen lediglich die drei Grundbefehle PUT, GET und DELETE und sind deshalb in ihren Funktionen limitiert (siehe Abschnitt 3.4.2). Spaltenfamilien-Datenbanken sind den relationalen Datenbanken sehr ähnlich und besitzen deshalb einen vergleichbaren Funktionsumfang (siehe Abschnitt 3.4.3). Die Dokument-Datenbanken werden als mächtigste Typen der NoSQL-Datenbanken beschrieben und besitzen im Gegensatz zu den Schlüssel-Wert-Datenbanken den entscheidenden Vorteil, dass sie bessere Indexierungsmöglichkeiten zum Selektieren und Filtern von Datensätzen benutzen. Aus diesem Grund verfügen sie über einen besonders hohen Erfüllungsgrad. (siehe Abschnitt 3.4.4) Die meisten Werte aus Graphdatenbanken werden aus den Beziehungen zwischen den Knoten abgeleitet. Allerding sind sie nicht für analytische Abfragen geeignet, die über viele Knoten und Attribute hinweg gehen. Normalerweise würden ihnen dadurch ein niedriger Erfüllungsgrad zugeordnet, doch durch die besondere Eigenschaft, Beziehungen verarbeiten zu können, heben sie sich von den Schlüssel-Wert-Datenbanken ab. (siehe Abschnitt 3.4.5) Somit besitzen die Dokument-Datenbanken die größte Mächtigkeit, gefolgt von den Spaltenfamilien-, den Graph- und zuletzt den Schlüssel-Wert-Datenbanken. Für den Funktionsumfang der NoSQL-Core-Modelle sieht die Punkteverteilung wie folgt aus (Tabelle 4.3).

Tabelle 4.3: Punkteverteilung für den Funktionsumfang

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
2	4	5	3

Konsistenzsicherung

Im Allgemeinen leidet die Erfüllung der Konsistenz durch die Anforderung nach einer verteilten Datenhaltung. Die referentielle Integrität wird bei Schlüssel-Wert-Datenbanken nicht unterstützt. Eine

Konsistenz ist nicht gegeben. Durch die Verteilung auf shards, ist es nicht möglich, sich mit anderen Beständen zu synchronisieren. (siehe Abschnitt 3.4.2) Spaltenfamilien-Datenbanken eignen sich nicht für Anwendungsfälle, in denen Konsistenz gefordert wird (siehe Abschnitt 3.4.3). Da beim Typ der Dokument-Datenbanken die Ausfalltoleranz und die Verfügbarkeit einen hohen Erfüllungsgrad besitzen, kann die Konsistenzsicherung nach dem CAP-Theorem nicht erreicht sein (siehe Abschnitt 3.4.1). Im Gegenteil zu den anderen Datenbanktypen bleiben gespeicherte Daten in Graphdatenbanken durch Transaktionen konsistent. Die Verwendung der Transaktionen stellt sicher, dass jede Beziehung einen Start- und einen Endpunkt besitzt. Außerdem kann ein vorhandener Knoten nur gelöscht werden, wenn keine Bedingungen mehr an ihm anliegen. (siehe Abschnitt 3.4.5) Aus diesem Grund erfüllen die Graphdatenbanken das Kriterium der Konsistenz am meisten. Den anderen Datenbanken werden aufgrund ihrer Eigenschaft der massiv verteilten Datenhaltung ein niedrigerer Erfüllungsgrad zugeordnet. Es ist anzumerken, dass das Datenbanksystem durch das BASE-Prinzip schlussendlich Konsistenz erreicht (siehe Abschnitt 3.4.1). Die Punkteverteilung für das Kriterium der Konsistenzsicherung sieht folgendermaßen aus (Tabelle 4.4).

Tabelle 4.4: Punkteverteilung für die Konsistenzsicherung

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
3	3	3	5

Leistungsfähigkeit (reads)

Die Leistungsfähigkeit, Leseanfragen schnell zu beantworten, wird bei den Datenbanktypen durch unterschiedliche Faktoren beeinflusst. Da bei Schlüssel-Wert-Datenbanken die referentielle Integrität nicht geprüft wird, können größere Datenmengen effizienter verarbeitet werden. Des Weiteren ist es möglich, eine In-Memory-Datenbank für die Ausführung des Datenbanksystems zu verwenden, wodurch die Performanz der Lese- und Schreibanforderungen nochmals um ein Vielfaches steigern. (siehe Abschnitt 3.4.2) Die Spaltenfamilien-Datenbanken sind im Allgemeinen für Anwendungen zu gebrauchen, die eine hohe Leistung der Lese- und Schreiboperationen voraussetzen (siehe Abschnitt 3.4.3). Die Dokument-Datenbanken sind in der Bearbeitung von Leseanforderungen sehr performant, da ihr Fokus auf der Verarbeitung von großen Datenmengen liegt. Sie werden durch die Replikation mit dem Master-Slave-Verfahren verteilt, lassen sich alle Leseoperationen durch die Slaves beantworten. Außerdem werden bei diesem Datenbanktypen die Abfragen mithilfe des MapReduce-Verfahrens parallelisiert bearbeitet, was zu einer zusätzlichen Verbesserung der Leistungsfähigkeit führt. (siehe Abschnitt 3.4.4) Zu den Eigenschaften der Graphdatenbanken zählen die einfache Abfrage und Analyse von Beziehungen (siehe Abschnitt 3.4.5). Es lässt sich verallgemeinern, dass alle NoSQL-Datenbanktypen eine hohe Leistungsfähigkeit in Bezug auf die Bearbeitung von Leseanfragen besitzen. Die Typen

der Schlüssel-Wert- und der Dokument-Datenbanken sind durch ihre Fähigkeit der Verarbeitung auf einer In-Memory-Datenbank, bzw. durch den Einsatz von MapReduce-Verfahren performanter als die Alternativen. In Tabelle 4.5 ist die Punktevergabe für die Leistungsfähigkeit bezogen auf die Leseanfragen zusammengefasst.

Tabelle 4.5: Punkteverteilung für die Leistungsfähigkeit (reads)

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
5	4	5	4

Leistungsfähigkeit (writes)

Zur Bewertung der Leistungsfähigkeit, Schreibanfragen zeitnah zu verarbeiten, werden ähnliche Faktoren eingesetzt, wie schon bei der Leistungsfähigkeit bezüglich der Leseanfragen. Bei Schlüssel-Wert-Datenbanken lassen sich große Datenmengen effizient verarbeiten, da die referentielle Integrität nicht überprüft wird. Mit der Datenspeicherung auf dem Hauptspeicher wird die Performanz zusätzliches um ein Vielfaches gesteigert. (siehe Abschnitt 3.4.2) Die Typen der Spaltenfamilien-Datenbanken werden bei Anwendungen eingesetzt, die eine hohe Performanz bei Schreib- und Leseoperationen voraussetzen. (siehe Abschnitt 3.4.3) Der Fokus von Dokument-Datenbanken liegt zwar auf dem Fokus einer Bearbeitung von großen Mengen von heterogenen Daten (siehe Abschnitt 3.4.4), jedoch sind die Schreibanfragen durch die Existenz eines Master-Servers limitiert (siehe Abschnitt 3.4.1). Die Bearbeitung von Schreibanfragen bei Graphdatenbanken ist wie schon bei den Dokument-Datenbanken durch einen Master-Server limitiert. Allerdings lassen sich die Beziehungen zwischen den Knoten einfach analysieren. (siehe Abschnitt 3.4.5) Die Erfüllungsgrade für die Leistungsfähigkeit, Schreibanfragen effizient zu verarbeiten, sind für die Schlüssel-Wert- und die Spaltenfamilien-Datenbanken unverändert. Bei den Dokument- und Graphdatenbanken ergibt sich durch das Distributionsmodell der Replikation nach dem Master-Slave-Verfahren eine Einschränkung der Leistungsfähigkeit. Aus Tabelle 4.6 sind die Punkte für die Leistungsfähigkeit der Bearbeitung von Schreibanfragen zu entnehmen.

Tabelle 4.6: Punkteverteilung für die Leistungsfähigkeit (writes)

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
5	4	3	3

Realisierung von Beziehungen

Bei den meisten NoSQL-Datenbankentypen lassen sich keine Beziehungen zwischen den Daten realisieren (siehe Abschnitt 3.4.1). Schlüssel-Wert-Datenbanken stoßen bei der Verarbeitung von Daten, die untereinander verbunden sind, an ihre Grenzen (siehe Abschnitt 3.4.2). Spaltfamilien-Datenbanken

ähneln den relationalen Datenbanken und haben deshalb bedingt die Möglichkeit, Beziehungen zwischen Daten zu speichern (siehe Abschnitt 3.4.3). Dokument-Datenbanken grenzen sich von den Schlüssel-Wert-Datenbanken durch den großen Unterschied ab, dass sie gespeicherte Daten strukturieren können. Außerdem können einfache Beziehungen durch Verweise auf die Schlüsselwerte von Dokumenten erstellt werden. (siehe Abschnitt 3.4.4) Die Graphdatenbanken zeichnen sich vor allem durch ihre Eigenschaft aus, Beziehungen einfach darzustellen und abzufragen. Ihre Struktur ist darauf aufgebaut, Beziehungen zwischen verschiedenen Knoten darzustellen. (siehe Abschnitt 3.4.5) Deshalb eignen sie sich in besonderem Maße für die Realisierung von Beziehungen. Aber auch die Dokumentdatenbanken erreichen durch die Möglichkeit zur Datenstrukturierung und zur Erstellung von Verweisen einen guten Erfüllungsgrad. Spaltfamilien haben durch ihre Ähnlichkeit zu relationalen Datenbanken ebenfalls die Chance, eine Beziehung zwischen den Daten herzustellen. Im Vergleich zu diesen drei Datenbanktypen ist es den Schlüssel-Wert-Datenbanken nicht möglich, in Beziehung stehende Daten zu verarbeiten. Aus Tabelle 4.7 lassen sich die verteilten Punkte bezüglich der Realisierungsmöglichkeit von Beziehungen ablesen.

Tabelle 4.7: Punkteverteilung für die Realisierung von Beziehungen

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
1	3	4	5

Schemaflexibilität

Für das Kriterium der Schemaflexibilität müssen die verschiedenen Datenbanken differenziert betrachtet werden. Schlüssel-Wert-Datenbanken nehmen im Normalfall nicht die Form von komplexen Datentypen an, obwohl die Datenobjekte prinzipiell nicht in ihrem Typ begrenzt werden (siehe Abschnitt 3.4.2). Die einzigen Schemaregeln, die für die Spaltenfamilien-Datenbanken existieren, betrifft die Spaltenfamilien selbst. Soll dem Schema eine neue Spaltenfamilie hinzugefügt werden, lässt sich dies lediglich durch die Änderung des Schemas umsetzen. Allerdings bietet diese Datenbanktechnologie dadurch einen sicheren Umgang mit Szenarien, die nicht vorhersehbar sind. (siehe Abschnitt 3.4.3) Die Dokument-Datenbank wird als der flexibelste NoSQL-Datenbankttyp bezeichnet. Erreicht wird dies über die Tatsache, dass abgespeicherte Dokumente unabhängig voneinander sind. Die Daten können deshalb große strukturelle Unterschiede aufweisen. (siehe Abschnitt 3.4.4) Der Typ der Graphdatenbanken ist ebenfalls flexibel in seinem Schema, da er die effektive Einbindung von neuen Knoten und Kanten unterstützt (siehe Abschnitt 3.4.5). Dokument-Datenbanken erfüllen das Kriterium der Schemaflexibilität in besonderem Maße und sind somit die flexibelsten NoSQL-Datenbanktypen. Es folgen die Graphdatenbanken, die ebenfalls flexibel in der Speicherung unterschiedlicher Datenstrukturen sind. Dahinter werden der Schlüssel-Wert-Datenbanken mit ihrer Speicherung überwiegend einfacher

Daten und der Spaltenfamilien-Datenbanken mit der Notwendigkeit der Änderung des Schemas, falls eine neue Spalte eingefügt wird, der gleiche Erfüllungsgrad zugeordnet. In Tabelle 4.8 sind die Punkte für das Kriterium der Schemaflexibilität des Datenbanktyps zusammengefasst.

Tabelle 4.8: Punkteverteilung für die Schemaflexibilität

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
3	3	5	4

Skalierbarkeit (reads)

Die Skalierbarkeit bezüglich Leseanfragen unterscheidet sich nach der Art des verwendeten Distributionsmodells (siehe Abschnitt 3.4.1). Die am meisten verbreitetste Methode bei Schlüssel-Wert-Datenbanken ist das Sharding. Dieses Modell wird verwendet, um die Anzahl der möglichen Lese- und Schreibanfragen gleichermaßen zu erhöhen. (siehe Abschnitt 3.4.2) Bei Spaltenfamilien-Datenbanken kommt eine Kombination aus Sharding und Replikation nach dem Peer-to-Peer-Verfahren zum Einsatz. Dadurch werden die Lese- und Schreibanfragen ebenfalls gleichermaßen skaliert. (siehe Abschnitt 3.4.3) Die Daten werden bei Dokument-Datenbanken nach dem Master-Slave-Verfahren auf viele Rechner repliziert. Als Resultat können eine größere Anzahl Leseanfragen bearbeitet werden, da eine Vielzahl von Slave-Servern existieren, die diese beantworten. (siehe Abschnitt 3.4.1 und 3.4.4) Die Graphdatenbanken nutzen die Replikation nach dem Master-Slave-Modell, weil der Datensatz des Graphen zu groß für den Arbeitsspeicher ist (siehe Abschnitt 3.4.5). Die verschiedenen Datenbanktypen lassen sich folglich mit dem gleichen Erfolg bezüglich Leseanfragen skalieren. Lediglich die Schlüssel-Wert-Datenbank eignet sich besonders gut, da sie sich sehr effektiv durch Sharding verteilen lässt. Für das Kriterium der Skalierbarkeit für Leseanfragen sind die Punkte in Tabelle 4.9 zusammengefasst.

Tabelle 4.9: Punkteverteilung für die Skalierbarkeit (reads)

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
5	4	4	4

Skalierbarkeit (writes)

Für die Skalierbarkeit bezüglich Schreibanfragen ist ebenfalls die Art des verwendeten Distributionsmodells entscheidend. Schlüssel-Wert-Datenbanken nutzen zur Verteilung das Modell des Shardings. Dabei werden die Lese- und Schreibanfragen gleichermaßen skaliert. (siehe Abschnitt 3.4.2) Die Spaltenfamilien-Datenbanken nutzen eine Kombination aus Sharding und Replikation nach dem Peer-to-Peer-Verfahren. Bei dieser Kombination erhöht sich die Anzahl der möglichen Lese- und Schreibanfragen ebenfalls gleichermaßen. (siehe Abschnitt 3.4.3) Bei Dokument-Datenbanken kommt die

Replikation nach dem Master-Slave-Verfahren zum Einsatz. Dieses Verfahren limitiert die Skalierbarkeit der Schreibanfragen durch den Master-Server. (siehe Abschnitt 3.4.1 und 3.4.4) Wenn der Graph einer Graphdatenbank auf mehreren Rechnern verteilt liegen würde, geht das Durchlaufen von diesem zu Lasten der Leistungsfähigkeit. Das Sharding eines Graphen wird von heutigen Datenbanken nicht unterstützt, da sich der Graph nicht effizient in mehrere Teilgraphen zerlegen lässt. Aus diesem Grund wird die Graphdatenbank lediglich nach dem Master-Slave-Modell repliziert, wodurch die Schreibanfragen durch den Master limitiert sind. Eine Skalierbarkeit bezüglich Schreibanfragen kann deshalb nicht erreicht werden. (siehe Abschnitt 3.4.1 und 3.4.5) Abhängig von der Wahl der Verteilungsmodells kann ein Datenbanktyp unterschiedlich gut geeignet für die Skalierbarkeit bezüglich Schreibanfragen geeignet sein. Durch die Replikation nach dem Master-Slave-Verfahren kann keine bessere Skalierbarkeit der Schreibanfragen erreicht werden. Beim Sharding ist dies nicht der Fall, da die Schreibanfragen durch dieses Verfahren auf mehrere Rechner verteilt werden können. Die Schlüssel-Wert-Datenbanken sind dafür besonders gut geeignet, da sie sich sehr einfach sharden lassen. Aus Tabelle 4.10 lassen sich die verteilten Punkte für die Skalierbarkeit der Schreibanfragen entnehmen.

Tabelle 4.10: Punkteverteilung für die Skalierbarkeit (writes)

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
5	4	3	3

Verfügbarkeit

Die Verfügbarkeit der Daten ist ebenfalls abhängig vom verwendeten Distributionsmodell. Bei den Schlüssel-Wert-Datenbanken leidet sie unter der Methode des Shardings. Wenn ein Rechner ausfällt, sind die darauf befindlichen Daten nicht mehr verfügbar. (siehe Abschnitt 3.4.2) Das Cluster einer Spaltenfamilien-Datenbank hat durch die Replikation nach dem Peer-to-Peer-Verfahren keinen Master, sondern alle Knoten nehmen die Rolle eines Peers ein. Dadurch ist die Verfügbarkeit der Daten sehr hoch (siehe Abschnitt 3.4.1 und 3.4.3) Die Verfügbarkeit von Dokument-Datenbanken wird durch das Replizieren nach dem Master-Slave-Verfahren erhöht. Der Ausfall eines Slaves hat keine Auswirkungen auf die Datenverfügbarkeit, da es mehrere Slaves in einem Cluster gibt. Auch der Ausfall eines Masters hat keine negativen Auswirkungen auf die Verfügbarkeit, da einer der Slaves die Rolle des Masters einnehmen kann. (siehe Abschnitt 3.4.1 und 3.4.4) Gleiches gilt für die Graphdatenbanken, die ebenfalls auf die Verteilung durch Replizieren mit dem Master-Slave-Modell setzt. (siehe Abschnitt 3.4.5) Schlüssel-Wert-Datenbanken besitzt durch ihr Verteilungsmodell im Gegensatz zu den anderen Datenbanktypen keine hohe Datenverfügbarkeit. Im Vergleich der beiden Replikationsverfahren Master-Slave und Peer-to-Peer, besitzt das Peer-to-Peer-Verfahren eine besonders hohe Verfügbarkeit. Die Punkteverteilung für die Verfügbarkeit ist in Tabelle 4.11 zusammengefasst.

Tabelle 4.11: Punkteverteilung für die Verfügbarkeit

Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
2	5	4	4

Die ermittelten Erfüllungsgrade sollen in Tabelle 4.12 zusammengefasst dargestellt werden.

Tabelle 4.12: Zusammengefasste Punkteverteilungen

	Schlüssel-Wert	Spaltenfamilien	Dokument	Graph
Ausfalltoleranz	2	5	4	3
Funktionsumfang	2	4	5	3
Konsistenzsicherung	3	3	3	5
Leistungsfähigkeit (reads)	5	4	5	4
Leistungsfähigkeit (writes)	5	4	3	3
Realisierung von Beziehungen	1	3	4	5
Schemaflexibilität	3	3	5	4
Skalierbarkeit (reads)	5	4	4	4
Skalierbarkeit (writes)	5	4	3	3
Verfügbarkeit	2	5	4	4

4.3.3 Gewichtung der Kriterien

Nach der Bewertung der unterschiedlichen NoSQL-Datenbanktypen in Kapitel 4.3.2, sind die Bewertungskriterien nach ihrer Relevanz einzuordnen, um eine differenzierte Aussage über die verschiedenen Technologien zu erhalten. Für diese Gewichtung wird die 100-Punkte-Methode verwendet. Dabei werden 100 Punkte so auf die in Kapitel 4.3.1 festgelegten Bewertungskriterien verteilt, dass dem wichtigsten Kriterium die meisten Punkte zugeteilt werden und dem unrelevantesten die wenigsten Punkte. (vgl. Kunze 1969, S. 56) Die Festlegung der Gewichtung erfolgt nach Haag et al. (vgl. 2011, S. 329) ebenso wie die Zuordnung von Erfüllungsfaktoren auf Basis von Erfahrungen, weshalb der Bewertungsspielraum relativ groß und subjektiv ist. Die Gewichtung der Bewertungskriterien erfolgt bezüglich der Anforderungen von vernetzten Produktionssystemen in der Automobilindustrie (siehe Kapitel 2.2 und 2.3) und den Herausforderungen der Industrie 4.0 (siehe Kapitel 2.1) und Big Data (siehe Kapitel 3.3). Es ist darauf hinzuweisen, dass sich die folgende Gewichtung auf die allgemeinen Anforderungen der genannten Themenbereiche bezieht. Aus diesem Grund ist vor jeder Anwendung der Methode

auf einen bestimmten Anwendungsfall zu überprüfen, ob die Relevanz der Bewertungskriterien hinsichtlich der Anforderungen des entsprechenden Anwendungsfalls verändert werden müssen.

Die Automobilindustrie setzt auf den Einsatz von vernetzten Produktionssystemen (siehe Abschnitt 2.3), da die maximale Effektivität und Effizienz eines Produktionssystems nur durch die verlustfreie Operation und Kooperation der eingesetzten Leistungseinheiten erreicht werden kann (siehe Abschnitt 2.2). Jedoch resultiert aus einem hohen Vernetzungsgrad nicht nur der steigende Wert eines Netzwerkes, sondern auch die Anforderungen nach leistungsstärkeren Rechnern, um die anfallenden Datenmengen effektiv verarbeiten zu können. (siehe Abschnitt 2.2) Die im Produktionssystem entstehenden Daten werden auch durch den Begriff Big Data beschrieben und haben die Eigenschaften, dass sie sehr umfangreich sind, oftmals keine Struktur aufweisen und aus unterschiedlichen Quellen stammen. Hinzu kommt, dass diese Daten möglichst in Echtzeit verarbeitet werden müssen (siehe Abschnitt 3.3). Zur Lösung von diesen Big Data-Problemen werden NoSQL-Datenbanksysteme eingesetzt, die sich durch eine massiv verteilte Datenhaltungsarchitektur auszeichnen, wodurch eine sehr hohe Leistungsfähigkeit erreicht wird und deren Fokus auf der Bearbeitung von Realtime-Anwendungen liegt. (siehe Abschnitt 3.4.1) Da die Daten bei den meisten Big Data-Anwendungen überwiegend gelesen werden und selten verändert werden, müssen insbesondere Leseanfragen schnell verarbeitet werden (siehe Abschnitt 3.3). Dies wird erreicht durch eine hohe Leistungsfähigkeit einer Datenbank, Leseanfragen beantworten zu können und durch die Möglichkeit, die Daten so zu skalieren, dass sich Leseanfragen performanter bedienen lassen (siehe Abschnitt 3.4.1). Die Anforderungen nach einer leistungsstarken Bearbeitung von Schreibanfragen ist für die meisten Big Data-Probleme allerdings nicht so wichtig, weshalb diesen Kriterien nicht allzu große Relevanz beigesteuert werden sollte. Da die Daten überwiegen gelesen werden, sind Funktionen zur Datenverarbeitung und -analyse nicht sehr bedeutsam. (siehe Abschnitt 3.3) Falls eine ausführliche Analyse der Daten gefordert ist, können relationale Datenbanken hinzugezogen werden. Da ein Konflikt zwischen der Konsistenzerhaltung von Daten und deren effizienten Verarbeitung existiert, wird sich nicht auf die Konsistenz von Daten fokussiert. Aus diesem Grund wird ein schwächeres Konsistenzmodell akzeptiert. (siehe Abschnitt 3.4.1) Weil die Daten aus unterschiedlichen Quellen anfallen und nicht einheitlich strukturiert sind, ist außerdem eine hohe Anforderung an die Schemaflexibilität der Datenbank gestellt (siehe Abschnitt 3.3). Durch die Eigenschaft der Vernetzung der Leistungseinheiten ergibt sich der Bedarf nach der Fähigkeit, Beziehungen in den Datenbanken realisieren zu können (siehe Abschnitt 3.4.1). Ebenfalls von großer Bedeutung sind die Kriterien der Ausfalltoleranz von Datenbanken und Verfügbarkeit der Daten, da die hohe Effizienz eines vernetzten Produktionssystem nur durch die verlustfreie Zusammenarbeit seiner Elemente möglich ist (siehe Abschnitt 2.2). Aus diesen Argumenten resultiert die Relevanz der einzelnen Bewertungskriterien, welche abhängig von ihrer Wichtigkeit eine Punktzahl zugeschrieben bekommen. Die Verteilung der 100 Punkte ist in Tabelle 4.13 dargestellt. Bei 100 zu verteilenden Punkten würde jedes

Kriterium im Falle einer gleichmäßigen Verteilung 10 Punkte bekommen. Demnach werden Kriterien, die für Datenbanken in vernetzten Produktionssystemen nicht besonders wichtig sind, sechs Punkte zugeteilt. Die wichtigen Kriterien bekommen dafür 12, bzw. 13 Punkte im Falle einer besonderen Relevanz.

Tabelle 4.13: Gewichtung durch Punkteverteilung auf die Kriterien

Kriterien	Gewichtung
Ausfalltoleranz	13
Funktionsumfang	6
Konsistenzsicherung	6
Leistungsfähigkeit (reads)	13
Leistungsfähigkeit (writes)	6
Realisierung von Beziehungen	12
Schemaflexibilität	12
Skalierbarkeit (reads)	13
Skalierbarkeit (writes)	6
Verfügbarkeit	13

4.3.4 Komplette Methode

Abschließend ist der Nutzen für jeden NoSQL-Datenbanktypen zu ermitteln, um die Auswirkungen der Bewertung aus Kapitel 4.3.2 deuten zu können. Der Nutzen einer Nutzwertanalyse ist das Ergebnis der Bewertung, das sich erst durch die Zerlegung einer komplexen Problematik ergibt (vgl. Kunze 1969, S. 58 f.). Durch diese Unterteilung ergibt sich der Wert des Gesamtnutzens aus der Summe der Werte der jeweiligen Teilnutzen (TNW), die durch Multiplikation von Gewichtsfaktor (G) des Kriteriums und Erfüllungsgrad (EG) des entsprechenden Datenbanktyps berechnet werden (vgl. Kunze 1969, S. 62). Abschließend werden die Datenbanktypen in eine Rangfolge eingeteilt, welche die zu bevorzugende Lösung für das vorliegende Problem angibt. Der Rang 1 wird dabei dem Datenbankmodell mit dem höchsten Nutzwert zugewiesen. Mit diesem Schritt ist die Nutzwertanalyse abgeschlossen. Der Entschluss, welcher Datenbanktyp ausgewählt wird, gehört nicht mehr zur Nutzwertanalyse. Sie hat die komplexe Problematik lediglich transparent gemacht und mit ihrer Beurteilung eine Grundlage geliefert, auf der sich eine Entscheidung treffen lässt. (vgl. Kunze 1969, S. 68 ff.)

Nachdem die Bewertungskriterien identifiziert (siehe Abschnitt 4.3.1) und gewichtet (siehe Abschnitt 4.3.3) wurden und Erfüllungsgrade für die Technologien beurteilt wurden (siehe Abschnitt 4.3.2), lassen sich die Ergebnisse der Nutzwertanalyse in einer Matrix (siehe Tabelle 4.14) zusammenfassen.

Tabelle 4.14: Nutzwertanalyse

		Schlüssel-Wert		Spaltenfamilien		Dokument		Graph	
Kriterien	G	EG	TNW	EG	TNW	EG	TNW	EG	TNW
Ausfalltoleranz	13	2	26	5	65	4	52	3	39
Funktionsumfang	6	2	12	4	24	5	30	3	18
Konsistenz-sicherung	6	3	18	3	18	3	18	5	30
Leistungsfähigkeit (reads)	13	5	65	4	52	5	65	4	26
Leistungsfähigkeit (writes)	6	5	30	4	24	3	18	3	18
Realisierung von Beziehungen	12	1	12	3	36	4	48	5	60
Schemaflexibilität	12	3	36	3	36	5	60	4	48
Skalierbarkeit (reads)	13	5	65	4	52	4	52	4	52
Skalierbarkeit (writes)	6	5	30	4	24	3	18	3	18
Verfügbarkeit	13	2	26	5	65	4	52	4	52
Nutzwert	100		320		396		413		361
Rangfolge		4		2		1		3	

Auf Basis dieser Ergebnisse lassen sich die Dokument-Datenbanken als favorisierte Alternative bestimmen. Das Resultat passt zu der Aussage von McCreary und Kelly (vgl. 2014, S. 86), dass Dokument-Datenbanken die flexibelsten, mächtigsten und verbreitetsten NoSQL-Datenbanktypen sind. Der Datenbanktyp mit dem zweitgrößten Nutzwert ist der der Spaltenfamilien-Datenbanken, gefolgt von den Graphdatenbanken. Den niedrigsten Nutzwert erreichen die Schlüssel-Wert-Datenbanken. Dieses Ergebnis bietet eine geeignete Entscheidungsgrundlage zur Auswahl des eingesetzten NoSQL-Datenbanktypes.

Es lassen sich einige Anmerkungen zu dieser Methode zusammenfassen. Zum können sich die Ergebnisse, wie schon in Abschnitt 4.3.2 und 4.3.3 erwähnt, je nach Zuordnung eines Erfüllungsgrades oder Gewichtung der Kriterien voneinander unterscheiden, da diese Schritte einen subjektiven

Bewertungsspielraum lassen (vgl. Haag et al. 2011, S. 329). Des Weiteren kann auch die Auswahl der Skala einen Einfluss auf das Endergebnis haben, da mit ihr eine genauere, bzw. ungenauere Unterteilung des Erfüllungsgrades möglich ist. Ein weiterer Punkt betrifft die Gewichtung der verschiedenen Bewertungskriterien. Wird diese Methode für einen Anwendungsfall benutzt, ist zuvor die Gewichtung der Kriterien mit den Anforderungen des jeweiligen Anwendungsfalls zu vergleichen, da eventuell Anpassungen nötig sind. Darüber hinaus ist diese Methode zur Überprüfung der Eignung von Datenbanktypen gedacht und nicht für eine bestimmte Software. Sollen verschiedene Tools verglichen werden, können alternative Bewertungsmethoden verwendet werden. Ein Bereich, der von dieser Methode nicht berücksichtigt wird, ist der der Polyglot Persistence. Bei diesem Verfahren ist es möglich, verschiedene Datenbanktypen und ihre Vorteile zu kombinieren, um somit ein Problem effektiver zu bearbeiten. (siehe Abschnitt 3.5) Dieses Verfahren sollte unbedingt bei der Entscheidungsfindung berücksichtigt werden.

5 Validierung der Methode

In diesem Kapitel soll sichergestellt werden, ob die entwickelte Methode den Anforderungen entspricht und sie sich auf selbst gewählte Praxisbeispiele anwenden lässt. Beim ersten Beispiel wird die erarbeitete Methode durch die Anwendung an ein Beispielsystem validiert. Das zweite Beispiel befasst sich mit einem Anwendungsfall, der eine Änderung der Gewichtung verlangt, da dieser Fall andere Anforderungen an Datenbanken hat. Es wird das Industriebeispiel aus Kapitel 2.3, bei dem fahrerlose Transportsysteme für den Transport von Bauteilen genutzt werden, aufgegriffen und mit eigenen Ideen verändert. Die Ergebnisse werden nach der Anwendung auf die Beispiele interpretiert.

5.1 Fahrzeugmontage

In einer Produktionshalle eines Automobilherstellers laufen dank der Einführung der Modulstrategie eine Vielzahl von Fahrzeugmodellen über das Montageband. Durch den Einsatz von zahlreichen selbststeuernden und selbstregelnden Anlagen, die untereinander verbunden sind und Daten bereitstellen, lesen und auswerten, ist das vorhandene Produktionssystem in hohem Grad vernetzt. (siehe Abschnitt 2.3) Es existiert kein papiergestützter Prozess, denn die Fahrzeuge sind durch den Einsatz von RFID-Technologie zu intelligenten Produkten geworden. Dadurch stellt ein Fahrzeug alle benötigten Daten, beispielsweise für einen Prozessschritt, zur richtigen Zeit am richtigen Ort bereit und speichert darüber hinaus Daten zu den absolvierten Arbeitsschritten, oder Testberichte von durchlaufenen Prüfzentren. Die Anlagen lesen diese Daten und können unverzüglich mit dem Bearbeitungsprozess anfangen. Damit der Mensch den Überblick über diese Prozesse behält, muss er in Echtzeit über die Vorgänge in der Montagehalle informiert werden. Dafür nutzt er beispielsweise ein Tablet, welches die Daten filtert und ihm die für ihn relevanten Informationen anzeigt. (siehe Abschnitt 2.2) Diese vernetzten Prozesse haben zur Folge, dass eine große Anzahl verschiedener Datentypen aus unterschiedlichen Quellen anfallen, die möglichst in Echtzeit verarbeitet werden sollen (siehe Abschnitt 3.3). Es soll überprüft werden, welcher Typ von NoSQL-Datenbanken am besten für die Datenspeicherung eines solchen Systems geeignet ist. Dafür wird die entwickelte Methode aus Kapitel 4 verwendet. Es stellt sich heraus, dass das zu untersuchende vernetzte Produktionssystem die gleiche Gewichtung der Bewertungskriterien fordert, wie die in Kapitel 4.3.3 bestimmte Gewichtung. Im nächsten Schritt wird die Nutzwertanalyse durchgeführt, dessen Ergebnisse in Tabelle 5.1 zu erkennen sind.

Tabelle 5.1: Nutzwertanalyse - Fahrzeugmontage

		Schlüssel-Wert		Spaltenfamilien		Dokument		Graph	
Kriterien	G	EG	TNW	EG	TNW	EG	TNW	EG	TNW
Ausfalltoleranz	13	2	26	5	65	4	52	3	39
Funktionsumfang	6	2	12	4	24	5	30	3	18
Konsistenz-	6	3	18	3	18	3	18	5	30

sicherung									
Leistungsfähigkeit (reads)	13	5	65	4	52	5	65	4	26
Leistungsfähigkeit (writes)	6	5	30	4	24	3	18	3	18
Realisierung von Beziehungen	12	1	12	3	36	4	48	5	60
Schemaflexibilität	12	3	36	3	36	5	60	4	48
Skalierbarkeit (reads)	13	5	65	4	52	4	52	4	52
Skalierbarkeit (writes)	6	5	30	4	24	3	18	3	18
Verfügbarkeit	13	2	26	5	65	4	52	4	52
Nutzwert	100		320		396		413		361
Rangfolge		4		2		1		3	

Aus diesen Ergebnissen lässt sich deuten, dass die Dokument-Datenbanken durch ihre hohe Leistungsfähigkeit, in der Bearbeitung von Leseanfragen und der Möglichkeit, diese Anfragen zu skalieren, sehr gut für den Einsatz in vernetzten Produktionssystemen geeignet sind. Dieser Typ besitzt den Vorteil, dass er sehr flexibel im Schema ist und somit viele unterschiedliche Datentypen abspeichern kann. Außerdem erfüllt er die Kriterien der Ausfalltoleranz und der Verfügbarkeit, die ebenfalls von vernetzten Produktionssystemen gefordert werden. Den nächstbesten Datenbanktyp stellen nach dieser Methode die Spaltenfamilien-Datenbanken dar, die keine hohe Schemaflexibilität besitzen und deshalb hinter den Dokument-Datenbanken einzuordnen sind. Auf Rang drei werden die Graphdatenbanken gelistet. Bei diesem Datenbanktyp lassen sich besonders gut Beziehungen realisieren. Allerdings besitzen sie keine hohe Ausfalltoleranz, was bei vernetzten Produktionseigenschaften jedoch unerlässlich ist. Da bei vernetzten Produktionssystemen auf die Konsistenz nach dem ACID-Prinzip verzichtet werden kann und dafür eine lockerere Konsistenzforderung zum Einsatz kommt, sind Graphdatenbanken durch ihren hohen Erfüllungsgrad der Konsistenz jedoch nicht besser für die vorliegende Anwendung geeignet. Auf dem letzten Rang sind die Schlüssel-Wert-Datenbanken. Sie besitzen zwar eine hohe Leistungsfähigkeit und Skalierbarkeit, haben dafür aber Schwachstellen in den relevanten Kriterien der Ausfalltoleranz, Verfügbarkeit und Schemaflexibilität.

5.2 Fahrerlose Transportsysteme

In einer Produktionshalle eines Automobilherstellers befindet sich die Logistik im Erdgeschoss und die Montage im Obergeschoss. Die autonomen Fertigungsanlagen am Montageband melden eigenständig Bedarf an, wenn neue Bauteile benötigt werden. Der Teiletransport wird von fahrerlosen Transportsystemen übernommen, die über einen Aufzug Zugang zu beiden Etagen haben. Die FTS navigieren in der Halle ohne physische Leitspur, da sie ihren Standort über mehrere Sensoren erhalten und diesen laufend mit dem Hallenlayout vergleichen. Sie haben zusätzlich die Möglichkeit, durch einen optischen Sensor naheliegende Gegenstände zu erkennen und können deshalb Hindernissen ausweichen. Ein erkanntes Hindernis wird unverzüglich anderen FTS gemeldet, die sich in der Nähe befinden, sodass sie früh genug ausweichen können. Es herrscht somit eine Schwarm-Intelligenz unter den eingesetzten FTS. Wenn ein neuer Auftrag durch eine Fertigungsanlage ausgeschrieben wird, können ihn alle FTS lesen und nach einem Vergleich der Positionen und Auftragslage jedes einzelnen FTS wird das FTS bestimmt, das den Auftrag am effektivsten umsetzen kann. Ein weiterer Faktor, der über die Verteilung des Auftrags entscheidet, ist die verbleibende Reichweite der FTS, da sie aus Gründen der Nachhaltigkeit mit einem Elektromotor betrieben werden. Ein FTS kann sich selbst den Aufzug rufen, sodass es sich ohne Zuhilfenahme menschlicher Aktionen zwischen den Stockwerken bewegen kann. Um die Effizienz zu erhöhen, fährt ein Aufzug nicht für ein einziges FTS, sondern transportiert mehrere. Eine Ausnahme kann gemacht werden, wenn ein Eilauftrag bearbeitet werden muss, da es sonst zum Bandstopp kommen würde. Es soll ein NoSQL-Datenbanktyp gefunden werden, der für die Datenspeicherung eines derartigen Systems geeignet ist.

Aufgrund der veränderten Anforderungen kann die Gewichtung aus Kapitel 4.3.4 nicht mehr verwendet werden. Am wichtigsten ist das Kriterium der Realisierung von Beziehungen, da die FTS die Positionen, Auftragslagen und Akkustände der anderen FTS kennen müssen, um zu entscheiden, wer einen ausgeschriebenen Auftrag annimmt. Eine weitere Eigenschaft, die in einem hohen Maße gefordert wird, ist die der Konsistenz. Bei der Standortüberprüfung müssen beispielsweise die Positionen der anderen FTS berücksichtigt werden, um Unfälle zu vermeiden. Aus diesem Grund müssen Daten immer in einem konsistenten Zustand sein. Außerdem wird dadurch verhindert, dass ein Auftrag doppelt angenommen wird. Die Kriterien der Ausfalltoleranz und Verfügbarkeit sind ebenfalls sehr wichtig, da es bei Nichterreichen zu einem Stillstand der Produktion kommen könnte. Eine vergleichbare Relevanz besitzen der Leistungsfähigkeit und Skalierbarkeit bezüglich Leseanfragen, da in diesem Anwendungsfall die Daten überwiegend abgefragt und nicht verändert werden. Darum werden keine hohen Forderungen an die Leistungsfähigkeit und Skalierbarkeit bezüglich Schreibanfragen gestellt. Beim Anfragen von Daten werden keine komplizierten Funktionen gestellt, weshalb das Kriterium des Funktionsumfangs ebenfalls nicht relevant ist. Das unrelevanteste Kriterium ist jedoch das der geforderten Schemaflexibilität, da die verwendeten Daten eine simple Struktur besitzen, wie beispielsweise

Koordinaten zur Positionsbestimmung oder die verbleibende Akkulaufzeit. Die Gewichtung und die damit einhergehende Nutzwertanalyse werden in Tabelle 5.2 veranschaulicht.

Tabelle 5.2: Nutzwertanalyse - Fahrerlose Transportsysteme

		Schlüssel-Wert		Spaltenfamilien		Dokument		Graph	
Kriterien	G	B	TNW	Bew.	TNW	Bew.	TNW	Bew.	TNW
Ausfalltoleranz	12	2	24	5	60	4	48	3	36
Funktionsumfang	6	2	12	4	24	5	30	3	18
Konsistenz-sicherung	14	3	42	3	42	3	42	5	70
Leistungsfähigkeit (reads)	12	5	60	4	48	5	60	4	48
Leistungsfähigkeit (writes)	6	5	30	4	24	3	18	3	18
Realisierung von Beziehungen	16	1	16	3	48	4	62	5	80
Schemaflexibilität	4	3	12	3	12	5	20	4	16
Skalierbarkeit (reads)	12	5	60	4	48	4	48	4	48
Skalierbarkeit (writes)	6	5	30	4	24	3	18	3	18
Verfügbarkeit	12	2	24	5	60	4	48	4	48
Nutzwert	100		310		390		394		400
Rangfolge		4		3		2		1	

Das Ergebnis unterscheidet sich deutlich von dem aus Kapitel 5.1, da die Graphdatenbanken den höchsten Nutzwert erreichen. Dies lässt sich allerdings durch die hohe Forderung nach der Realisierung von Beziehungen und Konsistenz begründen. An zweiter Stelle kommen die Dokument-Datenbanken, da sie ebenfalls die Möglichkeit bieten, Beziehungen zu realisieren. Schlüssel-Wert-Datenbanken sind zwar sehr leistungsfähig in der Bearbeitung von Lese- und Schreibfragen und deren Skalierbarkeit, Allerdings belegen sie durch die niedrige Ausfalltoleranz und Verfügbarkeit wie schon in Kapitel 5.1 den letzten Platz. Auf Rang drei liegen die Spaltenfamilien-Datenbanken, die in etwa den gleichen Nutzwert erreichen wie die Dokument-Datenbanken. Da es bei diesem Verfahren einen relativ großen subjektiven Bewertungsspielraum gibt, kann man bei solch kleinen Differenzen, wie zwischen den Spaltenfamilien- und den Dokument-Datenbanken, nicht festlegen, wer für den Anwendungsfall besser geeignet ist. Dieses Beispiel sollte aufzeigen wie mit der Methode umzugehen ist, wenn unterschiedliche Anforderungen von einem vernetzten Produktionssystem gestellt werden.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Methode zur Prüfung der Eignung von NoSQL-Datenbanksystemen zur Datenspeicherung in vernetzten Produktionssystemen der Automobilindustrie entwickelt. Die Theorie zu den vernetzten Produktionssystemen beschreibt den aktuellen Wandel zu immer stärker vernetzten Systemen. Diese werden benötigt, um mit den Herausforderungen der Automobilindustrie umgehen zu können und um die gestiegenen Kundenbedürfnisse befriedigen. Mit einer wachsenden Vernetzung werden die Prozesse zwar immer leistungsfähiger, doch der hohe Grad der Vernetzung und die dadurch entstehenden Big Data-Anwendungen bringen viele weitere Herausforderungen.

Es besteht der Bedarf nach einer Datenbanktechnologie, mit der sich umfangreiche, unstrukturierte Daten aus verschiedenen Quellen effizient bearbeiten lassen. Die für lange Zeit marktbeherrschenden relationalen Datenbanken werden diesem Problem nicht mehr gerecht. Die Lösung bieten die NoSQL-Datenbanktechnologien, die sich durch ihre massiv verteilte Datenhaltung auf ein großes Cluster und der damit einhergehenden Eigenschaft auszeichnet, große Datenmengen performant zu speichern und zu verarbeiten. Diese neue Datenbanktechnologie lässt sich in vier Kernmodelle unterteilen, die verschiedene Vorteile haben und für unterschiedliche Anwendungen gebraucht werden. Es wird zwischen Schlüssel-Wert-Datenbanken, Spaltenfamilien-Datenbanken, Dokument-Datenbanken und Graphdatenbanken unterschieden. Damit bei der Einführung von dieser Datenbanktechnologie in vernetzten Produktionssystemen der richtige Typ gewählt wird und somit der Betrieb noch performanter gestaltet werden kann, wird eine Methode zur Eignungsprüfung entwickelt, deren Ergebnis eine fundierte Grundlage für die finale Entscheidungsfindung bieten soll.

Es wird sich im folgenden Schritt mit den Grundlagen der Technologiebewertung und einigen unterschiedlichen Verfahren befasst, um ein geeignetes Verfahren für die zu erstellende Methode zu finden. Dabei ist zu berücksichtigen, dass ein qualitatives Verfahren zu wählen ist, da sich die Datenbanken nicht durch quantitative, beziehungsweise monetäre Werte beschreiben lassen. Die Wahl fällt auf das Bewertungsverfahren der Nutzwertanalyse, da sich dieses Verfahren zur Beurteilung von mehreren Alternativen in Hinblick auf komplexe Problemstellungen eignet. Zuerst wird das Problem durch die Wahl von einer Anzahl relevanter Bewertungskriterien in mehrere Teilprobleme heruntergebrochen. Durch die verschiedenen Kriterien lassen sich die unterschiedlichen Eigenschaften der Datenbanktypen umfassend beschreiben. Anschließend wird jedes Kriterium in Hinblick auf die Ausprägung bei jedem Datenbanktyp analysiert und es wird ein Erfüllungsgrad vergeben der beschreibt, inwieweit ein Datenbanktyp ein Kriterium erfüllt. Als letzter Schritt werden die Kriterien mit einem Gewichtungsfaktor versehen, da nicht jedes Kriterium für die vernetzten Produktionssysteme in der Automobilindustrie von gleicher Bedeutung ist. Als Ergebnis haben die Dokument-Datenbanken den höchsten Nutzenwert und sind nach der verwendeten Methode am besten für den Einsatz in vernetzten

Produktionssystemen in der Automobilindustrie geeignet. Allerdings bietet dieses Ergebnis lediglich eine Handlungsempfehlung und Verfahren wie das Polyglot Persistence, der Kombination mehrerer Datenbanktypen für ein Problem, um die Vorteile zu kombinieren, werden nicht berücksichtigt. Es sei außerdem angemerkt, dass die Festlegung des Erfüllungsgrades und der Gewichtung durch Erfahrungen erfolgt und somit einen relativ großen subjektiven Spielraum bietet. Die Gewichtung wurde aufgrund der allgemeinen Charakteristika von vernetzten Produktionssystemen bestimmt. Deshalb muss vor jeder Anwendung der Methode geprüft werden, ob durch besondere Anforderungen des Zielsystems gegebenenfalls eine Anpassung der Gewichtung erfolgen muss.

Zur Validierung wird die Methode auf zwei Fallbeispiele angewandt. Beim ersten Anwendungsfall wird ein vernetztes Produktionssystem beschrieben, das die gleiche Gewichtung besitzt, wie in der Methodenentwicklung erarbeitet. Der zweite Anwendungsfall wird ein Industriebeispiel als Basis genommen und durch weitere Eigenschaften ergänzt. Dabei verändern sich die Anforderungen, weshalb beim zweiten Beispiel die Gewichtungsfaktoren angepasst werden müssen. Die Ergebnisse werden bei beiden Fallbeispielen interpretiert.

Die NoSQL-Datenbanktechnologie ist für viele Anwendungsgebiete als effektivere Alternative zu den relationalen Datenbanken zu verwenden. Wenn sich diese Technologie vollkommen etabliert und vielfach verwendet wird, werden sicherlich viele Prozesse effizienter ablaufen. Jedoch sind neue Technologien oftmals mit einem Risiko verbunden, weshalb Unternehmen diese Datenbanktypen möglicherweise noch nicht als Lösungsalternative in Betracht ziehen. Es wäre wünschenswert, wenn die Ergebnisse dieser Arbeit dazu führen, dass NoSQL-Datenbanktechnologien als Alternative zu den relationalen Datenbanken in den Entscheidungsprozess mit eingeführt würden. In Zukunft wird es spannend die Entwicklung der Datenbanktechnologien und der vernetzten Produktionssysteme zu beobachten. Dabei sollte man sicherlich neue Technologien wie Blockchain oder NewSQL verfolgen oder die Forschungsergebnisse im Bereich der vernetzten Produktionssysteme, beispielsweise durch den Forschungscampus ARENA2036, im Blick behalten.

7 Abkürzungsverzeichnis

A – Availability

AP – Kombination aus Availability und Partition Tolerance

C – Consistency

CA – Kombination aus Consistency und Availability

CP – Kombination aus Consistency und Partition Tolerance

CPPS – Cyber-Physische Produktionssysteme

CPS – Cyber-Physical System

CPU – Central Processing Unit

DBMS – Database Management System

Dokument – Dokument-Datenbank

EG – Erfüllungsgrad

FTF – Fahrerlose Transportfahrzeuge

FTS – Fahrerlose Transportsysteme

G – Gewichtsfaktor

Graph – Graphdatenbank

P – Partition Tolerance

RAM – Random-Access Memory

Schlüssel-Wert – Schlüssel-Wert-Datenbank

Spaltenfamilien – Spaltenfamilien-Datenbank

SQL – Structured Query Language

TNW – Teilnutzwert

TPS – Toyota Produktionssystem

YCSB – Yahoo! Cloud Serving Benchmark

8 Abbildungsverzeichnis

9 Tabellenverzeichnis

10 Literaturverzeichnis

Affenzeller, P.; Hartlieb, E.; Willmann, R.: Industrie 4.0 – Evaluierung der Relevanz für Unternehmen mit physischen Angeboten. In: Granig, P.; Hartlieb, E.; Heiden, B. (Hrsg.): Mit Innovationsmanagement zu Industrie 4.0. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2018, S. 83-96

Bauernhansl, T.: Die Vierte Industrielle Revolution – Der Weg in ein wertschaffendes Produktionsparadigma. In: Bauernhansl, T.; ten Hompel, M.; Vogel-Heuser, B. (Hrsg.): Industrie 4.0 in Produktion, Automatisierung und Logistik. Wiesbaden: Springer Fachmedien, 2014, S. 5-35

Bischoff, O.: Big Data in der industriellen Fertigung. In: VDI-Z 159, 2017, Nr. 4, S. 32 – 34

Brewer, E.: Keynote – towards robust distributed systems. In: Proceedings of the 19th ACM Symposium on Principles of Distributed Computing, Portland, 2000

Celko, J.: Joe Celko's complete guide to NoSQL. Waltham, Massachusetts: Elsevier Inc., 2014

Celko, J.: Joe Celko's SQL for smarties, 5. Auflage. Waltham, Massachusetts: Elsevier Inc. 2015

Codd, E. F.: Relational Database: A practical foundation for productivity. In: Communications of the ACM Volume 25, Issue 2, New York, 1982, S. 107-117

Cooper, B. F.; Silberstein, A.; Tam, E.; Ramakrishnan, R.; Sears, R.: Benchmarking Cloud Serving Systems with YCSB. In: Proceedings of the 1st ACM symposium on Cloud computing, Indiana, Indianapolis, 2010, S. 143-154

Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation, San Francisco, 2004, S. 137-149

Diesner, M.: Funktionale Vernetzung und der „Digital Thread". In: VDI-Z 160, 2018, Nr. 10, S. 46-47

Fasel, D.: Übersicht über NoSQL-Technologien und -Datenbanken. In: Fasel, D.; Meier, A. (Hrsg.): Big Data – Grundlagen, Systeme und Nutzungspotenziale. Wiesbaden: Springer Fachmedien, 2016, S. 109-137

Fasel, D.; Meier, A.: Was versteht man unter Big Data und NoSQL?. In: Fasel, D.; Meier, A. (Hrsg.): Big Data – Grundlagen, Systeme und Nutzungspotenziale. Wiesbaden: Springer Fachmedien, 2016, S. 3-16

Frank, H.; Riess, M.: Cyber-Physische Produktionssysteme – Produktivitäts- und Flexibilitätssteigerung durch die Vernetzung intelligenter Systeme in der Fabrik. In: Reinhart, G.; Scholz-Reiter, B.; Wahlster, W.; Wittenstein, M.; Zühlke, D. (Hrsg.): Intelligente Vernetzung in der Fabrik. Stuttgart: Fraunhofer Verlag, 2015, S. 9-33

Göpfert, I.; Schulz, M.; Wellbrock, W.: Trends in der Automobillogistik. In: Göpfert, I.; Braun, D.; Schulz, M.: Automobillogistik, 3. Auflage. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, S. 1-26

Gronau, N.: Der Einfluss von Cyber-Physical Systems auf die Gestaltung von Produktionssystemen. In: Kersten, W.; Koller, H.; Lödding, H. (Hrsg.): Industrie 4.0 – Wie intelligente Vernetzung und kognitive Systeme unsere Arbeit verändern. Berlin: GITO mbH Verlag, 2014, S. 279-295

Haag, C.; Schuh, G.; Kreysa, J.; Schmelter, K.: Technologiebewertung. In: Schuh, G. (Hrsg.); Klappert, S. (Hrsg.): Technologiemanagement, 2. Auflage. Berlin, Heidelberg: Springer-Verlag, 2011, S. 309-366

Hoffmeister, W.: Investitionsrechnung und Nutzwertanalyse – Eine entscheidungsorientierte Darstellung mit vielen Beispielen und Übungen, 2. Auflage. Berlin: Berliner Wissenschafts-Verlag GmbH, 2008

Huber, W.: Industrie 4.0 kompakt – Wie Technologie unsere Wirtschaft und unsere Unternehmen verändern. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2018

Huber, D. und Kaiser, T.: Wie das Internet der Dinge neue Geschäftsmodelle ermöglicht. In: Reinheimer, S. (Hrsg.): Industrie 4.0 – Herausforderungen, Konzepte und Praxisbeispiele. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2017, S. 17-26

Hüttenrauch, M.; Baum, M.: Effiziente Vielfalt – Die dritte Revolution in der Automobilindustrie. Berlin, Heidelberg: Springer-Verlag, 2008

Jurney, R.: Agile Data Science 2.0. Sebastopol: O'Reilly Media, Inc., 2017

Kellner, F.; Lienland, B.; Lukesch, M.: Produktionswirtschaft. Berlin: Springer-Verlag GmbH Deutschland, 2018

Kemper, A.; Eickler, A.: Datenbanksysteme – Eine Einführung, 10. Auflage. Berlin/Boston: Walter de Gruyter GmbH, 2015

King, S.: Bit Data – Potential und Barrieren der Nutzung im Unternehmenskontext. Wiesbaden: Springer Fachmedien Wiesbaden, 2014

Klappert, S.; Schuh, G.; Aghassi, S.: Einleitung und Abgrenzung. In: Schuh, G. (Hrsg.); Klappert, S. (Hrsg.): Technologiemanagement, 2. Auflage. Berlin, Heidelberg: Springer-Verlag, 2011, S. 5-10

Kunze, D. M.; Blanek, H.-D.; Simons, D.: Nutzwertanalyse als Entscheidungshilfe für Planungsträger. Frankfurt am Main: Kuratorium für Technik und Bauwesen in der Landwirtschaft, 1969

McCreary, D.; Kelly, A.: Making Sense of NoSQL. Shelter Island: Manning Publications Co., 2014

Meier, A.: Datenmanagement mit SQL und NoSQL. In: Fasel, D.; Meier, A. (Hrsg.): Big Data – Grundlagen, Systeme und Nutzungspotenziale. Wiesbaden: Springer Fachmedien, 2016, S. 17-38

Meier, A.: Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL & Co.. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2018

Meier, A.; Kaufmann, M.: SQL- & NoSQL-Datenbanken, 8. Auflage. Berlin, Heidelberg: Springer-Verlag, 2016

Müller, S.: Erweiterung des Data Warehouse um Hadoop, NoSQL & Co. In: Fasel, D.; Meier, A. (Hrsg.): Big Data – Grundlagen, Systeme und Nutzungspotenziale. Wiesbaden: Springer Fachmedien, 2016, S. 139-158

Müller, A.; Schröder, H.; von Thienen, L.: Lean IT-Management – Was die IT aus Produktionssystemen lernen kann. Wiesbaden: Gabler Verlag, 2011

Neubauer, W.: Trends in der Automobilindustrie. In: Rudow, B.; Neubauer, W.: Trends in der Automobilindustrie. München: Oldenbourg Wissenschaftsverlag GmbH, 2012, S. 1-14

Rumpelt, T.: Die Smart Factory lernt Laufen. In: Automobil Industrie Ausgabe 207, 2016, S. 8-10

Saake, G.; Sattler, K.-U.; Heuer, A.: Datenbanken – Konzepte und Sprachen, 6. Auflage. Frechen: mitp Verlags GmbH & Co. KG, 2018

Sadalage, P. J.; Fowler, M.: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. New Jersey: Pearson Education, Inc., 2013

Schallow, J.; Hengstebeck, A.; Deuse, J.: Industrie 4.0 – Eine Bestandsaufnahme. In: Wagner, R. M. (Hrsg.): Industrie 4.0 für die Praxis. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2018, S. 15-28

Schicker, E.: Datenbanken und SQL, 5. Auflage. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2017

Schlick, J.; Stephan, P.; Loskyll, M.; Lappe, D.: Industrie 4.0 in der praktischen Anwendung. In: Bauernhansl, T.; ten Hompel, M.; Vogel-Heuser, B. (Hrsg.): Industrie 4.0 in Produktion, Automatisierung und Logistik. Wiesbaden: Springer Fachmedien, 2014, S. 56-84

Schuh, G.; Klappert, S.; Moll, T.: Ordnungsrahmen Technologiemanagement. In: Schuh, G. (Hrsg.); Klappert, S. (Hrsg.): Technologiemanagement, 2. Auflage. Berlin, Heidelberg: Springer-Verlag, 2011, S. 11-31

Seidemann, C.: Monitoring von Cyber-Physischen Produktionssystemen. In: Reinhart, G.; Scholz-Reiter, B.; Wahlster, W.; Wittenstein, M.; Zühlke, D. (Hrsg.): Intelligente Vernetzung in der Fabrik. Stuttgart: Fraunhofer Verlag, 2015, S. 305-313

Soder, J.: Use Case Production. In: Vogel-Heuser, B.; Bauernhansl, T.; ten Hompel, M. (Hrsg.): Handbuch Industrie 4.0 Bd.1 – Produktion, 2. Auflage. Berlin: Springer-Verlag GmbH Deutschland, 2017, S. 3-25

Steegmüller, D.; Zürn, M.: Wandlungsfähige Produktionssysteme für den Automobilbau der Zukunft. In: Bauernhansl, T.; ten Hompel, M.; Vogel-Heuser, B. (Hrsg.): Industrie 4.0 in Produktion, Automatisierung und Logistik. Wiesbaden: Springer Fachmedien, 2014, S. 103-119

Tiwari, S.: Professional NoSQL. Indianapolis: John Wiley & Sons, Inc., 2011

Unterstein, M.; Matthiessen, G.: Relationale Datenbanken und SQL in Theorie und Praxis, 5. Auflage. Berlin, Heidelberg: Springer-Verlag, 2012

Vaish, G.: Getting Started with NoSQL. Birmingham: Packt Publishing, 2013

Wagner, R.: Einleitung: Industrie 4.0 und Digitalisierung – Erfolgspotenziale für Unternehmen. In: Wagner, R. M. (Hrsg.): Industrie 4.0 für die Praxis. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2018, S. 3-13

Westkämper, E.; Löffler, C.: Strategien der Produktion. Berlin, Heidelberg: Springer-Verlag, 2016

Wibbe, C.; Rohde, D.: Industrie im automobilen Umfeld. In: Göpfert, I.; Braun, D.; Schulz, M.: Automobillogistik, 3. Auflage. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, S. 37-52

Wiese, L.: Advanced Data Management. Berlin/Boston: Walter de Gruyter GmbH, 2015

Wildeman, H.: Entwicklungslinien der Produktionssysteme in der Automobilindustrie. In: Göpfert, I.; Braun, D.; Schulz, M.: Automobillogistik, 3. Auflage. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, S. 161-184

Winkelhake, U.: Die digitale Transformation der Automobilindustrie. Berlin: Springer-Verlag GmbH Deutschland, 2017

Zangemeister, C.: Nutzwertanalyse in der Systemtechnik – Eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen, 4. Auflage. München: Wittemannsche Buchhandlung, 1976

11 Eidesstattliche Versicherung

Name, Vorname _____ Matr.-Nr. _____

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit / Masterarbeit mit dem Titel _____ selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum _____ Unterschrift _____

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt und/oder eine falsche eidesstattliche Versicherung abgibt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum _____ Unterschrift _____